Mathematische Sprachtheorie

Marcus Kracht
II. Mathematisches Institut
Freie Universität Berlin
Arnimallee 3
D – 14195 Berlin
kracht@math.fu-berlin.de

8. Januar 2010

Vorwort

Der vorliegende Text ist aus Vorlesungen an der FU Berlin über formale Sprachen sowie einer Vorlesung mit dem Titel ‘Mathematische Sprachtheorie’ entstanden. Ich möchte dem Fachbereich Mathematik/Informatik für die stets guten Arbeitsbedingungen danken.

Ich danke Helmut Alt, Benjamin Fabian, Stefanie Gehrke, Timo Hanke, Franz Koniecny, Thomas Kosiol, Zsuzsanna Lipták, Jens Michaelis, Stefan Salinger, Harald Stamm und Ngassa Tchao für ihre Mithilfe am Zustandekommen dieses Manuskripts.

Bielefeld, den 8. Januar 2010

Marcus Kracht

Einleitung

Dieses Buch ist — wie der Titel sagt — ein Buch über mathematische Sprachtheorie, das heißt, über die Beschreibung von Sprache und Sprachen mit mathematischen Methoden. Es wendet sich an Studenten der Mathematik, Informatik, Computerlinguistik und Sprachwissemschaft, und an alle, die sich mit dem Aufbau von Sprache befassen wollen oder müssen. Es ist vorwiegend ein mathematisches Buch; es kann und will die Einführungen in die Sprachwissenschaft sowie speziell der Syntax nicht ersetzen. Wer sich damit vertraut machen will, dem seien die Bücher von Lyons [35], von Grewendorf, Hamm und Sternefeld [20] und von von Stechow und Sternefeld [50] empfohlen. Wir besprechen hier keine der syntaktischen Theorien im Detail. Der Leser wird allerdings — so ist zu hoffen — nach dem Studium dieses Buches in die Lage versetzt, jede dieser Theorien tiefer als bisher zu verstehen.

Ich ziehe keinen Trennungsstrich zwischen formalen Sprachen und natürlichen Sprachen. Diese werden hier völlig gleichbehandelt. Es ist ja völlig unerheblich, ob eine Sprache künstlichen oder natürlichen Ursprungs ist, wenn wir ihren Aufbau verstehen wollen. Auch wenn insbesondere Noam Chomsky betont hat, daß es einen fundamentalen Unterschied zwischen natürlichen und nicht–natürlichen Sprachen gibt, so muß sich dieser natürlich in der formalen Beschreibung erweisen können. Insofern sollte diese Unterschied, wenn er denn existiert, nicht zur Grundlage eines solchen Buches werden. Das vorliegende Buch ist ebenfalls keine Einführung in die formalen Sprachen im herkömmlichen Sinne, sondern eher eine mathematische Einführung in die Sprachwissenschaft. Man wird viele Themen und Ergebnisse aus der Theorie der formalen Sprachen vermissen. Dies liegt einfach daran, daß sie erfahrungsgemäß keine so wesentliche Rolle spielen. Auf der anderen Seite geht dieses Buch über die sonst üblichen Einführungen in einigen Punkten hinaus. Der wichtigste Punkt ist, daß wir Sprachen nicht einfach nur als Mengen wohlgeformter Zeichenketten auffassen, sondern als *Zeichensysteme*. Dies kommt der Sprachwirklichkeit natürlich sehr viel näher. Dieser Ansatz wird in der Einleitung kurz vorgestellt und dann in seinen konkreten Einzelheiten in Kapitel 3 eingeführt.

Ein **Zeichen** ist ein Tripel

$$\sigma = \langle E, T, M \rangle$$

Dabei ist E der **Exponent** von σ, üblicherweise eine Zeichenkette, T der **syntaktische Typ** von σ, und M die **Bedeutung**. Dadurch wird jetzt eine Zeichenkette an eine Menge von Bedeutungen gebunden; E bedeutet in S genau dann M, wenn es einen Typ T gibt derart, daß $\langle E, T, M \rangle \in S$. Die Aufgabe von Sprachtheorie

ist so gesehen nicht einfach nur zu sagen, welche Zeichenketten als Exponenten gewisser Zeichen auftreten, wie dies zum Beispiel in der Theorie der Ersetzungssysteme wie auch in der Grammatiktheorie die Regel ist, sondern welche Bedeutung eine gegebene Zeichenkette haben kann. Im Zentrum steht das sogenannte *Kompositionalitätsprinzip*, welches in seiner schwächsten Formulierung lautet, daß man die Bedeutungen einer Zeichenkette auf die Weise gewinnt, daß man ihre syntaktischen Ableitungen homomorph auf die Semantik abbildet. Kompositionalität wird im Kapitel 3 eingeführt und in seinen Folgen ausführlich diskutiert. Wir werden dort auch auf die Montague Semantik zu sprechen kommen, allerdings wird die Diskussion relativ kurz ausfallen. Ohnehin gibt es zu diesem Thema so viel gute Literatur, daß man den Leser getrost dorthin verweisen kann. Wir verweisen hier auf die Einführungen von Dowty, Wall und Peters [10] sowie auf das Buch des Autorenkollektivs Gamut [12]. Ein **Zeichensystem** ist eine partielle Algebra von Zeichen. Diese bedeutet, daß sie ist ein Paar $\langle \Sigma, M \rangle$, wo Σ eine Menge von Zeichen ist und M eine endliche Menge von sogenannten **Kompositionsmodi**. Standardmäßig nimmt man an, daß M nur einen einzigen Modus enthält, eine zweistellige Operation $\bullet$, mit welcher wir aus zwei Zeichen σ_1 und σ_2 ein neues Zeichen $\sigma_1 \bullet \sigma_2$ herstellen können. Diese Operation ist nicht immer definiert, wie man sich bald überzeugen kann. Die Wirkung von $\bullet$ muß nun auf den drei Komponenten der Zeichen definiert werden. Wir führen dies an einem einfachen Beispiel vor. Angenommen, wir haben die Zeichen

$$\begin{array}{lcl} \texttt{`rennt'} & = & \langle \texttt{rennt}, v, f \rangle \\ \texttt{`Paul'} & = & \langle \texttt{Paul}, n, p \rangle \end{array}$$

Hierbei sind v und n die syntaktischen Typen *(intransitives) Verb* und *Eigennamen*. p ist eine Konstante, welche ein Individuum bezeichnet (nämlich Paul), und f eine Funktion von Individuen nach $\{0, 1\}$. Auf der Ebene der Exponenten wählen wir die Wortverkettung (d.h. Verkettung mit eingeschobenem Leerzeichen); auf der Ebene der Bedeutungen die Funktionalapplikation. Schließlich sei $\bullet_t$ eine partielle Funktion, welche nur für $n \bullet_t v$ definiert ist und den Wert t ergibt. Jetzt setzen wir

$$\langle E_1, T_1, M_1 \rangle \bullet \langle E_2, T_2, M_2 \rangle := \langle E_1\ E_2, T_1 \bullet_1 T_2, M_2(M_1) \rangle$$

Dann ist `'Paul'` $\bullet$ `'rennt'` ein Zeichen, und es hat die folgende Form:

$$\texttt{`Paul'} \bullet \texttt{`rennt'} := \langle \texttt{Paul rennt}, t, f(p) \rangle$$

Wir sagen, dieser Satz ist wahr genau dann, wenn $f(p) = 1$; andernfalls sei er falsch. Übrigens ist `'Paul'` $\bullet$ `'Paul'` *kein* Zeichen. $\bullet$ ist also tatsächlich partiell.

Die Idee ist nun, daß die Zeichen zunächst einmal durch Strukturterme beschrieben werden, welche nichts weiter als variablenfreie Terme über der Signatur von M. Diese kann man durch eine kontextfreie Grammatik beschreiben. Dort ist also die Welt noch in Ordnung. Die Algebren der Exponenten, Typen oder Bedeutungen sind aber möglicherweise partiell, sodaß nicht jeder Term tatsächlich einem Zeichen entspricht. Zudem hat man es in der Regel bei den Exponenten nicht Zeichenketten und

bei ihrem Operationen nicht immer mit bloßer Verkettung zu tun, sodaß die entstehenden Sprachen selbst dann sehr viel komplizierter werden können als kontextfreie, wenn alle Operationen total sind. Bevor man all dies verstehen kann, ist es deswegen unerläßlich, sich mit Zeichenketten und Grammatiken im herkömmlichen Sinn zu beschäftigen. Dies tun wir in den ersten beiden Kapiteln. Wir führen Zeichenketten formal ein und definieren dann Semi–Thue Systeme und Grammatiken. Das erste Kapitel stellt die absolute Essenz dessen bereit, was man dazu wissen muß. In Kapitel 2 studieren wir dann reguläre und kontextfreie Sprachen im Detail. Insbesondere werden wir uns mit ihrer Charakterisierung durch Automatenklassen, Erkennung- und Analyse, Parsingkomplexität und Ambiguität befassen. Am Schluß beweisen wir den Satz von Parikh. Im Kapitel 3 beginnen wir mit Sprachen als Zeichensystemen. Zeichensysteme und -grammatiken werden in dem ersten Abschnitt definiert. Anschließend wenden wir uns einer besonderen Art von Zeichengrammatiken zu, den sogenannten Kategorialgrammatiken. Wir werden sowohl den Ajdukiewicz–Bar Hillel Kalkül wie auch den Lambek–Kalkül vorstellen, und von beiden zeigen, daß sie nur kontextfreie Sprachen erzeugen. Für den Lambek–Kalkül war dies eine lange offene Vermutung von Chomsky, welche erst Anfang der 90er Jahre durch Mati Pentus bestätigt worden ist. Im vierten Kapitel widmen wir uns den sogenannten PTIME Sprachen. Dies sind Sprachen, für das Parsingproblem in deterministisch polynomieller Zeit entschieden werden kann. Die Frage danach, ob natürliche Sprache kontextfrei sind oder nicht, galt bis in die 80er Jahre als entschieden: sie sind es in aller Regel nicht. Dann kippte der Konsens für einige Zeit, als gezeigt wurde, daß die meisten Konstruktionen, welche für anerkannt nicht–kontextfrei galten, allerdings doch kontextfrei sind. Dennoch stellte sich bald heraus, daß es genug Evidenz gegen die Kontextfreiheit gibt. Auf der Suche nach Eigenschaften, welche natürliche Sprache stets und immer haben, fand man unter anderem die PTIME–Klasse. Diese ist tatsächlich umfassend genug, daß — nach allem was man bisher weiß — alle bekannten Sprachen darin enthalten sind. Ferner existieren einige sehr interessante Charakterisierungen dieser Sprachklasse, zum Beispiel durch einfache Literalbewegungsgrammatiken, welche durch Annius Groenink vorgeschlagen worden sind. Das fünfte Kapitel ist dem logischen Ansatz zur Sprachbeschreibung gewidmet. Auch dieser ist im Wesentlichen in den 80er Jahren entstanden, und die Verbidung zu dem sogenannten *Constraint*–Programmieren ist sicherlich nicht von der Hand zu weisen. Es wurde vorgeschlagen, Grammatiken einfach als logische Theorien akzeptabler Sätze beziehungsweise akzeptabler Strukturen zu sehen. Dies ist insofern von dem Ansatz Chomsky's verschieden, als die Theorie in keinerlei Zusammenhang stehen muß zu einem System von erzeugenden Regeln, welche die zulässigen Strukturen oder Sätze bauen. Es stellt sich allerdings heraus, daß es sogar Algorithmen gibt, gewisse axiomatische Theorien in Grammatiken umzubauen; dies geht auf Arbeiten von Büchi von sowie Thatcher und Donner über Theorien in monadischer Logik zweiter Stufe zurück. Interessanterweise ist die Umkehrung, nämlich die Extraktion der Theorie aus den Regeln, ein subtiles Unterfangen. Je nach Beschreibungsstärke können mehr oder weniger Regeln in einer axiomatischen Theorie aufgefangen wer-

den. Dies führt zu einer rein logisch motivierten Komplexitätshierarchie, die indirekt natürlich auch etwas mit der Berechnungskomplexität zu tun hat. In den Kapiteln 4 und 5 wird der Leser mit allerhand Formalismen vetraut gemacht, welche in den letzten 20 Jahren in der Sprachwissenschaft vorgeschlagen worden sind.

Inhaltsverzeichnis

1 Grundlegende Strukturen **1**

1.1 Algebren und Strukturen 1

1.2 Halbgruppen und Zeichenketten 6

1.3 Grundlegendes aus der Sprachwissenschaft 17

1.4 Bäume 24

1.5 Ersetzungssysteme 33

1.6 Grammatik und Strukturbeschreibung 44

1.7 Turingmaschinen 58

2 Kontextfreie Sprachen **69**

2.1 Reguläre Sprachen 69

2.2 Normalformen für kontextfreie Grammatiken 77

2.3 Erkennung und Analyse 91

2.4 Ambiguität, Transparenz und Parsingstrategien 105

2.5 Der Satz von Parikh 120

2.6 Sind natürliche Sprachen kontextfrei? 129

3 Kategorialgrammatik und Formale Semantik **137**

3.1 Sprachen als Zeichensysteme 137

3.2 Der Syntaktische Typenkalkül 149
3.3 Der AB–Kalkül 161
3.4 Der Lambek–Kalkül 171
3.5 Der Satz von Pentus 176
3.6 Montague–Semantik 184
3.7 Anhang: Grundzüge des λ–Kalküls und der Kombinatorischen Logik . 191

4 PTIME Sprachen **201**

4.1 Mild–Kontextsensitive Sprachen 201
4.2 Literalbewegungsgrammatiken 212
4.3 Interpretierte Literalbewegungsgrammatiken 223
4.4 Diskontinuität 229
4.5 Adjunktionsgrammatiken 243
4.6 Indizierte Grammatiken 253
4.7 Kompositionalität 262

5 Linguistische Strukturen **277**

5.1 Kategorien 277
5.2 Axiomatische Klassen I: Zeichenketten 286
5.3 Einiges zu Phonologie und Silbenstruktur 299
5.4 Axiomatische Klassen II: Erschöpfend geordnete Bäume 308
5.5 Eine Detailstudie 317
5.6 Multiple Dominanz 318

Kapitel 1

Grundlegende Strukturen

1.1 Algebren und Strukturen

In diesem Abschnitt werden wir grundlegende Definitionen bereitstellen, welche im Folgenden immer wieder benötigt werden. Dazu gehören die Begriffe *Algebra* und *Struktur*. Diejenigen Leser, für die diese Begriffe völlig neu sind, sind aufgefordert, diesen Abschnitt beim ersten Lesen zu überspringen und nur dann zurückzukehren, wenn sie erste Anwendungen dieser Strukturen gesehen haben.

Wir setzen eine gewisse Vertrautheit mit der mathematischen Begriffswelt voraus, insbesondere den Umgang mit Mengen und üblichen Beweisverfahren wie vollständige Induktion. In der Mengenlehre definiert man Zahlen üblicherweise wie folgt.

$$\begin{array}{lcl} 0 & := & \varnothing \\ n+1 & := & \{k : k < n\} = \{0, 1, 2, \ldots, n-1\} \end{array}$$

Die Menge aller so konstruierbaren Zahlen, also der sogenannten *natürlichen Zahlen*, wird mit ω bezeichnet. Ihre Mächtigkeit heißt $\aleph_0$ (sprich: *Alef Null*). Zwar ist $\aleph_0$ auch die Menge ω, jedoch ist der Buchstabe ω eigentlich für den Wohlordnungstyp der natürlichen Zahlen reserviert. Ist M eine Menge, so bezeichnen wir mit $\wp(M)$ die Potenzmenge von M; diese ist die Menge aller Teilmengen von M. Ferner bezeichnet $|M|$ die Mächtigkeit von M. Diese ist im endlichen Fall eine natürliche Zahl, im allgemeinen eine Kardinalzahl. Ist $|M| \leq \aleph_0$, so heißt M **abzählbar**. Hat M die Mächtigkeit κ, so wird die Mächtigkeit von $\wp(M)$ mit 2^κ bezeichnet. $2^{\aleph_0}$ ist die Mächtigkeit der Menge aller Mengen natürlicher Zahlen. Es ist $2^{\aleph_0}$ strikt größer als $\aleph_0$. Mengen dieser Mächtigkeit sind also überabzählbar. Jedoch ist die Menge aller *endlichen* Mengen natürlicher Zahlen wieder abzählbar.

Sind M und N Mengen, so bezeichnet $M \times N$ die Menge aller Paare $\langle x, y \rangle$ mit $x \in$

M und $y \in N$. Man kann $\langle x, y\rangle$ definieren durch $\{x, \{x, y\}\}$, aber dies ist für unsere Zwecke meist unerheblich (siehe aber Abschnitt 3.7). Eine **Relation** von M nach N ist eine Teilmenge von $M \times N$. Wir schreiben $x\,R\,y$, falls $\langle x, y\rangle \in R$. Besonders interessant ist der Fall $M = N$. Eine Relation $R \subseteq M \times M$ heißt **reflexiv**, falls $x\,R\,x$ für alle $x \in M$; **symmetrisch**, falls aus $x\,R\,y$ folgt $y\,R\,x$. R heißt **transitiv**, falls aus $x\,R\,y$ und $y\,R\,z$ folgt $x\,R\,z$. Eine **Äquivalenzrelation** auf M ist eine reflexive, symmetrische und transitive Relation auf M. Ein Paar $\langle M, <\rangle$ heißt eine **geordnete Menge**, falls M eine Menge ist und $<$ eine transitive, irreflexive zweistellige Relation auf M. $<$ heißt eine **(strikte) Ordnung auf** M und M heißt dann auch **durch** $<$ **geordnet**. $<$ ist **linear**, falls für je zwei Elemente $x, y \in M$ gilt $x < y$ oder $x = y$ oder $y < x$. Eine **partielle Ordnung** ist eine Relation, welche reflexiv, transitiv und antisymmetrisch ist; letzteres bedeutet, daß aus $x\,R\,y$ und $y\,R\,x$ folgt $x = y$.

Ist $R \subseteq M \times M$ eine Relation, so bezeichnet $R^\smile := \{\langle x, y\rangle : y\,R\,x\}$ die sogenannte **konverse Relation** zu R. Ist S eine weitere Relation, so setze

$$\begin{array}{lcl} R \circ S & := & \{\langle x, y\rangle : \text{ für ein } z : x\,R\,z\,S\,y\} \\ R \cup S & := & \{\langle x, y\rangle : x\,R\,y \text{ oder } x\,S\,y\} \end{array}$$

Es ist $\Delta_M := \{\langle x, x\rangle : x \in M\}$ die sogenannte **Diagonale**. Nun sei

$$\begin{array}{lcl} R^0 & := & \Delta_M \\ R^{n+1} & := & R \circ R^n \\ R^+ & := & \bigcup_{0<i} R^i \\ R^* & := & \bigcup_{i\in\omega} R^i \end{array}$$

R^+ ist die kleinste transitive Relation, welche R enthält. Sie heißt deswegen auch die **transitive Hülle** von R. R^* ist die kleinste reflexive und transitive Relation, welche R enthählt.

Eine **Funktion** von M nach N ist eine Relation $f \subseteq M \times N$ derart, daß aus $x\,f\,y$ und $x\,f\,z$ folgt, daß $y = z$. Wir schreiben dann auch $y = f(x)$, falls $x\,f\,y$ und $f : M \to N$, falls f eine Funktion von M nach N ist. Schließlich schreiben wir $f : x \mapsto y$, falls $y = f(x)$. Ist $X \subseteq M$, so ist $f[X] := \{f(x) : x \in X\}$ das sogenannte **direkte Bild von** X **unter** f. Wir weisen darauf hin, daß zwischen $f(X)$ und $f[X]$ ein Unterschied besteht. Sei zum Beispiel $S : \omega \to \omega : x \mapsto x+1$. Dann ist gemäß der obenstehenden Definition von natürlichen Zahlen $S(4) = 5$ und $S[4] = \{1, 2, 3, 4\}$, da ja $4 = \{0, 1, 2, 3\}$. Eine Funktion heiß **injektiv**, falls für alle $x_1 \in M$ und alle $x_2 \in M$ genau dann $f(x_1) = f(x_2)$ gilt, wenn $x_1 = x_2$; f heißt **surjektiv**, falls für jedes $y \in N$ ein $x \in M$ existiert mit $y = f(x)$. f ist **bijektiv**, wenn f sowohl injektiv als auch surjektiv ist. Es sei $y \in N$ und $Y \subseteq N$; dann sei $f^{-1}(y) := \{x : f(x) = y\}$ sowie $f^{-1}[Y] := \{x : f(x) \in Y\}$. Ist f injektiv, so bezeichnet $f^{-1}(y)$ auch dasjenige x mit $f(x) = y$. Wir werden dafür sorgen, daß dies im Folgenden nicht zu Konfusionen Anlaß gibt.

Es sei M^n, $n \in \omega$, die Menge aller n–Tupel von Elementen aus M. Dies kann man wie folgt präzisieren.

$$\begin{array}{lcl} M^1 & := & M \\ M^{n+1} & := & M^n \times M \end{array}$$

Ferner ist $M^0 := 1 (= \{\varnothing\})$. Damit ist ein n–Tupel von Elementen aus M schlicht ein Element von M^n. Wir schreiben je nach Bedarf $\langle x_i : i < n \rangle$ oder $\langle x_0, x_2, \ldots, x_{n-1} \rangle$ für ein n–Tupel über M.

Eine n–**stellige Relation** auf M ist eine Teilmenge von M^n, eine n–**stellige Funktion** auf M ist eine Funktion $f : M^n \to M$. Hierbei ist $n = 0$ ausdrücklich erlaubt. Eine 0–stellige Relation ist demnach eine Teilmenge von 1, also entweder die leere Menge oder 1; eine 0–stellige Funktion auf M ist eine Abbildung $c : 1 \to M$. Diese bezeichnen wir auch als **Konstante**. Der **Wert** dieser Konstanten ist das Element $c(\varnothing)$. Es sei R eine n–stellige Relation und $\vec{x} \in M^n$. Dann schreiben wir anstatt $\vec{x} \in R$ auch $R(\vec{x})$.

Es sei nun F eine beliebige Menge und $\Omega : F \to \omega$. Das Paar $\langle F, \Omega \rangle$, oft schlicht durch Ω bezeichnet, heißt eine **Signatur** und F die Menge der **Funktionssymbole**.

Definition 1.1.1 *Es sei $\Omega : F \to \omega$ eine Signatur und A eine nichtleere Menge. Sei ferner Π eine Abbildung, welche jedem $f \in F$ eine $\Omega(f)$–stellige Funktion auf A zuordnet. Dann ist das Paar $\mathfrak{A} := \langle A, \Pi \rangle$ eine Ω–**Algebra**. Ω–Algebren werden im allgemeinen durch Frakturbuchstaben gekennzeichnet.*

Um nicht in der Notation zu ersticken, werden wir folgende allgemein übliche Bezeichnungsweise einführen. Ist $\mathfrak{A}$ eine Ω–Algebra, so bezeichnet $f^{\mathfrak{A}}$ die Funktion $\Pi(f)$. Anstatt also $\mathfrak{A}$ als das Paar $\langle A, \Pi \rangle$ zu notieren, wird jetzt $\mathfrak{A}$ durch $\langle A, \{f^{\mathfrak{A}} : f \in F\} \rangle$ bezeichnet. Der Leser sei jedoch davor gewarnt, daß die letztere Schreibweise zu Mißverständnissen Anlaß geben kann, da ja Funktionen gleicher Stelligkeit verschiedenen Funktionssymbolen zugeordnet werden können. Diese Probleme werden jedoch im Folgenden nicht auftreten.

Die Menge der Ω–Terme ist die kleinste Menge T, für die gilt:

$$\text{Ist } f \in F \text{ und } t_i \in T \text{ für alle } i < \Omega(f), \text{ so ist } f(t_0, \ldots, t_{\Omega(f)-1}) \in T.$$

Terme sind abstrakte Gebilde; sie nicht gleichzusetzen mit Funktionen noch mit den Zeichenketten, durch welche wir sie bezeichnen. Zunächst definieren wir die Stufe

eines Terms wie folgt. Ist $\Omega(f) = 0$, so ist $f()$ ein Terme der Stufe 0, welchen wir auch mit 'f' bezeichnen. Sind t_i, $i < \Omega(f)$, Terme der Stufe n_i, so ist $f(t_0, \ldots, t_{\Omega(f)-1})$ ein Term der Stufe $1 + max\{n_i : i < \Omega(f)\}$. Viele Beweise laufen über die Stufe von Termen, man spricht dann auch von *Induktion über den Termaufbau*. Induktiv wird die Gleichheit von Termen definiert.

> Zwei Terme u und v sind gleich, falls sie die gleiche Stufe haben, und ferner gilt:
>
> 1. u und v haben die Stufe 0, und es existiert ein $f \in F$ mit $u = v = f()$.
> 2. Es existiert ein $f \in F$, und Terme s_i, t_i, $i < \Omega(f)$, derart, daß $u = f(s_0, \ldots, s_{\Omega(f)-1})$ und $v = f(t_0, \ldots, t_{\Omega(f)-1})$ sowie $s_i = t_i$ für alle $i < \Omega(f)$.

Ein wichtiges Beispiel einer Ω–Algebra ist die sogennante *Termalgebra*. Wir wählen dazu eine beliebige Menge X von Symbolen. Dabei soll X disjunkt sein zu F. Die Signatur wird auf X ausgedehnt; und zwar soll jedes Symbol aus X die Stelligkeit 0 haben. Die Terme über dieser neuen Signatur werden als Ω–**Terme über** X bezeichnet. Ihre Menge bezeichnen wir mit $Tm_\Omega(X)$. Jeder Term wird nun zu einem Objekt der Algebra, und jedes Funktionssymbol zu einer Funktion. Es sei nämlich $\Pi(f)$ die folgende Funktion:

$$\Pi(f) : \langle t_i : i < \Omega(f)\rangle \mapsto f(t_1, \ldots, t_{\Omega(f)-1})$$

Dann ist $\langle Tm_\Omega(X), \Pi\rangle$ eine Ω–Algebra, genannt **die von X erzeugte Termalgebra**.

Definition 1.1.2 *Es seien $\mathfrak{A} = \langle A, \{f^{\mathfrak{A}} : f \in F\}\rangle$ und $\mathfrak{B} = \langle B, \{f^{\mathfrak{B}} : f \in F\}\rangle$ Ω–Algebren und $h : A \to B$. h heißt **Homomorphismus**, falls für alle $f \in F$ und alle $\Omega(f)$–Tupel $\vec{x} \in A^{\Omega(f)}$ gilt*

$$h(f^{\mathfrak{A}}(x_0, x_1, \ldots, x_{\Omega(f)-1})) = f^{\mathfrak{B}}(h(x_0), h(x_1), \ldots, h(x_{\Omega(f)-1}))$$

*Wir schreiben $h : \mathfrak{A} \to \mathfrak{B}$, falls h ein Homomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$ ist. Ferner schreiben wir $h : \mathfrak{A} \twoheadrightarrow \mathfrak{B}$, falls h surjektiv ist und $h : \mathfrak{A} \rightarrowtail \mathfrak{B}$, falls h injektiv ist. h ist ein **Isomorphismus**, falls h sowohl injektiv wie auch surjektiv ist. $\mathfrak{B}$ heißt **isomorph** zu $\mathfrak{A}$ falls es einen Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$ gibt; wir schreiben dann $\mathfrak{A} \cong \mathfrak{B}$. Ist $\mathfrak{A} = \mathfrak{B}$, so spricht man von einem **Endomorphismus** von $\mathfrak{A}$; ist h zusätzlich bijektiv, so heißt h ein **Automorphismus** von $\mathfrak{A}$.*

Ist $h : A \to B$ ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$, so ist $h^{-1} : B \to A$ ein Isomorphismus von $\mathfrak{B}$ nach $\mathfrak{A}$.

Definition 1.1.3 *Es sei $\mathfrak{A}$ eine Ω–Algebra und Θ eine zweistellige Relation auf A. Θ heißt eine **Kongruenzrelation** auf $\mathfrak{A}$, falls Θ eine Äquivalenzrelation ist und für alle $f \in F$ und alle $\vec{x}, \vec{y} \in A^{\Omega(f)}$ gilt*

$$\textit{Ist } x_i \,\Theta\, y_i \textit{ für alle } i < \Omega(f), \textit{ so ist } f^{\mathfrak{A}}(\vec{x}) \,\Theta\, f^{\mathfrak{A}}(\vec{y}).$$

Ist Θ eine Äquivalenzrelation, so setze

$$[x]\Theta := \{y : x \,\Theta\, x\}$$

Wir nennen $[x]\Theta$ die **Äquivalenzklasse von** x. Es gilt dann, daß $[x]\Theta = [y]\Theta$ oder aber $[x]\Theta \cap [y]\Theta = \varnothing$. Ferner ist stets $x \in [x]\Theta$. Ist nun Θ auch noch Kongruenzrelation, so gilt: ist $y_i \in [x_i]\Theta$ für alle $i < \Omega(f)$, so ist $f^{\mathfrak{A}}(\vec{y}) \in [f^{\mathfrak{A}}(\vec{x})]\Theta$. Deshalb ist folgende Definition repräsentantenunabhängig.

$$[f^{\mathfrak{A}}]\Theta([x_0]\Theta, [x_1]\Theta, \ldots, [x_{\Omega(f)-1}]\Theta) := [f^{\mathfrak{A}}(x_0, x_1, \ldots, x_{\Omega(f)-1})]\Theta$$

Seien nämlich $y_0 \in [x_0]\Theta$, …, $y_{\Omega(f)-1} \in [x_{\Omega(f)-1}]\Theta$. Dann gilt $y_i \;\Theta\; x_i$ für alle $i < \Omega(f)$. Daraus folgt nach der Kongruenzbedingung sofort $f^{\mathfrak{A}}(\vec{y}) \;\Theta\; f^{\mathfrak{A}}(\vec{x})$. Dies bedeutet nichts anderes als $f^{\mathfrak{A}}(\vec{y}) \in [f^{\mathfrak{A}}(\vec{x})]\Theta$.

Setze $A/\Theta := \{[x]\Theta : x \in A\}$. Wir bezeichnen die Algebra $\langle A/\Theta, \{[f^{\mathfrak{A}}]\Theta : f \in F\}\rangle$ mit $\mathfrak{A}/\Theta$. Wir nennen $\mathfrak{A}/\Theta$ die **Faktorisierung** von $\mathfrak{A}$ nach Θ. Die Abbildung $h_\Theta : x \mapsto [x]\Theta$ erweist man schnell als Homomorphismus.

Sei umgekehrt $h : \mathfrak{A} \to \mathfrak{B}$ ein Homomorphismus. Dann sei $ker(h) := \{\langle x, y\rangle \in A^2 : h(x) = h(y)\}$. $ker(h)$ ist eine Kongruenzrelation auf $\mathfrak{A}$. Ferner ist $\mathfrak{A}/ker(h)$ isomorph zu $\mathfrak{B}$, falls h surjektiv ist. Eine Menge $B \subseteq A$ heißt unter $f \in F$ **abgeschlossen**, falls für alle $\vec{x} \in B^{\Omega(f)}$ gilt $f^{\mathfrak{A}}(\vec{x}) \in B$.

Definition 1.1.4 *Es sei $\langle A, \{f^{\mathfrak{A}} : f \in F\}\rangle$ eine Ω–Algebra und $B \subseteq A$ unter allen $f \in F$ abgeschlossen. Setze $f^{\mathfrak{B}}(\vec{x}) := f^{\mathfrak{A}}(\vec{x})$. Dann ist $f^{\mathfrak{B}} : B^{\Omega(f)} \to B$. Das Paar $\langle B, \{f^{\mathfrak{B}} : f \in F\}\rangle$ heißt dann eine **Unteralgebra** von $\mathfrak{A}$.*

Oft werden wir es mit Strukturen zu tun haben, in denen neben Funktionen auch Relationen definiert sind. Die Definitionen (soweit sie in diesem Fall noch Sinn machen) sind völlig analog, jedoch steigt der notationelle Aufwand erheblich.

Definition 1.1.5 *Es seien F und G Mengen sowie $\Omega : F \to \omega$ und $\Xi : G \to \omega$ Funktionen. Ein Tripel $\mathfrak{A} = \langle A, \Pi, R\rangle$ heißt eine $\langle\Omega, \Xi\rangle$–**Struktur**, falls für $f \in F$ $\Pi(f)$ eine $\Omega(f)$–stellige Funktion auf A und für jedes $g \in G$ $R(g)$ eine $\Xi(g)$–stellige Relation auf A ist. Ω heißt die **funktionale Signatur**, Ξ die **relationale Signatur** von $\mathfrak{A}$.*

Sooft wir können, werden wir auf den Zusatz '$\langle \Omega, \Xi \rangle$' verzichten und nur von Strukturen sprechen. Ist $\langle A, \Pi, R \rangle$ eine $\langle \Omega, \Xi \rangle$–Struktur, so ist $\langle A, \Pi \rangle$ eine Ω–Algebra. Eine Ω–Algebra läßt sich in natürlicher Weise als $\langle \Omega, \varnothing \rangle$–Struktur auffassen, wo $\varnothing$ die leere relationale Signatur ist. Wir gebrauchen eine Konvention ähnlich der für Algebren. Ferner bezeichnen wir die Relationen mit R, S usw. Es seien nun $\mathfrak{A} = \langle A, \{f^{\mathfrak{A}} : f \in F\}, \{R^{\mathfrak{A}} : R \in G\} \rangle$ und $\mathfrak{B} = \langle B, \{f^{\mathfrak{B}} : f \in F\}, \{R^{\mathfrak{B}} : R \in G\} \rangle$ Strukturen gleicher Signatur. Eine Abbildung $h : A \to B$ heißt **Isomorphismus** von $\mathfrak{A}$ nach $\mathfrak{B}$, falls h bijektiv ist und für alle $f \in F$ und alle $\vec{x} \in A^{\Omega(f)}$ gilt

$$h(f^{\mathfrak{A}}(\vec{x})) = f^{\mathfrak{B}}(h(\vec{x}))$$

sowie für alle $R \in G$ und alle $\vec{x} \in A^{\Xi(R)}$

$$R^{\mathfrak{A}}(x_0, x_1, \ldots, x_{\Xi(R)-1}) \quad \text{gdw.} \quad R^{\mathfrak{B}}(h(x_0), h(x_1), \ldots, h(x_{\Xi(R)-1}))$$

1.2 Halbgruppen und Zeichenketten

Sprachen können in erster Näherung als Mengen von Zeichenketten über einem Alphabet aufgefaßt werden. Das Alphabet ist dabei eine endliche, nichtleere Menge A. Das Alphabet hat keinerlei Struktur, es definiert lediglich den Vorrat an primitiven Zeichen. Sinnvollerweise machen wir keine Annahmen über die Anzahl der Elemente von A. Das lateinische Alphabet besteht aus 26 Buchstaben, jeweils in Groß– und Kleinschrift, einigen Satzzeichen und dem Leerzeichen. Das Chinesische 'Alphabet' dagegen besteht aus mehreren Tausend Zeichen!

Zeichenketten sind sehr grundlegende Strukturen. Ohne ein Verständnis von ihnen könnte man zum Beispiel diesen Text nicht lesen. Wir wollen unter einer Zeichenkette über einem Alphabet A zunächst nichts anderes verstehen als das Ergebnis der Verkettung oder Hintereinanderschreibung endlich vieler Symbole aus A. Diese müssen nicht verschieden sein. Ist zum Beispiel $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, so sind `abb`, `bac`, `caaba` jeweils Zeichenketten über A. Zeichenketten bezeichnen wir in der Regel mit Vektorpfeil, etwa $\vec{w}$, $\vec{x}$, $\vec{y}$ etc., um sie von einzelnen Zeichen unterscheiden zu können. Eine formale Definition ist wie folgt.

Definition 1.2.1 *Es sei A eine beliebige Menge. Eine **Zeichenkette über** A ist eine Funktion $\vec{x} : n \to A$ für eine gewisse natürliche Zahl n. n heißt auch die **Länge** von $\vec{x}$ und wird mit $|\vec{x}|$ bezeichnet. $\vec{x}(i)$, $i < n$, heißt das i**te Segment** oder der i**te Buchstabe** von $\vec{x}$. Die eindeutig bestimmte Zeichenkette der Länge 0 wird mit ε bezeichnet. Sind $\vec{x} : m \to A$ und $\vec{y} : n \to A$ Zeichenketten über A, so ist $\vec{x} \cdot \vec{y}$*

diejenige Zeichenkette der Länge $m+n$, für die gilt:

$$(\vec{x}\cdot\vec{y})(j) := \begin{cases} \vec{x}(j), & \textit{falls } j < m, \\ \vec{y}(j-m), & \textit{sonst.} \end{cases}$$

Wir schreiben oft $\vec{x}\vec{y}$ anstelle von $\vec{x}\cdot\vec{y}$. Die Menge A heißt im Zusammenhang dieses Buches auch das ***Alphabet****, ein Element von A heißt auch* ***Buchstabe****. A ist, falls nichts anderes gesagt wird, nicht leer und endlich.*

Eine Zeichenkette schreiben wir durch einfaches Aneinanderreihen der Folgenglieder. So ist also $\texttt{abc}\cdot\texttt{baca} = \texttt{abcbaca}$. Man beachte, daß zwischen den einzelnen Zeichenketten kein Leerzeichen steht, denn das Leerzeichen *ist ein Zeichen*. Wir notieren es mit $\Box$. Zwei Worte unserer Sprache werden jeweils durch ein Leerzeichen (oder auch durch Interpunktionszeichen) voneinander getrennt, und dieses Leerzeichen ist Teil unseres Alphabets! Wir können dies bemerken, wenn wir einen Text mit der Schreibmaschine oder dem Computer schreiben: falls wir ein Leerzeichen wünschen, müssen wir eine Taste drücken. Außerdem sei angemerkt, daß es aus rein formalen Gründen über jedem Alphabet auch die leere Zeichenkette gibt. Das Problem an ihr ist, daß man sie nicht sehen kann, da sie aus keinerlei Zeichen besteht. Sie muß deswegen durch ein spezielles Symbol bezeichnet werden, hier ε. Es gilt also

$$\vec{x}\cdot\varepsilon = \varepsilon\cdot\vec{x} = \vec{x}$$

Man sagt, die leere Zeichenkette sei ein **neutrales Element** oder eine **Eins** bezüglich der Verkettung. Es gilt für Zeichenketten $\vec{x}$, $\vec{y}$ und $\vec{z}$ stets

$$\vec{x}\cdot(\vec{y}\cdot\vec{z}) = (\vec{x}\cdot\vec{y})\cdot\vec{z}$$

Wir sagen daher, die Verknüpfung $\cdot$ sei *assoziativ*. Doch davon gleich mehr. Wir definieren noch die Schreibweise $\vec{x}^i$ induktiv über i.

$$\begin{array}{lcl} \vec{x}^0 & := & \varepsilon\,, \\ \vec{x}^{i+1} & := & \vec{x}\cdot\vec{x}^i\,. \end{array}$$

Ferner wird $\prod_{i<n}\vec{x}_i$ induktiv wie folgt definiert. Ist $n=0$, so sei $\prod_{i<n}\vec{x}_i := \varepsilon$. Ferner ist

$$\prod_{i<n+1} := (\prod_{i<n}\vec{x}_i)\cdot\vec{x}_n\,.$$

Man beachte, daß der Buchstabe $\texttt{a}$ verschieden ist von der Zeichenkette $\vec{x}: 1\to A: 0\mapsto\texttt{a}$. Notiert werden sie jedoch auf die gleiche Weise, nämlich $\texttt{a}$. Wir gebrauchen im Übrigen die Schreibmaschinenschrift, um konkrete Buchstaben und konkrete Zeichenketten hinzuschreiben. Sind $\texttt{a}$ und $\texttt{b}$ Buchstaben unseres Alphabets, so sind $\texttt{ab}$ und $\texttt{abba}$ Zeichenketten; hingegen steht x anstelle eines Buchstabens (ist also eine Metavariable für Buchstaben, wie man sagt), wie auch $\vec{x}$ anstelle von konkreten

Zeichenketten steht. Ist $\vec{x}$ eine Zeichenkette über A und $A \subseteq B$, so ist $\vec{x}$ auch eine Zeichenkette über B. Die Menge aller Zeichenketten über A wird mit A^* bezeichnet. Auf A^* definieren wir die sogenannte **lexikographische Ordnung** wie folgt. Wir ordnen A willkürlich durch $<$. Es ist dann $\vec{x} <_L \vec{y}$, falls es $\vec{u}$, $\vec{v}$ und $\vec{w}$ sowie a und b gibt derart, daß $\vec{x} = \vec{u} \cdot a \cdot \vec{v}$, $\vec{y} = \vec{u} \cdot b \cdot \vec{w}$ und $a < b$. Man überzeuge sich, daß $\vec{x}$ keineswegs kürzer sein muß als $\vec{y}$. Eine andere wichtige Ordnung ist die folgende. Es sei $\mu(a) = k$, falls a das $k$te Symbol von A in der Ordnung $<$ ist. Ferner sei $|A| = n$. Einem Wort $\vec{x} = x_0 x_1 \dots x_{p-1}$ ordnen wir die Zahl $Z(\vec{x})$ zu, wobei

$$Z(\vec{x}) := \sum_{i=0}^{p-1} \mu(x_i) n^{p-i-1}$$

Nun sei $\vec{x} <_N \vec{y}$, falls $Z(\vec{x}) < Z(\vec{y})$. Diese Ordnung wollen wir die **numerische Ordnung** nennen. Wir weisen darauf hin, daß diese Ordnungen von $<$ abhängen. Wir illustrieren die Ordnungen mit einem zweielementigen Alphabet, $A = \{\mathtt{a}, \mathtt{b}\}$. Es sei $\mathtt{a} < \mathtt{b}$. Dann sieht die numerische Ordnung wie folgt aus

$$\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aba}, \mathtt{abb}, \mathtt{baa}, \dots$$

Die lexikographische Ordnung ist etwas komplizierter. Wir illustrieren sie für Worte mit höchstens 4 Buchstaben.

ε,	a,	aa,	aaa,	aaaa,	aaab,
aab,	aaba,	aabb,	ab,	aba,	abaa,
abab,	abb,	abba,	abbb,	b,	ba,
baa,	baaa,	bab,	baba,	babb,	bb,
bba,	bbaa,	bbab,	bbb,	bbba,	bbbb

In der lexikographischen und der numerischen Ordnung ist ε stets das kleinste Element.

Ein **Monoid** ist ein Tripel $\mathfrak{M} = \langle M, 1, \circ \rangle$ dergestalt, daß $\circ$ eine zweistellige Operation auf M ist und 1 ein Element, sodaß für alle $x, y, z \in M$ Folgendes gilt:

$$\begin{array}{lcl} x \circ \varepsilon & = & x \\ 1 \circ x & = & x \\ x \circ (y \circ z) & = & (x \circ y) \circ z \end{array}$$

Zum Beispiel ist die Algebra $\langle \{0,1,2,3\}, 0, max \rangle$ ein Monoid. In der Terminologie des ersten Abschnitts ist ein Monoid eine Algebra bezüglich einer Signatur Ω, welche ein nullstelliges und ein zweistelliges Symbol hat.

Proposition 1.2.2 *Es sei* $\mathfrak{Z}(A) := \langle A^*, \varepsilon, \cdot \rangle$. *Dann ist* $\mathfrak{Z}(A)$ *ein Monoid.*

Ist $\mathfrak{M} = \langle M, 1, \circ\rangle$ ein Monoid und $N \subseteq M$ eine Teilmenge, welche 1 enthält und abgeschlossen ist unter $\circ$, so heißt das Tripel $\langle N, 1, \circ\rangle$ ein **Untermonoid** von $\mathfrak{M}$. Untermonoide von $\mathfrak{M}$ sind eindeutig durch ihre unterliegende Menge bestimmt, da die Operationen ja von $\mathfrak{M}$ herkommen. Seien $\mathfrak{M} = \langle M, 1, \circ\rangle$ und $\mathfrak{N} = \langle N, 1', \circ'\rangle$ Monoide. Eine Abbildung $h : M \to N$ heißt **Homomorphismus**, falls $h(1) = 1'$ und $h(x \circ y) = h(x) \circ' h(y)$ ist für alle $x, y \in M$. In dem Falle, daß h ein Homomorphismus ist, schreiben wir $h : \mathfrak{M} \to \mathfrak{N}$. Die Abbildung, welcher jeder Zeichenkette ihre Länge zuordnet, ist ein Homomorphismus von $\mathfrak{Z}(A)$ in das Monoid $\langle \omega, 0, +\rangle$. Dieser ist surjektiv, da A stets als nicht leer vorausgesetzt wird. Ein Homomorphismus von $\mathfrak{Z}(A)$ nach $\mathfrak{M}$ ist nun bereits eindeutig definiert, wenn er nur auf den Elementen aus A definiert ist. Ferner definiert jede Abbildung $v : A \to M$ einen eindeutig bestimmten Homomorphismus $\overline{v} : \mathfrak{Z}(A) \to \mathfrak{M}$.

Definition 1.2.3 *Sei $\mathfrak{M} = \langle M, 1, \circ\rangle$ ein Monoid und $X \subseteq M$. X **erzeugt** $\mathfrak{M}$, falls das kleinste X enthaltende Untermonoid $\mathfrak{M}$ ist. X heißt dann auch **Erzeugendensystem**. X erzeugt $\mathfrak{M}$ **frei**, falls es zu jedem Monoid $\mathfrak{N} = \langle N, 1, \cdot\rangle$ und zu jeder Abbildung $v : X \to N$ einen eindeutig bestimmten Homomorphismus $\overline{v} : \mathfrak{M} \to \mathfrak{N}$ gibt mit $\overline{v} \restriction X = v$.*

Proposition 1.2.4 *Das Monoid $\mathfrak{Z}(A)$ ist frei erzeugt von A.*

Der Beweis dieser Tatsache ist nicht schwer. Es sei $\mathfrak{N} = \langle N, 1, \circ\rangle$ ein Monoid und $v : A \to N$ eine beliebige Abbildung. Dann definieren wir zunächst eine Abbildung $\overline{v}$ wie folgt:

$$\begin{array}{lcl} \overline{v}(\varepsilon) & := & 1 \\ \overline{v}(a) & := & v(a) \\ \overline{v}(\vec{x} \cdot a) & := & \overline{v}(\vec{x}) \circ v(a) \end{array}$$

Diese Abbildung ist gewiß wohldefiniert, sofern wir in der letzten Zeile noch voraussetzen, daß $\vec{x} \neq \varepsilon$. Wir müssen zeigen, daß diese Abbildung ein Homomorphismus ist. Seien dazu $\vec{x}$ und $\vec{y}$ Worte. Wir wollen zeigen, daß

$$\overline{v}(\vec{x} \cdot \vec{y}) = \overline{v}(\vec{x}) \circ \overline{v}(\vec{y})$$

Wir beweisen diese Behauptung durch Induktion über die Länge von $\vec{y}$. Ist diese Länge 0, so ist die Behauptung gewiß richtig. Denn es ist $\vec{y} = \varepsilon$, und deswegen gilt $\overline{v}(\vec{x} \cdot \vec{y}) = \overline{v}(\vec{x}) = \overline{v}(\vec{x}) \circ 1 = \overline{v}(\vec{x}) \circ \overline{v}(\vec{y})$. Sei nun $|\vec{y}| > 0$. Dann ist $\vec{y} = \vec{w} \cdot a$ für ein $a \in A$.

$$\begin{array}{lcll} \overline{v}(\vec{x} \cdot \vec{y}) & = & \overline{v}(\vec{x} \cdot \vec{w} \cdot a) & \\ & = & \overline{v}(\vec{x} \cdot \vec{w}) \circ v(a) & \text{nach Definition} \\ & = & (\overline{v}(\vec{x}) \circ \overline{v}(\vec{w})) \circ v(a) & \text{nach Induktionsannahme} \\ & = & \overline{v}(\vec{x}) \circ (\overline{v}(\vec{w}) \circ v(a)) & \text{weil } \mathfrak{N} \text{ Monoid} \\ & = & \overline{v}(\vec{x}) \circ \overline{v}(\vec{y}) & \text{nach Definition} \end{array}$$

Dies zeigt die Behauptung. Der Beweis ist beendet.

Die Menge A ist im Übrigen die einzige Menge, die $\mathfrak{Z}(A)$ frei erzeugt. Denn da durch Komposition stets längere Worte entstehen, kann ein Buchstabe nicht aus Worten der Länge > 1 erzeugt werden. Das leere Wort ist stets entbehrlich. Denn wir haben in der Signatur ein nullstelliges Symbol, 1, dessen Wert in einer Algebra stets eine Eins sein muß. Nun kann ein Monoid mehrere Einsen haben, aber das Monoid $\mathfrak{Z}(A)$ hat nur eine. Also muß jede Menge von erzeugenden Elementen das Alphabet enthalten. Dann folgt aber leicht, daß die Menge genau aus den Buchstaben besteht, wenn sie $\mathfrak{Z}(A)$ frei erzeugt. Haben wir etwa $X = \{\mathtt{a}, \mathtt{b}, \mathtt{bba}\}$, so ist $\mathfrak{Z}(A)$ zwar durch X erzeugt, aber nicht frei erzeugt. Zum Beispiel gibt es zu der Abbildung $v : \mathtt{a} \mapsto \mathtt{a}, \mathtt{b} \mapsto \mathtt{b}, \mathtt{bba} \mapsto \mathtt{a}$ keinen Homomorphismus, der diese auf ganz A^* fortsetzt. Denn dann müßte einerseits $\overline{v}(\mathtt{bba}) = \mathtt{a}$ sein, andererseits aber $\overline{v}(\mathtt{bba}) = v(\mathtt{b}) \cdot v(\mathtt{b}) \cdot v(\mathtt{a}) = \mathtt{bba}$.

Die Tatsache, daß $\mathfrak{Z}(A)$ von A frei erzeugt ist, begründet nun zwei Prinzipien. Erstens muß ein Homomorphismus von $\mathfrak{Z}(A)$ in irgendein Monoid lediglich auf A definiert werden, um eindeutig bestimmt zu sein; zweitens aber läßt sich *jede* Abbildung von A in das Zielmonoid zu einem Homomorphismus fortsetzen. Als besondere Anwendung erhalten wir, daß jede Abbildung $v : A \to B^*$ sich zu einem Homomorphismus von $\mathfrak{Z}(A)$ nach $\mathfrak{Z}(B)$ fortsetzen läßt. Ferner haben wir folgendes Ergebnis, welches zeigt, daß die Monoide $\mathfrak{Z}(A)$ bis auf Isomorphie die einzigen frei erzeugten Monoide sind.

Theorem 1.2.5 *Es seien $\mathfrak{M} = \langle M, \circ, 1\rangle$ und $\mathfrak{N} = \langle N, \circ, 1\rangle$ frei erzeugte Monoide. Dann gilt entweder (a) oder (b).*

(a) Es existiert eine Injektion $i : \mathfrak{M} \rightarrowtail \mathfrak{N}$ und eine Surjektion $h : \mathfrak{N} \twoheadrightarrow \mathfrak{M}$ mit $h \circ i = 1_M$.

(b) Es existiert eine Injektion $i : \mathfrak{N} \rightarrowtail \mathfrak{M}$ und eine Surjektion $h : \mathfrak{M} \twoheadrightarrow \mathfrak{N}$ mit $h \circ i = 1_N$.

Beweis. Es sei $\mathfrak{M}$ von X und $\mathfrak{N}$ von Y frei erzeugt. Dann ist entweder $|X| \leq |Y|$ oder $|Y| \leq |X|$. Ohne Beschränkung der Allgemeinheit nehmen wir das Erste an. Dann existiert eine injektive Abbildung $p : X \rightarrowtail Y$ und eine Abbildung $q : Y \twoheadrightarrow X$ derart, daß $q \circ p = 1_X$. Da $\mathfrak{M}$ von X frei erzeugt wird, existiert ein Homomorphismus $\overline{p} : \mathfrak{M} \to \mathfrak{N}$ mit $\overline{p} \restriction X = p$. Ebenso existiert ein Homomorphismus $\overline{q} : \mathfrak{N} \to \mathfrak{M}$ mit $\overline{q} \restriction Y = q$, da $\mathfrak{N}$ frei erzeugt wird von Y. Nun ist die Einschränkung $\overline{q} \circ \overline{p}$ auf X die Identität. (Denn $\overline{q} \circ \overline{p}(x) = \overline{q}(p(x)) = q(p(x)) = x$.) Da $\mathfrak{M}$ von X frei erzeugt wird, existiert genau ein Homomorphismus, welcher 1_X auf ganz $\mathfrak{M}$ fortsetzt, und dies ist die Identität. Deswegen ist $\overline{q} \circ \overline{p} = 1_M$. Es folgt daraus, daß $\overline{q}$ surjektiv ist und $\overline{p}$ injektiv. Also ist Fall (a) eingetreten. Ist $|Y| \leq |X|$, so ist analog (b) der Fall. $\dashv$

Theorem 1.2.6 *In $\mathfrak{Z}(A)$ gelten auch folgende Kürzungsregeln.*

1. *Falls $\vec{x} \cdot \vec{u} = \vec{y} \cdot \vec{u}$, so $\vec{x} = \vec{y}$.*
2. *Falls $\vec{u} \cdot \vec{x} = \vec{u} \cdot \vec{y}$, so $\vec{x} = \vec{y}$.*

Ist $\vec{x}$ eine Zeichenkette, so ist $\vec{x}^T$ definiert durch

$$\begin{array}{lcl} \varepsilon^T & := & \varepsilon \\ a^T & := & a \\ (\vec{x} \cdot \vec{y})^T & := & \vec{y}^T \cdot \vec{x}^T \end{array}$$

$\vec{x}^T$ heiße das **Spiegelwort** von $\vec{x}$. Es ist leicht zu zeigen, daß $(\vec{x}^T)^T = \vec{x}$. Der Leser überzeuge sich davon, daß die Abbildung $\vec{x} \mapsto \vec{x}^T$ *kein* Homomorphismus ist.

Definition 1.2.7 *Es seien $\vec{x}, \vec{y} \in A^*$. Dann heißt $\vec{x}$ ein **Präfix** von $\vec{y}$, falls $\vec{y} = \vec{x} \cdot \vec{u}$ für ein $\vec{u} \in A^*$. $\vec{x}$ heißt **Postfix** oder **Suffix** von $\vec{y}$, falls $\vec{y} = \vec{u} \cdot \vec{x}$ für ein gewisses $\vec{u} \in A^*$. $\vec{x}$ heißt **Teilwort** von $\vec{y}$, falls $\vec{y} = \vec{u} \cdot \vec{x} \cdot \vec{v}$ für gewisse $\vec{u}, \vec{v} \in A^*$.*

Es ist leicht einzusehen, daß $\vec{x}$ genau dann Präfix ist von $\vec{y}$ wenn $\vec{x}^T$ Postfix ist von $\vec{y}^T$. Man beachte, daß ein gegebenes Wort $\vec{x}$ mehrmals als Teilwort in $\vec{y}$ auftreten kann. So tritt z. B. `aa` in `aaaaa` insgesamt viermal auf. Die Vorkommen von `aa` überschneiden sich auch. Ist allerdings $\vec{y}$ und $\vec{x}$ gegeben, so existiert höchstens ein Vorkommen von $\vec{x}$ als Präfix bzw. als Postfix von $\vec{y}$. Man unterscheide daher sorgfältig zwischen der Tatsache, daß ein Wort $\vec{x}$ Teilwort eines Wortes $\vec{y}$ ist und einem bestimmten Vorkommen von $\vec{x}$ in $\vec{y}$. Vorkommen von Teilworten kann man auf zwei Weisen eindeutig bestimmen. Einer Kette $x_0 x_1 \ldots x_{n-1}$ ordnen wir $n+1$ sogenannte **Positionen** zu. Diese Positionen können wir als Zeitpunkte interpretieren; jedes x_i definiert nämlich ein Ereignis — die Äußerung von x_i in der gesprochenen Sprache —, das zu einem Zeitpunkt t_i beginnt und zu einem späteren Zeitpunkt t_{i+1} aufhört. Der Zeitpunkt t_i bestimmt die Position i. Die x_i sind also stets zwischen die Positionen eingeschoben. Jedes Teilwort $x_i x_{i+1} \ldots x_{j-1}$, $i < j$, wird eindeutig bestimmt durch das Paar $\langle i, j \rangle$. Das leere Wort nimmt einen Sonderstatus ein. Zwar könnte man es als Paar $\langle i, i \rangle$ kodieren, aber dann hätte man je nach Wahl von i verschiedene leere Worte. Alternativ dazu kann man ein Teilwort durch die beiden links und rechts neben ihm verbleibenden Worte charakterisieren. Dazu definieren wir einen **Kontext** als ein Paar $C = \langle \vec{y}, \vec{z} \rangle$ von Zeichenketten. Eine **Einsetzung** von $\vec{x}$ in C, in Zeichen $C(\vec{x})$, ist definiert als die Zeichkette $\vec{y} \cdot \vec{x} \cdot \vec{z}$. Wir sagen, $\vec{x}$ **kommt in $\vec{v}$ im Kontext C vor**, falls $\vec{v} = C(\vec{x})$. Stets ist $\vec{x}$ ist ein Teilwort von $C(\vec{x})$. Jedes Vorkommen von $\vec{x}$ in einem Wort $\vec{v}$ ist eindeutig durch seinen Kontext bestimmt. Deswegen sagen wir, C sei ein **Teilwortvorkommen** von $\vec{x}$ in $\vec{v}$.

Definition 1.2.8 *Es seien $C = \langle \vec{u}_1, \vec{u}_2 \rangle$ und $D = \langle \vec{v}_1, \vec{v}_2 \rangle$ Vorkommen einer Zeichenkette $\vec{x}$ beziehungsweise $\vec{y}$ in $\vec{z}$. Dann sagen wir, C* ***sei in*** *D* ***enthalten****, falls $\vec{v}_1$ Präfix von $\vec{u}_1$ und $\vec{v}_2$ Suffix von $\vec{u}_2$ ist. Ferner sagen wir, C und D* ***überlappen****, falls (1) $\vec{u}_1\vec{x}$ kein Präfix von $\vec{v}_1$ und $\vec{v}_1\vec{y}$ kein Präfix von $\vec{u}_1$ und (2) $\vec{x}\vec{u}_2$ kein Suffix von $\vec{v}_2$, und $\vec{x}\vec{u}_2$ kein Suffix von $\vec{v}_2$ und $\vec{y}\vec{v}_2$ kein Suffix von $\vec{u}_2$ ist.*

Man beachte, daß $\vec{x}$ ein Teilwort von $\vec{y}$ sein kann, aber nicht jedes Vorkommen von $\vec{x}$ in jedem Vorkommen von $\vec{y}$ enthalten sein muß. Zum Beispiel ist $\mathtt{a}$ ein Teilwort von $\mathtt{ab}$ aber das Vorkommen $\langle \varepsilon, \mathtt{ab} \rangle$ in der Zeichenkette $\mathtt{aab}$ ist nicht in dem Vorkommen $\langle \mathtt{a}, \varepsilon \rangle$ sondern nur in dem Vorkommen $\langle \varepsilon, \mathtt{b} \rangle$ enthalten.

Definition 1.2.9 *Eine* ***Sprache*** *über einem Alphabet A ist eine beliebige Menge $S \subseteq A^*$.*

Diese Definition läßt zu, daß $S = \varnothing$ ist wie auch, daß $S = A^*$. Es gibt damit unabhängig von der Kardinalität von A $2^{\aleph_0}$ viele Sprachen. Denn A^* enthält abzählbar viele Elemente, und zwar genau $\aleph_0$ viele, da A nicht leer ist und endlich. Im Falle, wo A mindestens 2 Buchstaben enthält, läßt sich dieser Sachverhalt auch leicht direkt beweisen.

Theorem 1.2.10 *Dazu sei $C = \{c_i : i < p\}$, $p > 2$, ein beliebiges Alphabet und $A = \{\mathtt{a}, \mathtt{b}\}$. Ferner sei $\overline{v}$ die homomorphe Fortsetzung von $v : c_i \mapsto \mathtt{a}^i \cdot \mathtt{b}$. Die Abbildung $S \mapsto \overline{v}[S] : \wp(C^*) \to \wp(A^*)$ definiert durch $V(S) = \overline{v}[S]$ ist eine Bijektion zwischen $\wp(C^*)$ und den Sprachen, welche im vollen Bild von $\overline{v}$ enthalten sind.*

Der Beweis ist eine Übung. Die Menge aller Sprachen über A ist abgeschlossen bezüglich der Operationen $\cap$, $\cup$, und $-$, dem relativen Komplement bezüglich A^*. Ferner können wir die folgende Operationen definieren.

$$\begin{array}{lcl} S \cdot T & := & \{\vec{x} \cdot \vec{y} : \vec{x} \in S, \vec{y} \in T\} \\ S^0 & := & \{\varepsilon\} \\ S^{n+1} & := & S^n \cdot S \\ S^* & := & \bigcup_{n \in \omega} S^n \\ S/T & := & \{\vec{y} \in A^* : (\exists \vec{x} \in T)(\vec{y} \cdot \vec{x} \in S)\} \\ T\backslash S & := & \{\vec{y} \in A^* : (\exists \vec{x} \in T)(\vec{x} \cdot \vec{y} \in S)\} \end{array}$$

* heißt der **Kleenesche Stern**. Zum Beispiel ist S/A^* die Menge aller Zeichenketten, welche man zu einer Zeichenkette aus S verlängern kann, d. h. die Menge aller Präfixe von Ketten aus S. Wir bezeichnen diese Menge auch als die **Präfixhülle** von S, in Zeichen S^P. Analog ist $S^S := A^*\backslash S$ die **Suffix**– oder **Postfixhülle** von S. Daraus folgt dann, daß $(S^P)^S$ nichts weiter ist als die Teilworthülle von S.

Sei S eine Sprache über A, $C = \langle \vec{x}, \vec{y} \rangle$ ein Kontext und $\vec{u}$ eine Zeichenkette. Wir sagen, C **akzeptiert** $\vec{u}$ **in** S, falls $C(\vec{u}) \in S$. Es sei nun $M \subseteq A^*$ und $P \subseteq A^* \times A^*$. Dann sei $C_S(M)$ die Menge aller C, die alle Zeichenketten aus M akzeptieren; und es sei $Z_S(P)$ die Menge aller Zeichenketten, die von allen Kontexten aus P akzeptiert werden. Dann gilt:

Lemma 1.2.11 *Es sei S eine Sprache über A. Ferner seien $M, N \subseteq A^*$ und $P, Q \subseteq A^* \times A^*$. Dann gilt:*

1. *Genau dann ist $M \subseteq Z_S(P)$, wenn $C_S(M) \supseteq P$.*
2. *Ist $M \subseteq N$, so ist $C_S(M) \supseteq C_S(N)$.*
3. *Ist $P \subseteq Q$, so ist $Z_S(P) \supseteq Z_S(Q)$.*
4. *$M \subseteq Z_S(C_S(M))$.*
5. *$P \subseteq C_S(S_Z(P))$.*

Beweis. Es sei $M \subseteq Z_S(P)$. Dann wird jede Zeichenkette aus M von allen Kontexten aus P akzeptiert. Also ist jeder Kontext in P ein Kontext, der alle Zeichenketten aus M akzeptiert. Es gilt daher $P \subseteq C_S(M)$. Die Umkehrung beweist man genauso. Ist nun $M \subseteq N$ und ist C ein Kontext, der alle Zeichenketten aus N akzeptiert, so akzeptiert C auch alle Zeichenketten aus M. Dies zeigt die zweite Behauptung; die dritte wird analog gezeigt. Nun zur vierten. Mit der ersten Behauptung folgt aus $C_S(M) \supseteq C_S(M)$ bereits $M \subseteq Z_S(C_S(M))$. Ebenso folgt aus $Z_S(P) \subseteq Z_S(P)$ bereits $P \subseteq C_S(Z_S(P))$. $\dashv$

Wir nennen M **abgeschlossen**, falls $M = Z_S(C_S(M))$. Die abgeschlossenen Klassen bilden genau die sogenannten Distributionsklassen von Zeichenketten. Man nennt $Z_S(C_S(M))$ die **Sestier–Hülle** von M und die Abbildung $S_S : M \mapsto Z_S(C_S(M))$ den **Sestier–Operator**.

Definition 1.2.12 *Es sei M eine Menge und $H : \wp(M) \to \wp(M)$ eine Funktion. H heißt **Hüllenoperator** auf M, falls für alle $X, Y \subseteq M$ gilt:*

1. *$X \subseteq H(X)$.*
2. *Aus $X \subseteq Y$ folgt $H(X) \subseteq H(Y)$.*
3. *$H(X) = H(H(X))$.*

Proposition 1.2.13 *Der Sestier–Operator ist ein Hüllenoperator.*

Beweis. Die erste Behauptung ergibt sich aus Lemma 1.2.11. Ferner folgt aus $M \subseteq N$ erst $C_S(M) \supseteq C_S(N)$ und daraus wiederum $Z_S(C_S(M)) \subseteq Z_S(C_S(N))$. Nun haben wir wegen $M \subseteq Z_S(C_S(M))$ einerseits $C_S(M) \supseteq C_S(Z_S(C_S(M)))$, wegen der ersten Behauptung des vorigen Lemmas andererseits $C_S(M) \subseteq C_S(Z_S(C_S(M)))$, da $P \mapsto C_S(Z_S(P))$ monoton ist. Also gilt $C_S(M) = C_S(Z_S(C_S(M)))$, und daraus folgt $Z_S(C_S(M)) = Z_S(C_S(Z_S(C_S(M))))$. Dies ist aber die Behauptung, daß $S_S(M) = S_S(S_S(M))$. $\dashv$

Aus gewissen Gründen ist einem Mathematiker das Hantieren mit Zeichenketten suspekt, da es auf einer gewissem Anschauung beruht (dem Symbol bzw. der Kette von Symbolen auf dem Papier). Daher haben wir eine Zeichenkette als eine Funktion von einem Abschnitt der naürlichen Zahlen in das Alphabet definiert. Allerdings muß man sich klar machen, daß wir Terme durch Zeichenketten bezeichnet haben. Diese sind jedoch streng genommen nicht die Terme, sondern lediglich Repräsentanten. Dazu eine Definition.

Definition 1.2.14 *Es sei Ω eine Signatur. Eine **Repräsentation von Termen (mittels Zeichenketten über** A**)** ist eine Relation $R \subseteq Tm_\Omega \times A^*$ derart, daß zu jedem Term t eine Zeichenkette $\vec{x}$ existiert mit $\langle t, \vec{x}\rangle \in R$. $\vec{x}$ heißt **Repräsentant** oder **repräsentierende Zeichenkette** von t **in Bezug auf** R. $\vec{x}$ heißt **eindeutig lesbar**, falls aus $\langle t, \vec{x}\rangle, \langle u, \vec{x}\rangle \in R$ folgt, daß $t = u$. R heißt **eindeutig** beziehungsweise **eindeutig lesbar**, falls jedes $\vec{x} \in A^*$ eindeutig lesbar ist.*

Dem Leser wird als Übung überlassen zu zeigen, daß die im vorigen Abschnitt definierte Repräsentation tatsächlich eindeutig ist. Dies ist nicht selbstverständlich. Ein Term könnte unter Umständen mehrere repräsentierende Zeichenketten besitzen. Unsere übliche Art, arithmetische Terme zu notieren ist nicht notwendig eindeutig. Wir schreiben zum Beispiel $2+3+4$, obwohl dies sowohl für den Term $+(+(2,3),4)$ wie auch für den Term $+(2,+(3,4))$ stehen kann. Diese haben zwar denselben Wert (nämlich 9), sind aber als Terme offensichtlich verschieden. Diese Konvention ist also nützlich, führt aber zu nicht eindeutigen Termen.

Es gibt noch andere Konventionen, welche wir benutzen. Wir nennen einige Beispiele. (a) Ein zweistelliges Symbol wird stets zwischen seine Argumente anstelle des Komma gesetzt (das ist die sogenannte *Infixnotation*. Also schreiben wir nicht `+(2,3)` sondern `(2+3)`. (b) Äußere Klammern dürfen fortgelassen werden. `(2+3)` bezeichnet denselben Term wie `2+3`. (c) Das Multiplikationszeichen bindet stärker als das Additionszeichen. Es bezeichnen also folgende Zeichenketten stets denselben Term:

`(2+(3·5))` `2+(3·5)` `(2+3·5)` `2+3·5`

In der Logik benutzt man auch Punkte anstelle von Klammern. Die Schreibweise

$p \wedge q. \to .p$ ist kurz für $(p \wedge q) \to p$. Die Punkte werden also rechts und links von dem höheren Zeichen gesetzt.

Da allerdings die Zeichenkette `(2+3)·5` einen anderen Term repräsentiert als `2+3·5` (und beide auch ein anderes Ergebnis liefern, nämlich 30 beziehungsweise 17), so ist das Setzen von Klammern durchaus notwendig. Daß es auch ohne Klammern geht, diese Erkenntnis verdanken wir dem polnischen Logiker Jan Łukasiewicz. In seiner Notation, die deswegen auch **Polnische Notation** heißt, wird das Funktionssymbol stets vor seine Argumente gesetzt. Alternativ kann man es auch ans Ende setzen (dies ist die sogenannte **umgekehrt**(**e**) **polnische Notation**). Gewisse Rechner haben die umgekehrt polnische Notation implementiert. Man sucht deswegen vergebens nach einem Klammersymbol. Dafür bekommt man es aber mit der Taste namens '`enter`' zu tun. Diese braucht man nämlich, um zwei aufeinanderfolgende Zahlen zu trennen. Denn jetzt würde der Term $+(13, 5)$ wie folgt notiert: `135+`. Das Problem ist, daß auch der Term $+(1, 35)$ auf diese Weise geschrieben wird. Damit dies nicht geschieht, muß man die erste Zahl von der zweiten durch `enter` trennen. Die Eingabe ist somit `13` `enter` `5+`. (Hierbei macht die Box in der üblichen Schreibweise der Computerhandbücher aus der Zeichenkette `enter` einen Buchstaben, der der Taste gleichen Namens entspricht. Im Kapitel 3 werden wir im Übrigen noch ausführlich auf das Problem der Notation von Zahlen zurückkommen.)

Nun aber zu dem versprochenen Satz über die eindeutige Lesbarkeit. Wir zeigen, daß die Polnische Notation (PN) eindeutig lesbar ist. Es sei dazu F eine Menge von Symbolen und Ω eine Signatur über F, wie im vorigen Abschnitt definiert. Jedem Symbol f wird eine Zahl $\Omega(f)$, die Stelligkeit, zugeordnet. Nun wird eine Menge $Tm_\Omega(X)$ von Zeichenketten über der Symbolmenge und Variablen aus X definiert, welche wir der Kürze halber *wohlgeformt* nennen wollen. $Tm_\Omega(X)$ ist die kleinste Menge M von Zeichenketten, für die gilt:

1. Für jedes $x \in X$ ist $x \in M$.

2. Ist $f \in F$ und $\vec{x}_i \in M$, $i < \Omega(f)$, dann ist $f \cdot \vec{x}_0 \cdot \ldots \cdot \vec{x}_{\Omega(f)} \in M$.

Dies ist also die sogenannte Polnische Notation. Die Zeichenkette 'x' repräsentiert den Term 'x'. Falls $\vec{x}_i$ den Term t_i repräsentiert für jedes $i < \Omega(f)$, so repräsentiert die Zeichenkette $f \cdot \vec{x}_0 \cdot \ldots \cdot \vec{x}_{\Omega(f)-1}$ den Term $f(t_0, \ldots, t_{\Omega(f)-1})$. Wir wollen nun zeigen, daß die Polnische Notation eindeutig lesbar ist. (Ein anderer Beweis wie der hier vorgeschlagene findet sich in Abschnitt 2.4, Beweis zu Satz 2.4.4.) Dazu bedienen wir uns eines wichtigen Schlußverfahrens, der Induktion über die Erzeugung einer Zeichenkette. Wir beweisen nun induktiv: (1) kein echtes Anfangsstück einer wohlgeformten Zeichenkette ist eine wohlgeformte Zeichenkette, (2) ist $\vec{x}$ eine wohlgeformte Zeichenkette, so hat $\vec{x}$ mindestens die Länge 1, und es gilt:

1. Hat $\vec{x}$ die Länge 1, so ist $\vec{x} = a$ für ein $a \in X$, oder $a \in F$ und $\Omega(a) = 0$.
2. Hat $\vec{x}$ die Länge mindestens 2, so existieren f und $\vec{y}$ mit $\vec{x} = f \cdot \vec{y}$, wobei $\vec{y}$ eine Folge von $\Omega(f)$ vielen wohlgeformten Zeichenketten ist.

Der Beweis ist wie folgt. Seien t und u Terme, welche durch die Zeichenkette $\vec{x}$ repräsentiert werden. Habe $\vec{x}$ die Länge 1. Dann sind t und u Terme der Form a, $a \in X$ oder $a \in F$ mit $\Omega(a) = 0$. Es ist klar, daß dann $t = u$ ist. Ein echtes Anfangsstück von a ist die leere Zeichenkette, ε, welche nicht wohlgeformt ist. Dies sichert den Induktionsanfang. Nun sei $\vec{x}$ von der Länge mindestens 2. Dann existiert ein $f \in F$ und eine Folge $\vec{y}_i$, $i < \Omega(f)$, von wohlgeformten Ketten derart, daß

$$(\ddagger) \qquad \vec{x} = f \cdot \vec{y}_0 \cdot \ldots \cdot \vec{y}_{\Omega(f)-1}$$

Es existieren also Terme b_i, $i < \Omega(f)$, welche durch $\vec{y}_i$ repräsentiert werden. Diese Terme sind eindeutig, nach Induktionsannahme. Ferner ist das Symbol f eindeutig bestimmt. Seien nun $\vec{z}_i$, $i < \Omega(f)$, wohlgeformte Zeichenketten mit

$$\vec{x} = f \cdot \vec{z}_0 \cdot \ldots \cdot \vec{z}_{\Omega(f)-1}$$

Dann gilt $\vec{y}_0 = \vec{z}_0$. Denn es ist $\vec{y}_0$ Anfangsstück von $\vec{z}_0$ oder $\vec{z}_0$ Anfangsstück von $\vec{y}_0$. Aber kein echtes Anfangsstück von $\vec{y}_0$ oder $\vec{z}_0$ ist eine wohlgeformte Kette. Also sind beide Ketten gleich. Ist $\Omega(f) = 1$, so sind wir fertig. Andernfalls folgern wir nach demselben Muster, daß $\vec{y}_1 = \vec{z}_1$, $\vec{y}_2 = \vec{z}_2$, usw. Die Zerlegung in $\Omega(f)$ viele Ketten ist demnach eindeutig, und so repräsentiert die Kette genau einen Term.

Zu guter Letzt müssen wir noch sichern, daß kein echtes Anfangsstück von $\vec{x}$ eine wohlgeformte Kette ist. Dazu nehmen wir uns wieder die Zerlegung ($\ddagger$) vor. Ist $\vec{u}$ ein wohlgeformtes Anfangsstück, so ist $\vec{u} \neq \varepsilon$. Also $\vec{u} = f \cdot \vec{v}$ für ein $\vec{v}$, welches sich in $\Omega(f)$ viele wohlgeformte Ketten $\vec{w}_i$ zerlegen läßt. Wie vorher argumentieren wir nun, daß $\vec{w}_i = \vec{x}_i$ sein muß für alle $i < \Omega(f)$. Also $\vec{u} = \vec{x}$, und dies zeigt, daß kein echtes Anfangsstück wohlgeformt ist.

Übung 1. Beweisen Sie den Satz 1.2.10.

Übung 2. Zeigen Sie, daß die Postfixrelation eine partielle Ordnung ist, ebenso die Präfix– und Teilwortrelation. Zeigen Sie ferner, daß die Teilwortrelation die transitive Hülle der Vereinigung von der Postfixrelation mit der Präfixrelation ist.

Übung 3. Es seien F, X und $\{(,)\}$ drei paarweise disjunkte Mengen, Ω eine Signatur auf F. Wir definieren folgende Funktion von Ω–Termen in Zeichenketten über $F \cup X \cup \{(,)\}$:

$$\begin{array}{lcl} x^+ & := & x \\ f(t_1, \ldots, t_{\Omega(f)-1})^+ & := & f \cdot (\cdot t_1^+ \cdot \ldots \cdot t_n^+ \cdot) \end{array}$$

(Also: wir repräsentieren Terme durch die Zeichenkette, die wir in Abschnitt 1.1 hingeschrieben haben.) Man beweise die eindeutige Lesbarkeit. Man beachte, daß

dies nicht schon daraus folgt, daß wir ja diese Zeichenketten als repräsentierende Zeichenketten gewählt haben. Wir müssen die eindeutige Lesbarkeit wirklich zeigen!

Übung 4. Man gebe eine exakte obere Schranke an für die Anzahl der Präfixe (Postfixe) eines Wortes der Länge n, n eine natürliche Zahl, sowie eine Schranke für die Anzahl von (Vorkommen von) Teilworten.

Übung 5. Seien $S, T \subseteq A^*$. Definiere

$$\begin{array}{rcl} S//T & := & \{\vec{y} : (\forall \vec{x} \in T)(\vec{y} \cdot \vec{x} \in S)\} \\ T\backslash\backslash S & := & \{\vec{y} : (\forall \vec{x} \in T)(\vec{x} \cdot \vec{y} \in S)\} \end{array}$$

Man zeige: Für alle $S, T, U \subseteq A^*$ gilt

$$T \subseteq S\backslash\backslash U \qquad \Leftrightarrow \qquad S \cdot T \subseteq U \qquad \Leftrightarrow \qquad S \subseteq U//T$$

Übung 6. Man zeige, daß nicht alle Äquivalenzen der vorigen Übung gelten, wenn man statt \\ und // einfach \ und / nimmt. Welche Richtungen bleiben allerdings dennoch gültig?

1.3 Grundlegendes aus der Sprachwissenschaft

In diesem Abschnitt wollen wir einiges Grundsätzliche zu unserer Konzeption von Sprache sagen. Da wir die sprachwissenschaftlichen Termini nicht wirklich erklären werden, sei der Leser an dieser Stelle auf das Buch von Bussmann [5] verwiesen. Strukturell gesehen ist Sprache in vier Ebenen oder Schichten (sogenannten **Strata**) organisiert: dem phonologischen, dem morphologischen, dem syntaktischen und dem semantischen Stratum. Jedes Stratum besitzt elementare Einheiten und Kombinationsregeln; Strata sind untereinander durch Realisierungsregeln verbunden. Auf der phonologischen Ebene finden wir die bloße Repräsentation einer Äußerung in ihrer lautlichen Form. Elementare Einheiten sind *Phone*. Begriffe wie Phon, Silbe, Betonung beziehen sich auf diese Ebene. Auf der morphologischen Ebene finden wir die Elementarzeichen der Sprache, welche auch *Morphe* genannt werden. Diese sind die kleinsten sinntragenden Einheiten und durchaus von Worten verschieden. So ist das Wort `sehen` das Ergebnis der Kombination von zwei Morphen, nämlich der Wurzel `seh` und dem Infinitivmorph `en`. Das Morph `Baum` ist allerdings ein selbständiges Wort. Auf dem syntaktischen Stratum sind die Einheiten **Worte** (auch **Lexe** genannt), und auf der semantischen Ebene sind es die **Seme**. Gewisse Phone sind, wie man sich ausdrückt, nichts als alternative Realisierungen einer abstraktem Einheit, des *Phonems*. So wird `ch` in `Licht` anders ausgesprochen als `Nacht`; dies ist nach allgemeiner Auffasssung alleine durch den davorstehenden Vokal bestimmt. Man

nimmt also an, das Deutsche besitze ein Phonem `x`, welches je nach vorangehendem Vokal wie in `Licht`, oder wie in `Nacht` ausgesprochen wird. Desgleichen nimmt man für das Deutsche an, es gebe ein Pluralmorphem, obwohl es sicher zahlreiche Pluralmorphe gibt. Die folgende Liste gibt eine Auswahl möglicher Pluralmorphe.

Singular	Plural
Wagen	Wagen
Auto	Autos
Bus	Busse
Licht	Lichter
Vater	Väter
Nacht	Nächte

Der Plural wird also wahlweise durch die leere Zeichenkette, durch ein `s`–Suffix, ein `e`–Suffix (die Verdopplung ist ein phonologischer Effekt), ein `er`–Suffix, oder gar allein durch Umlaut oder Kombination von Umlaut mit `e`–Suffix bezeichnet. All dies sind verschiedene Morphe, aber sie gehören einem einzigen Morphem an; man sagt deswegen, sie seien **Allomorphe**. Die Schichtung in Ebenen erlaubt es, schrittweise Abstraktionen vorzunehmen und sich so von störenden Phänomenen zu befreien. Auf der sogenannten oberflächenmorphologischen Ebene ist `Nächte` die Kombination von zwei Morphen, dem Morph `Nacht` und dem Pluralmorph, welches den Umlaut bildet und ein `e`–Suffix anhängt. (Wie der Umlaut gebildet wird, das muß im Übrigen die phonologische Ebene bestimmen; zum Beispiel heißt es `Altäre` und nicht etwa `Ältare` oder `Ältäre`!) auf der sogenannten tiefenmorphologischen Ebene findet sich davon nur noch die Kombination von zwei Morphemen, dem Morphem `Nacht` und dem Pluralmorphem. Auf der syntaktischen Ebene ist davon wiederum nicht zu sehen. Dort haben wir ein einziges Lex, nämlich `Nächte`. Auf der phonologischen Ebene haben wir dagegen eine Folge von 5 (!) Phonen, welche in der Schreibung dem `n`, dem `ä`, dem `ch`, dem `t` und dem `e` entsprechen. Auch hier unterscheiden wir Oberflächenanalyse von Tiefenanalyse. Auf der tiefenphonologischen Ebene der Unterschied zwischen den Allophonen von `x` wiederum veschwunden.

In Abschnitt 3.1 werden wir einen Zeichentheoretischen Ansatz vorstellen. Dieser unterscheidet lediglich 3 Ebenen: diese entsprechen für ein Zeichen seiner *Realisierung*, seiner *Kombinationsfähigkeit* und seiner *Bedeutung*. Die Realisierung können wir wahlweise als Phon(em)sequenz, oder als Morph(em)sequenz oder gar als Lex(em) ansetzen. Jeder dieser Ansätze ist legitim und führt zu ganz neuen Einsichten.

In der Sprachwissenschaft unterscheidet man zwischen einem Buchstaben und Lauten. Diese betrifft — wie man sagt — den **Kanal**. Der Kanal ist das Medium, in welchem sich die Botschaft (man spricht auch von **Nachricht**) physikalisch manifestiert. Sprache manifestiert sich ihrem Wesen nach akustisch (dadurch, daß sie

gesprochen wird), in den meisten Fällen allerdings auch graphisch. [1] Jeder Kanal erlaubt — durch seine physikalische Beschaffenheit — eine ganz andere Kombinationsweise. Ein Blatt Papier ist ein zweidimensionales Gebilde, und wir sind, anders als in der gesprochenen Sprache, gar nicht gezwungen, Sprache linear fortlaufend zu notieren, auch wenn wir dies in der Regel so tun. Man denke aber an die Tatsache, daß zum Beispiel ein Chinesisches Zeichen ein graphisches Gebilde ist, welches die Zweidimensionalität wesentlich benutzt, indem Zeichen aus Teilzeichen zusammengesetzt werden. Dies nimmt ganz im Gegensatz zu westeuropäischen Sprachen keinen Bezug dazu, daß die repräsentierte Silbe eine Folge von Lauten ist. Das Devanagari, worin zum Beispiel Hindi geschrieben wird, ist eine Silbenschrift. Ein Zeichen bezeichnet zunächt einen Konsonant plus den Vokal `a`. Zusatzzeichen bestimmen, ob der nachfolgende Vokal verschieden von `a` ist. Eine Konsonantenhüfung wir durch Verklebung der Zeichen für die Einzelkonsonanten geschrieben. Man kann auch in der Mathematik sehen, was der schriftliche Kanal an Mitteln zuläßt: wir können Indizes, Superskripte, Subskripte, Unterstreichungen usw. benutzen oder sogar Funktionen einfach in einem Achsenkreuz darstellen.

Während die akustische Manifestation von Sprache in gewisser Weise essentiell für menschliche Sprache ist, muß man ihre schriftliche Manifestation erst erfinden. Während also die Lautstruktur von Sprachen natürlich gewachsen ist, ist ein Schriftssystem ein Kulturprodukt. Deswegen kommt es vor, daß Schriftsysteme gar nicht so einheitlich sind. Manche Schriftsysteme fassen ganze Worte als Einheiten (Chinesisch), andere nur Silben (Devanagari, worin zum Beispiel Hindi geschrieben wird), oder andere wiederum nur Laute. [2] Wir wollen im Folgenden immer Sprache als geschriebene Sprache studieren, wollen aber trotzdem einiges Grundsätzliches über die Beziehung zwischen Laut- und Schriftbild sagen. Wir benutzen das sogenannte *Lateinische Alphabet*, welches in den meisten europäischen Ländern üblich ist, wobei allerdings jedes Land dennoch einen mehr oder weniger spezifischen Satz an Buchstaben hat. Der Buchstabe ß ist zum Beispiel spezifisch für das Deutsche (wird aber in der Schweiz nicht verwendet). Das Dänische kennt das œ, das Polnische das ␣Ł, das Ungarische das ȟ usf. Der Vorrat an Einzelzeichen, welche wir in der Terminologie als **Buchstaben** bezeichnen, liegt irgendwo zwischen 60 und 100. Dazu gehören neben je einem kleinen und großen Buchstaben und den Satzzeichen sowie dem Leerzeichen, noch die eben genannten Sonderzeichen.

[1] Tausbtummensprachen sind hier ein Fall für sich, auf den wir nicht näher eingehen wollen.

[2] Man mag einwenden, daß im Chinesischen ein Zeichen immer nur eine Silbe bezeichnet, aber Worte aus mehreren Silben, also mehreren Zeichen bestehen. Trotzdem besteht ein Unterschied zum Devanagari. Viele Zeichen haben dieselbe Aussprache, aber unterschiedliche Bedeutung. Sie werden im Chinesischen also durch unterschiedliche Zeichen, nicht aber im Devanagari. Auch hier ist Vorsicht geboten. Das Französische nähert sich in dieser Hinsicht nämlich (trotz lateinischen Alphabets) dem Chinesischen an. Die folgenden Worte werden zum Beispiel völlig gleich ausgesprochen: `au`, `haut`, `eau`, `eaux`; desgleichen `vers`, `vert`, `verre`, `verres`.

Das Gegenstück zu dem Buchstaben ist in der gesprochenen Sprache das sogenannte **Phonem**. Jede sprachliche Äußerung ist in eine Folge von Phonemen analysierbar (sowie einem Rest, auf den wir noch weiter unten eingehen werden). Es gibt aber leider keine eineindeutige Zuordnung zwischen Phonemen und Buchstaben. Die Beziehung zwischen Laut- und Schriftbild ist ganz und gar nicht einheitlich. Die Buchstabenreihe `sch` bezeichnet in den meisten Fällen den einzelnen Laut, welcher zum Beispiel in `Schwamm` auftritt. Allerdings gibt es Ausnahmen, wie zum Beispiel `Menschenrechtscharta`. Der Buchstabe `u` wird im Englischen in den meisten Fällen nicht wie in Deutsch `und` ausgesprochen. Deswegen hat man ein sogenanntes Internationales Phonetisches Alphabet geschaffen, welches für jede Sprache gestattet, eine Äußerung auf einheitliche Weise aufzuschreiben, sodaß sie von jedem, der dieses Alphabet kennt, in der gleichen Weise gelesen wird. Dieses Alphabet bildet Laute eins zu eins auf Zeichen ab. Hier ist die Zuordnung also ideal.

Die eigentlich sinntragenden Einheiten der Sprache sind allerdings nicht die Laute oder Buchstaben. Sondern sie sind in aller Regel Folgen von Lauten. Folgen von Buchstaben, welche durch ein Satz- oder Leerzeichen getrennt werden, nennen wir ein **Wort**. (Hierbei ist `Satz-` wie in dem vorigen Satz verwendet, allerdings *kein* Wort. Das werden wir jedoch nicht weiter problematisieren.) Worte sind jetzt also Einheiten, welche zwar weiter analysiert werden können (zum Beispiel in Folgen von Buchstaben), welche wir aber in den meisten Fällen nicht weiter analysieren werden. Deswegen wird es oft dazu kommen, daß unser technisches Alphabet A nicht die Menge der Laute oder Buchstaben ist, sondern die der Worte. Da es zum Beispiel im Deutschen unendlich viele Worte gibt, muß man allerdings etwas vorsichtig sein, wie man das exakt verstehen will. Wir werden darauf später, in Abschnitt 4.7, noch eingehen.

Wir haben also Worte als Folgen von Buchstaben bzw. Lauten definiert und einen Satz als Folge von Worten. Dies impliziert unter anderem, daß ein Wort stets und immer in eine solche Folge von Lauten zerlegt werden kann. Die einzelnen Laute heißen auch *Segmente*. Zum Beispiel sind `o`, `d`, `e` und `r` die Segmente von `oder`. Wir wollen diese Zerlegungseigenschaft deswegen **Segmentalität** nennen. Diese ist allerdings bei näherem Hinsehen eine Illusion. Ein Fragesatz unterscheidet sich von einem Aussagesatz durch eine sogenannte *Intonationskontur* (zum Beispiel steigt die Tonhöhe gegen Ende des Satzes an anstatt zu fallen). Diese ist auf die ganze Äußerung verteilt, ohne daß man sie einem spezifischen Segment zugeordnet werden kann. Man nennt sie deswegen auch **suprasegmental**. Ein Beispiel aus der Schriftsprache ist der Schrägdruck. Wenn wir etwa das Wort 'Tafel' `Tafel` schreiben und nicht `Tafel`, so wollen wir das Wort 'Tafel' an dieser Stelle hervorheben. Dies tun wir, indem wir jeden einzelnen Buchstaben unterstreichen. Die Unterstreichung als Hervorhebungsmerkmal kommt dem ganzen Wort zu, nicht einem seiner Segmente. Von diesen Dingen abgesehen ist Sprache jedoch überwiegend segmental. Unabhängig davon, ob jede Äußerung segmentierbar ist, gibt es das Problem, daß wir einem Morphem

nicht einfach eine Zeichenkette als Realisierung geben können. Instruktiv ist hier das Indonesische. Dort wird der Plural durch dir Verdopplung ausgedrückt. So heißt zum Beispiel `anak` 'Kind', und `anak-anak` bedeutet 'Kinder'. Genauso heißt `orang` 'Mensch', und `orang-orang` heißt dementsprechend 'Menschen'. Dem Zeichen für Plural läßt sich offenbar keine spezielle Zeichenkette als Exponent zuordnen, sondern nur die Funktion $f : A^* \to A^* : \vec{x} \mapsto \vec{x}\text{-}\vec{x}$. Trotzdem ist jede einzelne Zeichenkette, im Singular wie im Plural segmental analysierbar, nur ist das Pluralzeichen selbst nicht segmental. Dies ist es im Übrigen auch nicht im Deutschen; hier sind allerdings die Regeln zur Formung des Plurals deutlich komplizierter, wie oben angedeutet.

Allerdings sind auch die Buchstaben oder Laute nicht einfach Einheiten, sondern sind ihrerseits komplexe Gebilde. Phoneme werden durch eine Menge von einfachen Merkmalen spezifiziert, welche man auch **distinktive Merkmale** nennt. Das `p` unterscheidet sich vom `b` dadurch, daß es stimmlos ist, während `b` stimmhaft ist. Stimmlos sind weiter auch `k`, `t`, während wiederum `g` und `d` stimmhaft sind. Dies ist aus folgendem Grund bedeutsam. Es gibt eine phonologische Regel im Deutschen, welche besagt, daß Konsonanten im Silbenauslaut stimmlos werden. So spricht man zum Beispiel `Jagd` als wäre es `Jakt` geschrieben. Man schreibt allerdings nicht so, denn erstens kommt das Wort von `jagen`, sodaß man `g` anstelle von `k` setzt; zweitens ist das `d` in `Jagden` im Silbenanlaut, also stimmhaft. Die Schreibung geht also dahingehend, denjenigen Konsonanten zu nehmen, welcher auftreten würde, wenn der Laut im Silbenanlaut wäre. Die Ausspracheregel ist dann natürlich sehr einfach. Phonematisch gesehen lautet die Regel: ersetze das Merkmal 'stimmhaft' durch das Merkmal 'stimmlos' im Silbenauslaut. Man notiert dies allerdings anders; es existiert nur das Merkmal STIMMHAFT, und es erhält entweder den Wert $+$, falls der Laut stimmhaft ist, oder den Wert $-$, falls der Laut stimmlos ist. Eine Vergleichbares Szenario bietet die Groß- und Kleinschreibung. Worte werden im Deutschen und vielen anderen Sprachen am Satzanfang groß geschrieben, auch wenn sie normalerweise klein geschrieben werden. Offensichtlich spielt der Satzanfang hier die gleiche Rolle wie der Silbenauslaut. Er verwischt den Unterschied zwischen Groß- und Kleinschreibung (man sagt, er neutralisiere die Opposition zwischen Groß– und Kleinschreibung). Im Lexikon werden die Worte deswegen so aufgeschrieben, wie sie geschrieben würden, wenn sie *nicht* am Satzanfang auftreten. Täte man dies nicht, so wäre jeder Anfangsbuchstabe groß, und wir müßten uns zusätzlich merken, ob er im Satzinneren durch einen kleinen ersetzt werden muß oder nicht. Falls sie dann am Anfang eines Satzes stehen, lautet die Regel so: ersetze den ersten Buchstaben falls nötig durch den entsprechenden Großbuchstaben. Damit man dies so sagen kann, muß man Buchstaben entsprechend analysieren. Jeder Buchstabe bekommt entweder das Merkmal [GROSS : $+$] oder das Merkmal [GROSS : $-$]. Der Buchstabe `A` unterscheidet sich von dem Buchstaben `a` zum Beispiel nur durch den Eintrag für das Merkmal GROSS. Da man im Wortinneren im Allgemeinen Großbuchstaben nicht verwendet, erlaubt dies auch eine relativ kompakte Kodierung von Worten, da man im Innern auf die Spezifizierung des Merkmals GROSS verzichten kann.

Wir haben im vorigen Abschnitt relativ ausführlich über Repräsentationen von Zeichenketten gesprochen. In der Sprachwissenschaft ist dies ein großes Thema, allerdings unter der Bezeichnung *Wortstellung*. Dies wollen wir jetzt von seiner rein kombinatorischen Seite analysieren. Sehen wir einmal von Wortklassen ab, so hat jedes Wort einer Sprache auch eine Stelligkeit. Das finite Verb sieht hat zum Beispiel die Stelligkeit 2. Da die Worte Marcus und Paul jeweils die Stelligkeit 0 haben, ist zum Beispiel sieht(Marcus, Paul) ein Term. Wir reden bei diesem Term von sieht als dem **Funktor** und Marcus und Paul als seinen **Argumenten**. In der Syntax redet man auch von **Kopf** und **Komplement**. Ferner ist Marcus das **Subjekt**, und **Objekt**. Die Begriffe 'Subjekt' und 'Objekt' bezeichnen sogenannte **grammatische Relationen**. Die Zuordnung von Argumentstellen zu grammatischen Relationen ist willkürlich und wird vom Lexikon sowie der Grammatik übernommen.

Wie wird nun dieser Term repräsentiert? Die Repräsentation von sieht ist offensichtlich `sieht`, die Repräsentation von Marcus ist `Marcus` und von Paul ist `Paul`, und der ganze Term wird durch die Zeichenkette

```
Marcus sieht Paul.
```

repräsentiert. (Also: das Verb erscheint nach dem Subjekt und vor dem Objekt, und am Ende setzt man einen Punkt.) Dies ist keineswegs so banal wie es den Anschein hat. Denn wir wollen den Term als sprachunabhängiges Gebilde betrachten, welches uns unter anderem eindeutig seine Bedeutung angeben läßt (hier: daß Marcus Paul sieht). Indem wir den Term in einer anderen Sprache repräsentieren, bekommen wir so die Übersetzung unseres deutschen Satzes in die jeweilige andere Sprache. Dies ist, auf ganz elementarem Niveau, die Konzeption von Montague. Wir werden darauf im Kapitel 3 eingehen. Man betrachte folgende Repräsentationen in ausgewählten Sprachen.

	sieht	Marcus	Paul
Englisch	`sees`	`Marcus`	`Paul`
Lateinisch	`vidit`	`Marcus`	`Paulus`
Ungarisch	`látja`	`Marcus`	`Pál`

Im Englischen gilt nun, daß das Verb zwischen seine Argument tritt. Wie im Deutschen ist das Subjekt zuerst:

```
Marcus sees Paul.
```

Englisch wird deswegen auch eine SVO–Sprache genannt, weil in transitiven Konstruktionen das Subjekt vor dem Verb, und dieses vor dem Objekt steht. Dies entspricht im Übrigen der Infixnotation; natürliche Sprachen haben allerdings keine

Klammern, und deswegen kann es zu Mehrdeutigkeiten kommen. Mit den Kürzeln 'S', 'V' und 'O' kann man insgesamt 6 Sprachtypen unterscheiden: SOV, SVO, VSO, OSV, OVS, VOS. Diese sind in den Sprachen der Welt nicht gleichverteilt. Etwa 40 % der Sprachen sind SOV Sprachen, etwa 36 % SVO Sprachen, und noch einmal etwa 6 % sind VSO Sprachen. Somit gilt fast immer, daß das Subjekt vor dem Objekt erscheint, während das Verb entweder vor beiden (VSO), zwischen beiden (SVO) oder aber nach beiden (SOV) steht. Man sagt im ersten Fall, die Sprache sei **Kopfinitial**, im zweiten **Kopfmedial** und im dritten, sie sei **Kopffinal**. Für Neugierige sei erwähnt, daß das Deutsche entgegen dem Anschein eine SOV–Sprache ist, also kopffinal. Dies kann man daran sehen, daß lediglich der finite Anteil des Verbs an die zweite Stelle des Satzes rückt, und dies auch nur im Haupsatz. Man ersieht das unter Anderem aus folgenden Sätzen.

(1.1) `Marcus sieht Paul.`
(1.2) `Marcus will Paul sehen.`
(1.3) `Marcus will Paul sehen können.`
(1.4) `..., weil Marcus Paul sieht.`
(1.5) `..., weil Marcus Paul sehen will.`
(1.6) `..., weil Marcus Paul sehen können will.`

Nun ist es so, daß viele Sprachen alternative Wortstellungen erlauben. Ein instruktives Beispiel ist das Latein. Jede Permutation der Worte des folgenden Satzes repräsentiert unseren Term:

`Marcus vidit Paulum.`

Man beachte allerdings, daß Subjekt und Objekt durch sogenannte *Kasus* unterschieden werden. Wir haben nämlich `Paulum` (im Akkusativ) anstelle von `Paulus` (im Nominativ), wie unsere Tabelle nahelegte. Deswegen muß man die Analyse verfeinern. Wir wollen hier nicht in die Einzelheiten gehen. Im Kapitel 4 werden wir auf Kasusmarkierung noch einmal zu sprechen kommen. Es sei nur gesagt, daß man beim Latein von **freier Wortstellung** spricht. Dies bedeutet allerdings nur, daß die Reihenfolge von Kopf und seinen Argumenten nicht festgelegt ist. Es heißt, daß für jeden Satz grundsätzlich jede Reihenfolge seiner Worte zulässig ist und dasselbe bedeutet.

In natürlichen Sprachen ist die Stelligkeit allerdings nicht eindeutig bestimmt; man spricht hier von **Polyvalenz**. Das Verb `rollen` kann sowohl einstellig sein wie zweistellig. Dies ist in unserer Terminologie nicht vorgesehen. Man kann allerdings die Definitionen so umbauen, daß sie polyvalente Zeichen zulassen.

1.4 Bäume

Zeichenketten kann man auch als Paare $\langle \mathfrak{L}, f\rangle$ betrachten, wo $\mathfrak{L} = \langle L, <\rangle$ eine endliche, linear geordnete Menge ist und $f : L \to A$ eine Funktion, die wir **Markierungsfunktion** nennen wollen. Da L endlich ist, ist $\langle L, <\rangle$ isomorph zu $\langle n, <\rangle$ für ein gewisses n. (Man beachte, daß n ja auch eine Menge ist.) So kann also eine Zeichenkette als ein Tripel $\langle n, <, f\rangle$ angesehen werden oder schlicht als ein Paar $\langle n, f\rangle$, da ja die Ordnung auf n bereits festliegt. Wir werden im Folgenden stets Strukturen von der Gestalt $\langle M, \vec{R}, \ell\rangle$ treffen, wo M eine Menge ist, $\vec{R}$ eine Folge gewisser Relationen auf M und ℓ eine Funktion von M nach A. Diese heißen *Strukturen über A*.

Eine sehr wichtige Struktur in der Sprachanalyse ist der *Baum*. Ein Baum ist ein spezieller Fall eines sogennanten gerichteten Graphen. Ein **gerichteter Graph** ist eine Struktur $\langle G, <\rangle$, wo $< \subseteq G^2$ eine binäre Relation ist. Wie allgemein üblich werden wir $x \leq y$ schreiben, falls $x < y$ oder $x = y$. Ferner sollen x und y **vergleichbar** heißen, wenn $x \leq y$ oder $y \leq x$. Eine **(gerichtete) Kette** der Länge k ist eine Folge $\langle x_i : i < k+1\rangle$ mit $x_i < x_{i+1}$, für alle $i < k$. Eine **ungerichtete Kette** der Länge k ist eine Folge $\langle x_i : i < k+1\rangle$ mit $x_i < x_{i+1}$ oder $x_{i+1} < x_i$, für alle $i < k$. Ein gerichteter Graph heißt **zusammenhängend**, falls zu je zwei Elementen x und y eine ungerichtete Kette von x nach y gibt. Eine gerichtete Kette der Länge k heißt **Zyklus**, falls $x_k = x_0$. Eine zweistellige Relation heißt **zykelfrei**, falls sie nur Zykeln der Länge 0 besitzt. Eine **Wurzel** ist ein Element r derart, daß für alle x eine gerichtete Kette von r nach x existiert.

Definition 1.4.1 *Ein **gerichteter azyklischer transitiver Graph (dat–Graph)** ist ein Paar $\mathfrak{G} = \langle G, <\rangle$ dergestalt, daß $< \subseteq G^2$ eine azyklische transitive Relation ist.*

Definition 1.4.2 *$\mathfrak{G} = \langle G, <\rangle$ heißt ein **Wald**, falls $<$ transitiv und irreflexiv ist und für $x < y, z$ gilt, daß y und z vergleichbar sind. Ein Wald mit einer Wurzel ist ein **Baum**.*

Dazu einige Überlegungen. In einem zusammenhängenden dat–Graphen ist eine Wurzel stets mit jedem von ihr verschiedenen Knoten vergleichbar. Dies folgt aus der Transitivität der Relation. Man mache sich klar, daß in Gegenwart der Transitivität $<$ genau dann azyklisch ist, wenn $<$ irreflexiv ist. Denn wenn $<$ reflexiv ist, so hat $<$ einen Zyklus der Länge 1. Umgekehrt, falls es einen Zyklus $\langle x_i : i < k+1\rangle$ der Länge $k > 0$ gibt, so gilt $x_0 < x_k = x_0$.

Falls $x < y$ und es kein z gibt mit $x < z < y$, so heißt x **Tochter** von y, und y

Mutter von x. Wir schreiben $x \prec y$. Man mache sich klar, daß in endlichen Bäumen Folgendes gilt.

Lemma 1.4.3 *Es sei $\langle B, < \rangle$ ein endlicher Baum. Falls $x < y$, so existiert ein $\widehat{x}$ mit $x \leq \widehat{x} \prec y$ und es existiert ein $\widehat{y}$ mit $x \prec \widehat{y} \leq y$. $\widehat{x}$ und $\widehat{y}$ sind durch x und y jeweils eindeutig bestimmt.* $\dashv$

In unendlichen Strukturen muß dies nicht gelten! Man nehme zum Beispiel die Struktur $\langle \mathbb{Q}, < \rangle$. Diese ist ein dat–Graph, aber Proposition 1.4.3 Ist nicht anwendbar. Ein ebenfalls wichtiges Faktum ist das Vorfahrenlemma. Wir definieren $x \circ y$ durch $x \leq y$ oder $y \leq x$ und sagen x und y **überlappen**.

Lemma 1.4.4 (Vorfahrenlemma) *Es sei $\mathfrak{B}$ ein endlicher Baum. Es seien x und y Knoten, welche nicht überlappen. Dann existieren eindeutig bestimmte u, v und w, sodaß gilt: $x \leq u \prec w$, $y \leq v \prec w$ und $v \neq u$.*

Beweis. Zunächst existiert ein w_0 mit $x, y \leq w_0$, da $\mathfrak{B}$ Baum ist. Nun setze $I := \{w' : x, y \leq w'\}$ und $w := minI$. Da I nicht leer ist und linear geordnet, ist dies wohldefiniert. Nun ist $x, y \leq w$. Da aber x und y nicht vergleichbar sind, ist sogar $x < w$ und $y < w$. Nach Lemma 1.4.3 existieren deswegen eindeutig bestimmte u und v mit $x \leq u \prec w$ und $y \leq v \prec w$. Gewiß ist $v \neq u$, ansonsten wäre w nicht minimal in I. Dies zeigt die Existenz eines solchen Tripels und die Eindeutigkeit von u und v. Daß w auch eindeutig bestimmt ist, sieht man so. Ist $z \in I$ und $z \neq w$, so gilt $x < w < z$ und $y < w < z$. Dann existiert ein $\widehat{w}$ mit $w \leq \widehat{w} \prec z$. Dann ist aber auch $x < \widehat{w} \prec z$ und $y < \widehat{w} \prec z$. Mit dem vorigen Lemma folgt nun wiederum, daß kein Tripel der gewünschten Art existiert. $\dashv$

Ein Knoten y **verzweigt sich** n**–fach nach unten**, falls er genau n Töchter hat, und er verzweigt sich n**–fach nach oben**, falls er genau n Mütter hat. Wir sagen, ein Knoten **verzweige nach oben** (**nach unten**), falls er nach oben (unten) n–fach verzweigt für ein $n > 1$. Ein Wald ist dadurch charakterisiert, daß kein Knoten nach oben verzweigt. Daher reden wir speziell bei Wäldern und Bäumen schlicht von n–fach verzweigenden Knoten, wenn wir Knoten meinen, die sich n–fach *nach unten* verzweigen. Es heißt x **Blatt**, falls kein $y < x$ existiert, d. h. wenn x sich 0–fach nach unten verzweigt. Die Menge der Blätter in $\mathfrak{G}$ wird mit $b(\mathfrak{G})$ bezeichnet. Wir definieren ferner folgende Bezeichnungen.

$$\begin{array}{rcl} \downarrow x & := & \{y : y \leq x\} \\ \uparrow x & := & \{y : y \geq x\} \end{array}$$

Nach Definition eines Baumes ist $\uparrow x$ stets linear geordnet durch die Einschränkung von $<$ auf $\uparrow x$. Ebenso prüft man leicht nach, daß $\downarrow x$ versehen mit der Einschränkung von $<$ wieder ein Baum ist mit Wurzel x.

Wichtig sind auch die Begriffe *Pfad* und *Zweig*. Eine Menge $P \subseteq G$ heißt ein **Pfad**, wenn sie bezüglich $<$ linear geordnet ist und konvex, d. h. mit $x, y \in P$ ist auch $z \in P$ für jedes z mit $x < z < y$. Mit der **Länge** von P bezeichnen wir $|P| - 1$. Ein **Zweig** ist ein bezüglich Mengeninklusion maximaler Pfad. Die **Höhe** von x in einem dat–Graphen, in Zeichen $h_{\mathfrak{G}}(x)$ oder schlicht $h(x)$, ist die maximale Länge eines Zweiges in $\downarrow x$. Die Höhe läßt sich wie folgt definieren.

$$\begin{array}{lcll} h(x) & := & 0, & \text{falls } x \text{ Blatt,} \\ h(x) & := & 1 + max\{h(y) : y \prec x\}, & \text{sonst.} \end{array}$$

Dual dazu ist die **Tiefe** definiert:

$$\begin{array}{lcll} t(x) & := & 0, & \text{falls } x \text{ Wurzel,} \\ t(x) & := & 1 + max\{t(y) : y \succ x\} & \text{sonst.} \end{array}$$

Definition 1.4.5 *Es seien $\mathfrak{G} = \langle G, <_G \rangle$ und $\mathfrak{H} = \langle H, <_H \rangle$ gerichtete Graphen und $G \subseteq H$. Dann heißt $\mathfrak{G}$ ein **Teilgraph** von $\mathfrak{H}$, falls $<_G = <_H \cap G^2$.*

Sind $\mathfrak{G}$ und $\mathfrak{H}$ nun dat–Graphen, Wälder oder Bäume, so heißt $\mathfrak{G}$ ein **Teil–dat–Graph**, **Teilwald** bzw. **Teilbaum**. Ein Teilbaum mit unterliegender Menge $\downarrow x$ wird **Konstituente** genannt.

Definition 1.4.6 *Es sei A ein Alphabet. Ein **dat–Graph (Baum) über** A ist ein Paar $\langle \mathfrak{G}, f \rangle$, wo $\mathfrak{G} = \langle G, < \rangle$ ein dat–Graph (Baum) ist und $f : G \to A$ eine beliebige Funktion.*

Alternativ sprechen wir von einem **dat–Graphen** (**Baum**) **mit Marken in** A, oder schlicht von einem markierten Baum, wenn das Alphabet feststeht. Die Begriffe der Teilstruktur werden analog ausgedehnt auf markierte Strukturen.

Die Baumstruktur soll die hierarchischen Beziehungen zwischen Elementen widerspiegeln, nicht die (zeitlich oder örtlich) linearen. Die letzteren müssen zusätzlich eingeführt werden. Dies geschieht, indem wir eine zusätzliche zweistellige Relation, notiert $\sqsubset$, in die Signatur aufnehmen. Unsere Sprechweise ist so. Wir sagen, x sei **vor** y oder **links von** y, falls $x \sqsubset y$. Wir sagen, x sei **über** y oder **dominiere** y, falls $x > y$. Wir denken uns dabei, daß die Baumstruktur eine zusätzliche hierarchische Struktur über einer Zeichenkette artikuliert. Diese Zeichenkette ist auf den Blättern des Baumes realisiert. Ein beliebiger Knoten x des Baumes entspricht einer Zeichenkette, nämlich derjenigen aller Blätter unterhalb von x. Die Ordnung auf den Blättern wird nun auf eine Ordnung auf dem gesamten Graphen angehoben, indem wir sagen, daß $x \sqsubset y$ immer dann gilt, wenn für alle Blätter $u \leq x$ und alle

Blätter $v \leq y$ gilt $u \sqsubset v$. Diese Definition ist allerdings nur dann unproblematisch, wenn Knoten nach oben nicht verzweigen. Daher werden wir im Folgenden nur noch von Bäumen reden. Die folgende Definition beschreibt die so erhaltenen Strukturen intrinsisch.

Definition 1.4.7 *Ein **geordneter Baum** ist ein Tripel $\langle B, <, \sqsubset \rangle$ derart, daß Fogendes gilt.*

(0) $\langle B, < \rangle$ *ist ein Baum.*
(1) $\sqsubset$ *ist eine lineare, irreflexive Ordnung auf den Blättern.*
(2) *Ist $x \sqsubset z$ sowie $y < x$, so ist auch $y \sqsubset z$.*
Ist $x \sqsubset z$ sowie $y < z$, so ist auch $x \sqsubset y$.
(3) *Falls x kein Blatt ist und für alle $y < x$ gilt $y \sqsubset z$, so ist auch $x \sqsubset z$.*
Falls z kein Blatt ist und für alle $y < z$ gilt $x \sqsubset y$, so ist auch $x \sqsubset z$.

Die Bedingung (2) ist eine Kohärenzforderung an die Ordnung $\sqsubset$, die uns sichert, daß $x \sqsubset y$ nur dann gelten kann, wenn alle Blätter unterhalb von x vor allen Blättern unterhalb von y sind. (3) ist dagegen eine Vollständigkeitsforderung, die sichert, daß dies auch eine zureichende Bedingung ist.

Wir verabreden folgende Bezeichnung. Es sei $x \in G$ ein beliebiger Knoten in einem dat–Graphen. Dann sei $[x] := \downarrow x \cap b(\mathfrak{G})$. Wir nennen dies die **Extension** von x. $[x]$ ist linear geordnet durch $\sqsubset$. Wir schreiben $k(x) := \langle [x], \sqsubset \rangle$ und nennen es die mit x **assoziierte (Zeichen)Kette**. Es kann vorkommen, daß verschiedene Knoten dieselbe assoziierte Kette haben. Die zu dem Graphen assoziierte Kette ist schlicht $k(\mathfrak{G}) := \langle b(\mathfrak{G}), \sqsubset \rangle$. Eine Konstituente heißt **kontinuierlich**, falls die assoziierte Kette eine bezüglich $\sqsubset$ konvexe Teilmenge ist von $k(\mathfrak{G})$.

Theorem 1.4.8 *Es sei $\langle B, < \rangle$ eine Baum und $\sqsubset$ eine lineare Ordnung auf den Blättern. Dann existiert genau eine Relation $\sqsubset' \supseteq \sqsubset$ derart, daß $\langle B, <, \sqsubset' \rangle$ ein geordneter Baum ist.*

Beweis. Wir definieren folgende Schreibweise. Für Mengen M, N sei $M \sqsubset N$ genau dann, wenn für alle $x \in M$ und alle $y \in N$ gilt $x \sqsubset y$. Es gilt dann wegen (2), daß aus $x \sqsubset' y$ folgt $[x] \sqsubset [y]$. Aus (3) wiederum bekommen wir, daß wenn $[x] \sqsubset [y]$, so ist $x \sqsubset' y$. Also gilt

$$(\dagger) \quad x \sqsubset' y \qquad \Leftrightarrow \qquad [x] \sqsubset [y]$$

Daraus bekommen wir sofort die Eindeutigkeit der Fortsetzung. Diese erfüllt (2) und (3). Sie erfüllt (1) nach Voraussetzung. (0) ist ebenfalls nach Vorausstzung erfüllt. $\dashv$

Wir weisen darauf hin, daß die Ordnung $\sqsubset'$ nicht linear sein kann, falls $|B| > 1$. Es kann sogar vorkommen, daß $\sqsubset' = \sqsubset$. Man kann zeigen, daß überlappende Knoten

niemals vergleichbar sein können bezüglich $\sqsubset$. Denn sei $x \circ y$, etwa $x \leq y$. Sei $u \leq x$ ein Blatt. Angenommen, $x \sqsubset y$; dann ist nach (ii) $u \sqsubset y$ und auch $u \sqsubset u$. Dies widerspricht jedoch der Anforderung, daß $\sqsubset$ irreflexiv sein muß. Genauso kann $y \sqsubset x$ nicht gelten. Zwei Knoten sind also in der Tat höchstens dann mittels $\sqsubset$ vergleichbar, wenn sie nicht überlappen. Es ist nun aber gut möglich, daß zwei Knoten *genau dann* $\sqsubset$–vergleichbar sind wenn sie nicht überlappen. In diesem Fall nennen wir $\sqsubset'$ **erschöpfend**. Ein Kriterium dafür wollen wir jetzt entwickeln.

Theorem 1.4.9 *Es sei $\langle B, < \rangle$ ein Baum und $\sqsubset$ eine lineare Ordnung auf den Blättern. Genau dann existiert eine erschöpfende Erweiterung $\sqsubset' \supseteq \sqsubset$, wenn jede Konstituente kontinuierlich ist.*

Beweis. Zunächst einmal existiert immer eine Ordnung, welche den Baum zu einem geordneten Baum macht. Zu klären bleibt, wann die so definierte Ordnung erschöpfend ist. Wir nehmen an, jede Konstituente sei kontinuierlich. Seien x und y nicht überlappende Knoten. Dann ist $[x] \cap [y] = \varnothing$. Deswegen muß $[x] \sqsubset [y]$ oder $[y] \sqsubset [x]$ sein. Also ist $x \sqsubset' y$ oder $y \sqsubset' x$. Die Ordnung ist also erschöpfend. Sei umgekehrt $\sqsubset'$ erschöpfend. Angenommen, u ist ein Blatt mit $u \notin [x]$. Dann überlappt u nicht mit x. Nach Voraussstzung muß $u \sqsubset' x$ oder $x \sqsubset' u$ gelten, also $[u] \sqsubset [x]$ oder $[x] \sqsubset [u]$. Dies bedeutet nichts anderes, als daß entweder $u \sqsubset y$ für alle $y \in [x]$ oder $y \sqsubset u$ für alle $y \in [x]$. Also ist $[x]$ kontinuierlich. x war beliebig, also ist jede Konstituente kontinuierlich. $\dashv$

Lemma 1.4.10 (Konstituentenlemma) *Es sei $\langle B, <, \sqsubset, \ell \rangle$ ein erschöpfend geordneter A–Baum. Ferner sei $p < q$. Dann existiert ein Kontext $C = \langle \vec{u}, \vec{v} \rangle$ derart, daß*

$$k(q) = C(k(p)) = \vec{u} \cdot k(p) \cdot \vec{v}$$

Es gilt nicht die Umkehrung. Ferner gilt, daß durchaus $k(q) = k(p)$ sein kann, auch wenn $q < p$.

Proposition 1.4.11 *Sei $\langle B, <, \sqsubset \rangle$ ein geordneter Baum. Genau dann ist $[x] = [y]$ gleichbedeutend mit $x = y$, wenn es keinen 1–fach verzweigenden Knoten gibt.*

Beweis. Es sei x ein 1–fach verzweigender Knoten mit Tochter y. Dann gilt $[x] = [y]$, aber $x \neq y$. Es sei nun x mindestens 2–fach verzweigend. Sei $\{y_i : i < n\}$ die Menge der Töchter von x, $n > 1$. $[x]$ ist die Vereinigung der $[y_i]$. Daher gilt $[y_i] \subsetneq [x]$ für alle $i < n$. Falls nun $[x] = [z]$, so muß mindestens $x \leq z$ oder $z \leq x$ gelten. Falls aber nach Voraussetzung alle Knoten mindestens zweifach verzweigen, so kann weder $x < z$ noch $z < x$ sein, da dann $[x] \subsetneq [z]$ oder $[z] \subsetneq [x]$ sein müßte. $\dashv$

Wir sagen, ein Baum verzweige **echt**, falls es keinen 1–fach verzweigenden Knoten gibt. Eine andere Methode, Bäume zu definieren, ist die folgende. Es sei B eine Menge, und $\prec$ eine zykelfreie Relation auf B. Dann ist $\langle B, < \rangle$ ein Baum, wo $<:=\prec^+$ die transitive Hülle von $\prec$ ist. Ferner ist $x \prec y$ genau dann, wenn x Tochter von y ist. Es bezeichne $T(x)$ die Menge der Töchter von x. Nun sei zusätzlich eine Relation P gegeben, derart, daß (a) $x\ P\ y$ nur dann, wenn $x, y \in T(z)$ für ein gewisses z, (b) P linear auf $T(z)$ ist für jedes z. P ordnet also die Mengen der Töchterknoten linear. Dann setze $x \sqsubset y$ genau dann, wenn $x' \geq x$ existiert und $y' \geq y$ mit $x'\ P\ y'$. $\prec$ und P sind die unmittelbare Nachbar–Relationen. Der folgende Satz liefert uns die Rechtfertigung dafür, daß $\langle B, <, \sqsubset \rangle$ ein erschöpfend geordneter Baum ist.

Proposition 1.4.12 *Es sei $\langle B, <, \sqsubset \rangle$ ein erschöpfend geordneter Baum. Dann ist $x \sqsubset y$ genau dann, wenn $x' \geq x$ und $y' \geq y$ existieren, welche gemeinsame Töchter eines Knotens z sind, und es ist $x' \sqsubset y'$.*

Zum Schluß sei noch ein weiteres nützliches Konzept erwähnt, nämlich das der *Konstituentenstruktur*.

Definition 1.4.13 *Es sei M eine beliebige Menge. Eine **Konstituentenstruktur** über M ist ein System $\mathfrak{C}$ von Teilmengen von M mit folgenden Eigenschaften. (0) $\{x\} \in \mathfrak{C}$ für jedes $x \in M$, (i) $\varnothing \notin \mathfrak{C}$, $M \in \mathfrak{C}$, (ii) falls $S, T \in \mathfrak{C}$ und $S \nsubseteq T$ sowie $T \nsubseteq S$, so $S \cap T = \varnothing$.*

Proposition 1.4.14 *Es sei M eine nichtleere Menge. Es besteht eine eineindeutige Zuordnung zwischen Konstituentenstrukturen über M und echt verzweigenden Bäumen, deren Menge von Blättern $\{\{x\} : x \in M\}$ ist.*

Beweis. Sei $\langle M, \mathfrak{C} \rangle$ eine Konstituentenstruktur. Dann ist $\langle \mathfrak{C}, \subsetneq \rangle$ ein Baum. Dazu muß man prüfen, daß $\subsetneq$ irreflexiv und transitiv ist, sowie, daß es eine Wurzel gibt. Dies ist leicht. Ferner, sei $S \subsetneq T, U$. Dann ist $U \cap T \supseteq S \neq \varnothing$, wegen Bedingung (i). Also muß $U \subseteq T$ oder $T \subseteq U$ gelten. Dies bedeutet nichts anderes, als daß T und U vergleichbar sind. Die Menge der Blätter ist genau $\{\{x\} : x \in M\}$, wie man leicht nachprüft. Sei nun umgekehrt $\mathfrak{B} = \langle B, < \rangle$ ein Baum. Definiere $M = b(\mathfrak{B})$. Setze $\mathfrak{C} = \{[x] : x \in B\}$. Dies ist eine Konstituentenstruktur. Denn zunächst ist für jedes Blatt u $[u] = \{u\}$, also sind alle $\{u\} \in \mathfrak{C}$. Weiter ist für jedes x $[x]$ nicht leer, da der Baum endlich ist. Außerdem existiert eine Wurzel, r, und es gilt $[r] = B$, nach Definition. Zuletzt sei $[x] \nsubseteq [y]$ und $[y] \nsubseteq [x]$. Dann sind x und y unvergleichbar. Falls nun u ein Blatt ist und $u \in [x]$, so ist $u \leq x$. $u \leq y$ kann dann nicht gelten; denn $\uparrow u$ ist linear und dann wären x und y vergleichbar. Ebenso folgt aus $u \leq y$ sofort $u \not\leq x$. Daher ist $[x] \cap [y] = \varnothing$. $\dashv$

Im allgemeinen Fall kann man einem Baum stets eine Konstituentenstruktur zuordnen. Aus dieser läßt sich allerdings der Baum nur dann rekonstruieren, wenn er echt verzweigend ist.

Der Begriff der Konstituentenstruktur kann nun erweitert werden zu dem Begriff einer *geordneten* Konstituentenstruktur. Dabei fordert man analoge Bedingungen, wie sie bei geordneten Bäumen gefordert sind. Weiter können wir bei allen Strukturen auch noch *Marken* einführen, das heißt Funktionen $f : G \to A$, wo G der Grundbereich und A das Alphabet ist. Die Begriffsbildung verläuft in völliger Analogie zu dem unmarkierten Fall, so daß wir auch von *markierten* Konstituentenstrukturen, *markierten geordneten* Konstituentenstrukturen etc. reden können.

Wir wollen uns nun der Repräsentation von Termen durch (geordnete) Bäume zuwenden. Es gibt zwei verschiedene Methoden, einen Term durch einen Baum darzustellen, und beide haben ihren unterschiedlichen Nutzen. Bevor wir sie einführen, werden wir allerdings noch eine wichtige Methode bereitstellen, geordnete Bäume durch Mengen von Zahlenfolgen zu codieren.

Definition 1.4.15 *Es sei $B \subseteq \omega^*$ eine Menge von endlichen Zahlenfolgen. B heißt* ***Baumbereich****, falls Folgendes gilt:*

1. *Ist $\vec{x} \cdot i \in B$, so ist $\vec{x} \in B$.*
2. *Ist $\vec{x} \cdot i \in B$ und $j < i$, so ist auch $\vec{x} \cdot j \in B$.*

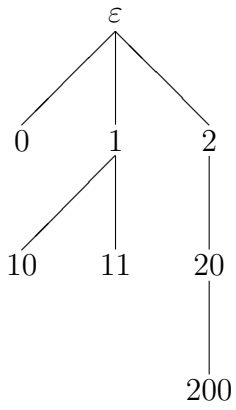
Einem Baumbereich B ordnen wir einen geordneten Baum zu, indem wir als Knotenmenge gerade B wählen und die Relationen $<$ und $\sqsubset$ uniform für alle Baumbereiche definieren. Es sei $\vec{x} < \vec{y}$ genau dann, wenn $\vec{y}$ echtes Präfix von $\vec{x}$ ist. Ferner sei $\vec{x} \sqsubset \vec{y}$ genau dann, wenn es Zahlen i, j gibt und Zahlenfolgen $\vec{u}$, $\vec{v}$, $\vec{w}$ mit (a) $i < j$ und (b) $\vec{x} = \vec{u} \cdot i \cdot \vec{v}$, $\vec{y} = \vec{u} \cdot j \cdot \vec{w}$. Mit diesen Relationen wird B tatsächlich zu einem erschöpfend geordneten Baum, wie man leicht nachrechnet. In Figur 1.1 wird der Baumbereich $B = \{\varepsilon, 0, 1, 2, 10, 11, 20, 200\}$ gezeigt. Ist B ein Baumbereich und $\vec{x} \in B$, so setze

$$B/\vec{x} := \{\vec{y} : \vec{x} \cdot \vec{y} \in B\}$$

Dies ist anschaulich gesprochen die Konstituente von B unterhalb des Knotens $\vec{x}$.

Sei nun umgekehrt ein erschöpfend geordneter Baum $\langle B, <, \sqsubset\rangle$ gegeben. Wir definieren den zugeordneten Baumbereich, B^β durch Induktion über die Tiefe $t(x)$ wie folgt. Ist $t(x) = 0$, so sei $x^\beta := \varepsilon$. In diesem Fall ist x die Wurzel des Baumes. Ist nun x^β bereits definiert, und y Tochter von x, so sei $y^\beta := x^\beta \cdot i$, wenn y genau die $i$te Tochter von links ist. (Es gilt somit $|x^\beta| = t(x)$.) Wir können ganz leicht sehen, daß die so definierte Menge ein Baumbereich ist. Denn zunächst ist ja $\vec{u} \in B^\beta$,

Abbildung 1.1: Baumbereich



sobald $\vec{u} \cdot j \in B^\beta$ ist für ein j. Also gilt 1. Ferner, ist $\vec{u} \cdot i \in B^\beta$ etwa $\vec{u} \cdot i = y^\beta$, so ist y die $i$te Tochter von einem Knoten x. Sei nun $j < i$. Dann sei z die $j$te Tochter von x (von links). Diese existiert gewiß, und es ist $z^\beta = \vec{u} \cdot j$. Nun ist man allerdings noch nicht fertig. Denn wir wollen noch zeigen, daß auf dem Baumbereich definierten kanonischen Relationen genau die sind, welche auf dem Baum definiert sind. Mit anderen Worten, wir wollen zeigen, daß $x \mapsto x^\beta$ ein Isomorphismus ist. Dazu müssen wir nur zeigen, daß diese Abbildung die Relationen *Tochter* und *linker Nachbar* treu erhält. Dies ist aber leicht zu zeigen.

Theorem 1.4.16 *Sei $\mathfrak{B} = \langle B, <, \sqsubset \rangle$ ein endlicher, erschöpfend geordneter Baum. Die Abbildung $x \mapsto x^\beta$ ist ein Isomorphismus von $\mathfrak{B}$ auf den zugeordneten Baumbereich $\langle \mathfrak{B}^\beta, <, \sqsubset \rangle$. Ferner gilt: genau dann ist $\mathfrak{B} \cong \mathfrak{C}$, wenn $\mathfrak{B}^\beta = \mathfrak{C}^\beta$.*

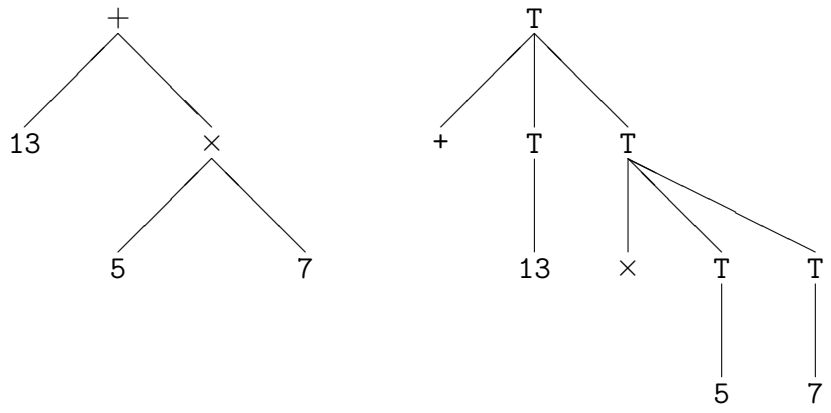
Mit Hilfe der Baumbereiche können wir nun die Kodierung von Termen relativ leicht vornehmen. Terme werden schlicht in Baumbereiche mit Marken übersetzt. Wir ordnen also jedem Term t einen Baumbereich t^b und eine Markierungsfunktion t^λ zu. Der zu t assoziierte markierte Baumbereich ist dann $t^m := \langle t^b, t^\lambda \rangle$, der wie folgt definiert ist. Der unterliegende Baumbereich des Terms x ist $\{\varepsilon\}$, und ε trägt die Marke x. Der Term $s := f(t_0, \ldots, t_{n-1})$ wird in folgenden Baumbereich übersetzt:

$$s^b := \{\varepsilon\} \cup \bigcup_{i<n} \{i \cdot \vec{x} : \vec{x} \in t_i^b\}$$

Dann ist s^λ wie folgt definiert:

$$\begin{array}{lcl} s^\lambda(\varepsilon) & := & f \\ s^\lambda(j \cdot \vec{x}) & := & t_j^\lambda(\vec{x}) \end{array}$$

Abbildung 1.2: Dependenz– und Strukturkodierung



Dies bedeutet, daß s^m aus einer Wurzel mit Namen f besteht und n Töchter hat, welche zu den Baumbereichen von t_0, …, $t_{\Omega(f)-1}$ isomorph sind. Wir nennen diese Kodierung, welche t auf t^m abbildet, die **Dependenzkodierung**. Sie ist sparsamer als die folgende Kodierung, welche wir die **Strukturkodierung** nennen. Wir wählen dazu ein neues Symbol T und definieren induktiv zu jedem Term t einen Baumbereich t^c und eine Markierungsfunktion t^μ. Setze $x^c := \{\varepsilon, 0\}$, $x^\mu(\varepsilon) := \mathtt{T}$, $x^\mu(0) := x$. Ferner sei für $s = f(t_0, \ldots, t_{n-1})$

$$\begin{array}{lcl} s^c & := & \{\varepsilon, 0\} \cup \bigcup_{0<i<n+1} \{i \cdot \vec{x} : \vec{x} \in t_i^c\} \\ \\ s^\mu(\varepsilon) & := & \mathtt{T} \\ s^\mu(0) & := & f \\ s^\mu(j+1 \cdot \vec{x}) & := & t_j^\mu(\vec{x}) \end{array}$$

(Man vergleiche die Strukturkodierung mit der assoziierten klammerfreien Zeichenkette.) In Figur 1.2 sind beide Kodierungen für den Term `(13 + (5 × 7))` im Vergleich gezeigt. Der Vorteil der Strukturkodierung ist daß die dem Term assoziierte Zeichenkette auch die assoziierte Zeichenkette des Baumes (bzw. des Baumbereiches) ist.

Übung 7. Analog zu erschöpfenden Ordnungen auf Bäumen sind erschöpfende Ordnungen auf Konstituentenstrukturen definiert. Zeigen Sie, daß eine Ordnung auf den Blättern eines Baumes genau dann zu einer erschöpfenden Ordnung erweiterbar ist, wenn sie es schon auf der zugehörigen Konstituentenstruktur ist.

Übung 8. Es sei $\mathfrak{B} = \langle B, < \rangle$ ein Baum und $\sqsubset$ eine 2-stellige Relation dergestalt, daß $x \sqsubset y$ nur dann gilt, wenn x, y Töchter desselben Knotens sind. Ferner sollen die Töchter eines Knotens jeweils durch $\sqsubset$ linear geordnet sein. Man zeige, daß es

eine erschöpfende Relation auf B gibt, welche $\sqsubset$ enthält.

Übung 9. Man zeige: Die Anzahl der 2–fach verzweigenden erschöpfend geordneten Bäume mit einer gegebenen Menge von $n+1$ Blättern ist genau

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Die Zahlen C_n heißen **Catalan Zahlen**.

Übung 10. Zeigen Sie, daß $C_n < \frac{1}{n+1}4^n$ ist. (Im Lehrbuch der Analysis von Heuser, Teil 1, S. 505, befindet sich ein Beweis, daß sich $\binom{2n}{n}$ im Limes der Folge $\frac{4^n}{\sqrt{\pi n}}$ nähert. Man kann sogar zeigen, daß letztere Folge die erste majorisiert. Für die in der Übung verlangte Tatsache existiert allerdings ein elementarer Beweis.)

Übung 11. Es sei L endlich mit n Elementen und $<$ eine lineare Ordnung auf L. Man konstruiere einen Isomorphismus von $\langle L, <\rangle$ nach $\langle n, <\rangle$.

1.5 Ersetzungssysteme

Sprachen sind nach Definition 1.2.9 beliebige Mengen von Zeichenketten über einem (endlichen) Alphabet. Dennoch sind die Sprachen, mit denen man es normalerweise zu tun hat, solche Mengen von Zeichenketten, welche man durch einen endlichen Prozeß gewinnen kann. Dies können Prozesse sein, welche Zeichenketten iterativ erzeugen, oder solche, welche zunächst andere Strukturen (z. B. Bäume) erzeugen, aus denen man dann wiederum Zeichenketten gewinnen kann. Der zweifelsohne populärste Ansatz ist der über Zeichenketten.

Definition 1.5.1 *Es sei A eine Menge. Ein **Semi–Thue System** über A ist eine endliche Menge $T = \{\langle \vec{x}_i, \vec{y}_i\rangle : i < m\}$ von Paaren von Zeichenketten über A. Ist T gegeben, so sei $\vec{u} \Rightarrow^1_T \vec{v}$, falls für gewisse $\vec{s}, \vec{t}$ und ein $i < m$ gilt $\vec{u} = \vec{s} \cdot \vec{x}_i \cdot \vec{t}$ und $\vec{v} = \vec{s} \cdot \vec{y}_i \cdot \vec{t}$. Es ist $\vec{u} \Rightarrow^{n+1}_T \vec{v}$, falls ein $\vec{z}$ existiert mit $\vec{u} \Rightarrow^1_T \vec{z} \Rightarrow^n_T \vec{v}$. Schließlich ist $\vec{u} \Rightarrow^*_T \vec{v}$ falls $\vec{u} \Rightarrow^n_T \vec{v}$ für ein $n \in \omega$. Wir sagen dann, $\vec{v}$ sei **in** T **aus** $\vec{u}$ **herleitbar**.*

Man kann $\vec{u} \Rightarrow \vec{v}$ auch wie folgt definieren: es existiert ein Kontext C und ein $\langle \vec{x}, \vec{y}\rangle \in T$ derart, daß gilt $\vec{u} = C(\vec{x})$ und $\vec{y} = C(\vec{y})$. Da das Semi–Thue System endlich ist, kommen dort nur endlich viele Buchstaben vor; deswegen genügt es, wenn man A endlich wählt. (Siehe dazu die Übungen.) Ein Semi–Thue System T heißt **Thue–System**, falls mit $\langle \vec{x}, \vec{y}\rangle \in T$ auch $\langle \vec{y}, \vec{x}\rangle \in T$. In diesem Falle kann $\vec{v}$ aus $\vec{u}$ genau dann hergeleitet werden, wenn auch $\vec{u}$ aus $\vec{v}$ hergeleitet werden kann. Eine **Ableitung** von $\vec{y}$ aus $\vec{x}$ in T ist eine endliche Folge $\langle \vec{v}_i : i < n+1\rangle$ derart, daß $\vec{v}_0 = \vec{x}$ und $\vec{v}_n = \vec{y}$, und für alle $i < n$ ist $\vec{v}_i \Rightarrow_T \vec{v}_{i+1}$. Die **Länge** dieser Ableitung ist n.

Eine *Grammatik* unterscheidet sich von einem Semi–Thue System wie folgt. Erstens wird eine Unterscheidung zwischen dem eigentlichen Alphabet und einem Hilfsalphabet vorgenommen, und zweitens wird die Sprache über ein spezielles Symbol, das sogenannte *Startsymbol* definiert.

Definition 1.5.2 *Eine **Grammatik** ist ein Quadrupel $G = \langle S, N, A, R\rangle$, bei dem N, A nichtleere, endliche und disjunkte Mengen sind, $S \in N$ und R ein Semi–Thue–System über $N \cup A$, bei dem $\langle \vec{\gamma}, \vec{\delta}\rangle \in R$ nur dann, wenn $\vec{\gamma} \notin A^*$. Es heißt S das **Startsymbol**, N das **Alphabet der nichtterminalen Symbole**, A das **Alphabet der Terminalsymbole** und R die **Menge der Regeln von** G.*

Wir wählen in der Regel $S = \mathtt{S}$. Dies ist jedoch nicht notwendig. Der Leser sei also gewarnt, daß `S` nicht immer das Startsymbol sein muß, es jedoch immer dann ist, wenn nichts anderes gesagt wird. Nichtterminalsymbole werden durch Großbuchstaben bezeichnet, Terminalsymbole durch Kleinbuchstaben. Ein griechischer Buchstabe wird benutzt, wenn es sich sowohl um ein Terminalsymbol wie auch um ein Nichtterminalsymbol handeln kann. Der Gebrauch der Vektorpfeile ist selbst erklärend. Wir schreiben $G \vdash \vec{\gamma}$ oder auch $\vdash_G \vec{\gamma}$ im Falle, daß $S \Rightarrow_R \vec{\gamma}$ und sagen, G **erzeuge** $\vec{\gamma}$. Außerdem schreiben wir $\vec{\gamma} \vdash_G \vec{\delta}$, falls $\vec{\gamma} \Rightarrow_R \vec{\delta}$. Die durch die Grammatik erzeugte Sprache ist definiert durch

$$L(G) := \{\vec{x} \in A^* : G \vdash \vec{x}\}$$

Man beachte, daß G Zeichenketten erzeugt, welche sowohl nichtterminale als auch terminale Symbole enthält. Jedoch sollen Zeichenketten, welche nichtterminale Symbole enthalten, nicht zu der von G erzeugten Sprache gehören. Eine Grammatik ist also ein Semi–Thue Prozeß, bei dem wir bestimmen, wie eine Ableitung anfängt und wann sie abgebrochen werden darf. Gegeben eine Grammatik G, so nennen wir das **Analyseproblem** (oder **Parsingproblem**) für G das Problem, zu gegeber Zeichenkette (1.) zu sagen, ob sie aus G ableitbar ist, und (2.) falls sie ableitbar ist, eine konkrete Ableitung zu benennen. Dabei ist (1.) alleine das **Erkennungsproblem** für G.

Eine Regel $\langle \vec{\alpha}, \vec{\beta}\rangle$ wird auch oft **Produktion** genannt und oft durch $\vec{\alpha} \to \vec{\beta}$ notiert. Wir nennen $\vec{\alpha}$ schlicht die **linke Seite** und $\vec{\beta}$ die **rechte Seite** der Produktion. Die **Produktivität** $p(\rho)$ einer Regel $\rho = \vec{\alpha} \to \vec{\beta}$ ist die Differenz $|\vec{\beta}| - |\vec{\alpha}|$. ρ heißt **expandierend** falls $p(\rho) \geq 0$, **strikt expandierend** falls $p(\rho) > 0$ und **kontrahierend**, falls $p(\rho) < 0$. Eine Regel heißt **terminal**, falls sie von der Form $\vec{\alpha} \to \vec{x}$ (d. h. mit $\vec{x} \in A^*$) ist.

Der Begriff einer Grammatik ist relativ allgemein. Zwar gibt es nur abzählbar viele Grammatiken über einem gegebenen Alphabet — also auch nur abzählbar viele Sprachen —, welche von Grammatiken erzeugt werden können; trotzdem ist

die Vielfalt dieser Sprachen sehr groß und sie können sehr kompliziert sein. Wie sich herausstellen wird, ist jede rekursiv aufzählbare Sprache schon erzeugbar durch eine Grammatik. Dies legt nahe, den Begriff der Grammatik einzuengen. Noam Chomsky hat folgende Hierarchie von Grammatiken und Sprachen vorgeschlagen. (Hierbei ist X_ε kurz für $X \cup \{\varepsilon\}$.)

* Eine Grammatik heißt im allgemeinen Fall vom **Typ 0**.
* Eine Grammatik heißt vom **Typ 1** oder **kontextsensitiv**, falls alle Regeln von der Form $\vec{\delta}_1 X \vec{\delta}_2 \to \vec{\delta}_1 \vec{\alpha} \vec{\delta}_2$ sind, und entweder ist stets $\vec{\alpha} \neq \varepsilon$, oder aber $S \to \varepsilon$ ist eine Regel, und S ist niemals auf der rechten Seite einer Regel.
* Eine Grammatik heißt vom **Typ 2** oder **kontextfrei**, falls sie kontextsensitiv ist und alle Regeln von der Form $X \to \vec{\alpha}$.
* Eine Grammatik heißt vom **Typ 3** oder **regulär**, falls sie kontextfrei ist und alle Regeln von der Form $X \to \vec{\alpha}$ mit $\vec{\alpha} \in A_\varepsilon \cdot N_\varepsilon$.

Eine *Sprache* heißt vom **Typ i**, falls sie von einer Grammatik vom Typ i erzeugt werden kann. Dabei ist unerheblich, ob es auch Grammatiken vom Typ j, $j < i$, geben kann, die diese Sprache erzeugen.

Wir geben Beispiele von Grammatiken vom Typ 3, 2 und 0. Es gibt eine reguläre Grammatik, mit der man einfache Zahlausdrücke erzeugen kann. Hierbei ist ein einfacher Zahlausdruck ein Paar von Ziffernfolgen, getrennt durch ein Komma und mit einem optionalen Vorzeichen. Die Grammatik ist wir folgt. Die Menge der Terminalsymbole ist $\{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{+}, \mathtt{-}, \mathtt{,}\}$, die Menge der Nichtterminalsymbole ist $\{\mathtt{V}, \mathtt{Z}, \mathtt{F}, \mathtt{K}, \mathtt{M}\}$. Das Startsymbol ist `V` und die Regeln lauten

$$
\begin{array}{lcl}
\mathtt{V} & \to & \mathtt{+\,Z} \mid \mathtt{-\,Z} \mid \mathtt{Z} \\
\mathtt{Z} & \to & \mathtt{0\,F} \mid \mathtt{1\,F} \mid \mathtt{2\,F} \mid \ldots \mid \mathtt{9\,F} \\
\mathtt{F} & \to & \mathtt{0\,F} \mid \mathtt{1\,F} \mid \mathtt{2\,F} \mid \ldots \mid \mathtt{9\,F} \mid \mathtt{K} \\
\mathtt{K} & \to & \varepsilon \mid \mathtt{,\,M} \\
\mathtt{M} & \to & \mathtt{0\,M} \mid \mathtt{1\,M} \mid \mathtt{2\,M} \mid \ldots \mid \mathtt{9\,M} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{2} \mid \ldots \mid \mathtt{9}
\end{array}
$$

Hierbei haben wir uns der folgenden, allgemein üblichen Schreibweise bedient. Das Symbol '|' auf der rechten Seite des Pfeils bedeutet, daß es sich um mehrere Produktionen handelt, welche jeweils dasselbe Nichtterminalsymbol ersetzen. So kann zum Beispiel `V` wahlweise durch `+Z`, `-Z` oder durch `Z` ersetzt werden. Die Syntax der Sprache `Algol` wurde ursprünglich in dieser Form notiert, wobei anstatt des Pfeils '::=' geschrieben wurde. Diese Schreibweise nennt man auch **Backus–Naur Form**.

Die Mengen der Terme über einer endlichen Signatur und endlicher Basismenge X können wir durch eine kontextfreie Grammatik erzeugen. Sei $F = \{\mathtt{F}_i : i < m\}$

und $\Omega(j) := \Omega(\mathtt{F}_i)$.

$$\begin{array}{lcll} \mathtt{T} & \to & x & x \in X \\ \mathtt{T} & \to & \mathtt{F}_i\mathtt{T}^{\Omega(i)} & (i < p) \end{array}$$

Damit die Menge der Regeln endlich ist, muß deswegen auch F endlich sein. Das Startsymbol ist $\mathtt{T}$. Diese Grammatik erzeugt die assoziierten Zeichenketten in Polnischer Notation. Man beachte auch, daß die Grammatik genau der Strukturkodierung entspricht. In der Tat erzeugt sie genau die Bäume, welche der Strukturkodierung entsprechen. Doch davon später. Die eben beschrieben Grammatik liefert Terme in Strukturkodierung. Falls wir Dependenzkodierung haben wollen, müssen wir folgende Grammatik wählen.

$$\mathtt{F}_i \quad \to \quad \mathtt{F}_{j_0}\mathtt{F}_{j_1}\ldots\mathtt{F}_{j_{\Omega(i)-1}}$$

Dies ist ein Regelschema, welches für endlich viele Regeln steht.) Hierbei muß man allerdings jedes Symbol als Startsymbol wählen dürfen. Zu dieser Problematik näheres weiter unten.

Als Beispiel für eine Typ 0 Grammatik wollen wir folgende Grammatik nehmen, welche wir dem Buch von Salomaa [44] entnommen haben.

$$\begin{array}{llcllcl} \text{(a)} & \mathtt{X}_0 & \to & \mathtt{a}, & \mathtt{X}_0 & \to & \mathtt{aXX}_2\mathtt{Z}, \\ \text{(b)} & \mathtt{X}_2\mathtt{Z} & \to & \mathtt{aa}, & & & \\ \text{(c)} & \mathtt{Xa} & \to & \mathtt{aa}, & \mathtt{Ya} & \to & \mathtt{aa}, \\ \text{(d)} & \mathtt{X}_2\mathtt{Z} & \to & \mathtt{Y}_1\mathtt{YXZ} & & & \\ \text{(e)} & \mathtt{XX}_1 & \to & \mathtt{X}_1\mathtt{YX}, & \mathtt{YX}_1 & \to & \mathtt{Y}_1\mathtt{YX}, \\ \text{(f)} & \mathtt{XY}_1 & \to & \mathtt{X}_1\mathtt{Y}, & \mathtt{YY}_1 & \to & \mathtt{Y}_1\mathtt{Y}, \\ \text{(g)} & \mathtt{aX}_1 & \to & \mathtt{aXXYX}_2, & & & \\ \text{(h)} & \mathtt{X}_2\mathtt{Y} & \to & \mathtt{XY}_2, & \mathtt{Y}_2\mathtt{Y} & \to & \mathtt{YY}_2, \\ & & & & \mathtt{Y}_2\mathtt{X} & \to & \mathtt{YX}_2\ . \end{array}$$

Als Startsymbol dient $\mathtt{X}_0$. Diese Grammatik erzeugt die Sprache $\{\mathtt{a}^{n^2} : n > 0\}$. Dies zeigt man so. Zunächst kann man mit (a) entweder die Kette $\mathtt{a}$ oder die Kette $\mathtt{aXX}_2\mathtt{Z}$ erzeugen. Sei nun $\vec{\gamma}_i = \mathtt{aX}\vec{\delta}_i\mathtt{X}_2\mathtt{Z}$, $\vec{\delta}_i \in \{\mathtt{X},\mathtt{Y}\}^*$. Wir betrachten Ableitungen, die von $\vec{\gamma}_i$ zu einem terminalen Wort führen. Zunächst ist nur (b) oder (d) anwendbar. Es werde (b) angewandt. Dann kann man nur noch mit (c) fortfahren, und dann wird ein Wort der Länge $1+3+|\vec{\gamma}_i|$ erzeugt. Da wir nur einen Buchstaben haben, ist das Wort damit schon eindeutig bestimmt. Nehmen wir nun an, (d) werde angewendet. Dann bekommen wir das Wort $\mathtt{aX}\vec{\gamma}_i\mathtt{Y}_1\mathtt{YXZ}$. Die einzige Möglichkeit weiterzumachen, ist (e) und (f) anzuwenden und damit den Index 1 nach links zu schieben. Diese Prozedur resultiert darin, daß vor jedes Vorkommen von $\mathtt{X}$ ein $\mathtt{Y}$ gesetzt wird. Wir bekommen so ein neues Wort $\mathtt{aXXYX}_2\vec{\delta}_i'\mathtt{YYXZ}$. Nun bleibt keine Wahl, als den Index 2 mit Hilfe von (h) nach rechts zu schieben. Dies ergibt ein Wort $\vec{\gamma}_{i+1} = \mathtt{aX}\vec{\delta}_{i+1}\mathtt{X}_2\mathtt{Z}$ mit $\vec{\delta}_{i+1} = \mathtt{XYX}\vec{\delta}_i'\mathtt{YY}$. Wir haben

$$|\vec{\delta}_{i+1}| = |\vec{\delta}_i| + \ell_x(\vec{\delta}_i) + 5$$

wo $\ell_x(\delta_i)$ die Anzahl der $\mathtt{X}$ in $\vec{\delta_i}$ zählt. Da nun $\ell_x(\vec{\delta}_{i+1}) = \ell_x(\vec{\delta_i}) + 2$, so schließen wir, daß $\ell_x(\vec{\delta_i}) = 2i$ ist und somit $|\vec{\delta_i}| = (i+1)^2 - 4$, $i > 0$.

Bei der Definition einer kontextsensitiven Grammatik muß man Folgendes bedenken. Der Intention nach ist eine kontextsensitive Grammatik eine Grammatik, in der keine Regel kontrahierend ist. Allerdings schließt eine solche Definition aus, daß ε der durch diese Grammatiken erzeugbaren Sprachen angehört. Also muß zumindest die Regel $S \to \varepsilon$ zugelassen werden. Hat man allerdings eine solche Regel, so muß man verhindern, daß diese Regel nach Anwendung einer anderen Regel angewendet werden kann, um nicht den Effekt zu zerstören, daß — bis auf die erwähnte Ausnahme — eine Ableitung niemals die Kette verkleinern kann. Dies erreicht man, indem man verlangt, daß das Startsymbol nie auf der rechten Seite einer Produktion auftauchen kann. Man beachte, daß diese Bedingung auch an eine kontextfreie Grammatik gestellt wird. Dies ist im Prinzip nicht nötig. Wir können Grammatiken unter Wahrung der erzeugten Sprache so umformulieren, daß S niemals auf der rechten Seite einer Produktion steht. Man ersetze einfach in allen Regeln, welche nicht von der Form $S \to \vec{\beta}$ sind, S durch Z, wobei Z ein neues Symbol ist; ferner nehme man für jede Regel $S \to \vec{\beta}$, noch einmal die Regel $Z \to \vec{\beta}$ hinzu. Dies garantiert, daß in einer Ableitung das Symbol S nur am Anfang steht. Die erzeugte Sprache ist gleichgeblieben. Man möge sich überzeugen, daß auch der Typ der Grammatik sich nicht geändert hat.

Die Menge der regulären Grammatiken werde mit RG, die der kontextfreien mit CFG, die der kontextsensitiven mit CSG und die der Typ 0 Grammatiken mit GG. Entsprechend werden die Mengen der durch sie erzeugbaren Sprachen mit RS, CFS, CSS und GS bezeichnet. Die Grammatiktypen bilden eine Hierarchie, das heißt, wir haben

$$\mathrm{RG} \subsetneq \mathrm{CFG} \subsetneq \mathrm{CSG} \subsetneq \mathrm{GG}$$

Dies ist nicht schwer zu sehen. Daraus folgt unmittelbar, daß die entsprechenden Sprachmengen analog durch Inklusion geordnet sind. Nicht unmittelbar klar ist allerdings, daß auch die zugehörigen Sprachmengen *echt* ineinander enthalten sind. Wir haben nämlich auch

$$\mathrm{RS} \subsetneq \mathrm{CFS} \subsetneq \mathrm{CSS} \subsetneq \mathrm{GS}$$

Wir werden die Nachweise der Echtheit dieser Inklusionen im einzelnen erbringen. Wir werden in Abschnitt 1.7 (Satz 1.7.8) zeigen, daß es Sprachen vom Typ 0 gibt, welche nicht vom Typ 1 sind. Ferner wird aus dem Pumplemma (Satz 1.6.13) für kontextfreie Sprachen folgen, daß $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \in \omega\}$ nicht kontextfrei ist. Diese ist aber eine Typ 1 Sprache (der Nachweis soll in den Übungen erbracht werden). Als Letztes wird sich zeigen, daß $\{\mathtt{a}^n\mathtt{b}^n : n \in \omega\}$ zwar sicherlich kontextfrei ist, aber nicht regulär. (Siehe Übung 2.1.) Es sollte erwähnt werden, daß üblicherweise in kontextfreien Grammatiken auch Regeln der Form $X \to \varepsilon$, d. h. kontrahierende

Regeln, zugelassen werden. In den Übungsaufgaben soll gezeigt werden, daß diese Definition in Bezug auf die erzeugten Sprachen nicht allgemeiner ist. Allerdings ist der Fall der kontrahierenden Produktionen oft sehr störend; deswegen haben wir ihn aus der Definition herausgenommen. Ferner ergibt sich, wenn man kontrahierende Regeln ausschließt, sofort, daß die Sprachen eine Hierarchie bilden, da eine Grammatik vom Typ j auch eine Grammatik von Typ i ist mit $i \leq j$.

Um über Äquivalenz von Grammatiken zu sprechen, muß man häufig über Ableitungen sprechen. Wir wollen dazu die folgende Begriffsbildung vorschlagen. Es sei G eine Grammatik. Eine **Ableitung der Länge** n ist eine Folge $\langle A_i : i < n\rangle$ von Anwendungen von Regeln aus G. Mit $\mathbf{der}(G, \vec{\alpha})$ bezeichnen wir die Menge der Ableitungen in G aus der Zeichenkette α und $\mathbf{der}(G) := \mathbf{der}(G, S)$. Der Begriff der Anwendung einer Regel ist noch zu präzisieren. Sei $\rho = \vec{\gamma} \to \vec{\delta}$. Wir nennen ein Tripel $A = \langle \vec{\alpha}_i, C, \vec{\alpha}_{i+1}\rangle$ eine **Anwendung von** ρ, wenn C ein Vorkommen von $\vec{\gamma}$ in $\vec{\alpha}_i$ ist und gleichzeitig ein Vorkommen von $\vec{\delta}$ in $\vec{\alpha}_{i+1}$. Dies bedeutet im Einzelnen, daß $\vec{\kappa}_1$ und $\vec{\kappa}_2$ existieren mit $C = \langle \vec{\kappa}_1, \vec{\kappa}_2\rangle$ und $\vec{\alpha}_i = \vec{\kappa}_1 \cdot \vec{\gamma} \cdot \vec{\kappa}_2$ und $\vec{\alpha}_{i+1} = \vec{\kappa}_1 \cdot \vec{\delta} \cdot \vec{\kappa}_2$. Wir nennen C den **Bereich** von A.

Diese Definitionen sind mit einigem Bedacht gewählt worden. Sei $\langle A_i : i < n\rangle$, $A_i = \langle \vec{\alpha}_i, C_i, \vec{\alpha}_{i+1}\rangle$ eine Ableitung in G. Dann heißt $\langle \vec{\alpha}_i : i < n + 1\rangle$ die zugehörige **Kettenfolge**. Man beachte, daß die Kettenfolge um ein Glied länger ist als die Ableitung. Im Folgenden werden wir Kettenfolgen auch schlicht als Ableitung bezeichnen. Aus der Kettenfolge läßt sich jedoch im Allgemeinen die Ableitung nicht rekonstruieren. Ein Beispiel mag dies verdeutlichen. Es sei G folgende Grammatik.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{AB} \\ \mathtt{A} & \to & \mathtt{AA} \\ \mathtt{B} & \to & \mathtt{AB} \end{array}$$

Die Kettenfolge $\langle \mathtt{S}, \mathtt{AB}, \mathtt{AAB}\rangle$ entspricht zwei verschiedenen Ableitungen:

$$\langle\langle \mathtt{S}, \langle \varepsilon, \varepsilon\rangle, \mathtt{AB}\rangle, \langle \mathtt{AB}, \langle \varepsilon, \mathtt{B}\rangle, \mathtt{AAB}\rangle\rangle$$
$$\langle\langle \mathtt{S}, \langle \varepsilon, \varepsilon\rangle, \mathtt{AB}\rangle, \langle \mathtt{AB}, \langle \mathtt{A}, \varepsilon\rangle, \mathtt{AAB}\rangle\rangle$$

Nach Anwendung von ρ wird zwar $\vec{\gamma}$ ersetzt, aber $\vec{\kappa}_1$ und $\vec{\kappa}_2$ lassen sich noch immer identifizieren. In diesem Sinne können wir im folgenden Lemma davon reden, daß der Bereich einer auf der Anwendung von ρ folgenden Regelanwendung von einer Regel σ disjunkt sei zu dem Bereich von ρ, nämlich wenn der Bereich von σ in $\vec{\kappa}_1$ oder $\vec{\kappa}_2$ enthalten ist.

Lemma 1.5.3 (Regelvertauschung) *Es sei $\langle \vec{\alpha}, \vec{\beta}\rangle$ eine Anwendung von ρ und $\langle \vec{\beta}, \vec{\gamma}\rangle$ eine Anwendung von σ. Es seien die Bereiche dieser Anwendungen disjunkt. Dann existiert eine Anwendung $\langle \vec{\alpha}, \vec{\delta}\rangle$ von σ sowie eine Anwendung $\langle \vec{\delta}, \vec{\gamma}\rangle$ von ρ, und beide Anwendungen haben disjunkte Bereiche.*

Der Beweis ist einfach und eine Übung. Wir geben ein Beispiel. Es seien die folgenden Regeln gegeben:

$$\begin{array}{lcllcl} \mathtt{AX} & \to & \mathtt{XA} & \mathtt{XA} & \to & \mathtt{Xa} \\ \mathtt{XB} & \to & \mathtt{Xb} & \mathtt{Xa} & \to & \mathtt{a} \end{array}$$

Auf die Zeichenkette AXB können nun zwei Regeln angewendet werden. Die erste hat den Bereich $\langle \varepsilon, \mathtt{B} \rangle$, die zweite den Bereich $\langle \mathtt{A}, \varepsilon \rangle$. Die Bereiche überlappen; und in der Tat zerstört die Anwendung der einen Regel den Bereich, auf den die zweite angewendet werden kann. Wenden wir die Regel $\mathtt{AX} \to \mathtt{XA}$ an, so können wir nicht zu einer terminalen Kette kommen:

$$\mathtt{AXB} \Rightarrow \mathtt{XAB} \Rightarrow \mathtt{XaB}$$

Wenden wir dagegen die Regel $\mathtt{XB} \to \mathtt{Xb}$ an, so bekommen wir:

$$\mathtt{AXB} \Rightarrow \mathtt{AXb} \Rightarrow \mathtt{XAb} \Rightarrow \mathtt{Xab} \Rightarrow \mathtt{ab}$$

Soweit zu einem Beispiel, wo die Anwendungen nicht vertauschen. Nun nehmen wir die Zeichenkette AXXB. Wiederum sind die beiden erwähnten Regeln in Konkurrenz; aber diesmal zerstört keine der Anwendungen den Bereich der anderen:

$$\mathtt{AXXB} \Rightarrow \mathtt{AXXb} \Rightarrow \mathtt{XAXb}$$

$$\mathtt{AXXB} \Rightarrow \mathtt{XAXB} \Rightarrow \mathtt{XAXb}$$

Daraus kann dann wie vorher die Zeichenkette ab abgeleitet werden. Man mache sich klar, daß zum Beispiel in einer kontextfreien Grammatik grundsätzlich jedes Paar konkurrierender Regelanwendungen auf verschiedenen Nichtterminalen vertauscht. Dies liegt einfach daran, daß die Bereiche dieser Regeln stets nur das einzelne Symbol umfaßt.

Wir wollen noch ein paar weitere Vereinfachungen vorschlagen. Zunächst kann man statt eines einzelnen Startsymbols auch eine Menge von Startsymbolen haben. Wir definieren eine Grammatik* als ein Quadrupel $G = \langle \Sigma, N, A, R \rangle$, wo alles ist wie oben, außer daß $\Sigma \subseteq N$. Dann ist $G \vdash \vec{\gamma}$ falls es ein $S \in \Sigma$ gibt, derart, daß $S \Rightarrow_R \vec{\gamma}$. Man kann die Grammatik* G wie folgt in eine Grammatik $G^{\heartsuit}$ umformen. Man nimmt ein $S_\Sigma \notin A \cup N$ zusammen mit Regeln $S_\Sigma \to X$, für alle $X \in \Sigma$. Es ist dann leicht zu zeigen, daß $L(G^{\heartsuit}) = L(G)$ ist.

In der Definition einer Grammatik wurde gefordert, daß die Menge der nichtterminalen Symbole sowie die Menge der terminalen Symbole disjunkt sind. Wir wollen uns überlegen, daß diese Bedingung keine Einschränkung bedeutet. Wir werden sie deshalb auch manchmal fallenlassen, was sowohl aus sachlichen Gründen wie auch aus Einfachheitsüberlegungen geboten sein kann.

Definition 1.5.4 *Eine **Quasi–Grammatik** ist ein Quadrupel $\langle \mathtt{S}, N, A, R \rangle$ bei dem A und N endliche nichtleere Mengen, $\mathtt{S} \in N$, und R ein Semi–Thue System über $N \cup A$ derart, daß wenn $\langle \vec{\alpha}, \vec{\beta} \rangle \in R$, so enthält α ein Symbol aus N.*

Proposition 1.5.5 *Zu jeder Quasi–Grammatik existiert eine Grammatik, welche die gleiche Sprache erzeugt.*

Beweis. Sei $\langle \mathtt{S}, N, A, R\rangle$ eine Quasi–Grammatik. Setze $N_1 := N \cap A$. Dann sei mit jedem $a \in N_1$ $\mathtt{Y}_a$ ein neues Symbol. Wir setzen $Y := \{\mathtt{Y}_a : a \in N_1\}$, $N^\circ := (N - N_1) \cup Y$, $A^\circ := A$. Jetzt ist $N^\circ \cap A^\circ = \varnothing$. Wir setzen $\mathtt{S}^\circ := \mathtt{S}$, falls $\mathtt{S} \notin A$ und $\mathtt{S}^\circ := \mathtt{Y}_\mathtt{S}$, falls $\mathtt{S} \in A$. Als letztes definieren wir die Regeln. Es sei $\vec{\alpha}^\circ$ das Ergebnis der Ersetzung von jedem $a \in N_1$ durch das entsprechende $\mathtt{Y}_a$. Dann ist

$$R^\circ := \{\vec{\alpha}^\circ \to \vec{\beta}^\circ : \vec{\alpha} \to \vec{\beta} \in R\} \cup \{\mathtt{Y}_a \to a : a \in N_1\}$$

Mit $G^\circ := \langle \mathtt{S}^\circ, N^\circ, A^\circ, R^\circ\rangle$ gilt jetzt $L(G^\circ) = L(G)$. Die Begründung ist wie folgt. Wir definieren einen Homomorphismus $h : (A \cup N)^* \to (A^\circ \cup N^\circ)^*$ durch $h(a) := a$ für $a \in A - N_1$, $h(a) := \mathtt{Y}_a$ für $a \in N_1$ und $h(X) := X$ für alle $X \in N - N_1$. Dann gilt $h(S) = S^\circ$, sowie $h(R) \subseteq R^\circ$. Daraus folgt unmittelbar, daß wenn $G \vdash \vec{\alpha}$, dann auch $G^\circ \vdash h(\vec{\alpha})$. Dies weist man durch Induktion über die Länge der Ableitung nach. Da nun in G° $\vec{\alpha}$ aus $h(\vec{\alpha})$ herleitbar ist, so gilt sicherlich $L(G) \subseteq L(G^\circ)$. Für die Umkehrung muß man sich überlegen, daß die Anwendung einer Regel $\mathtt{Y}_a \to a$ stets an das Ende der Ableitung gelegt werden kann. Denn wenn $\vec{\alpha} \to \vec{\beta}$ eine Regel ist, so ist sie entweder vom Typ $\mathtt{Y}_b \to b$ und ersetzt ein $\mathtt{Y}_b$ durch b; und sie vertauscht daher mit der Anwendung der ersten Regel. Oder sie ist von einer anderen Form, nämlich von der Form $\vec{\alpha}^\circ \to \vec{\beta}^\circ$; da a nicht in $\vec{\alpha}^\circ$ vorkommt, so vertauschen auch diese Anwendungen von Regeln. Ist dies gezeigt, so gilt mit $G^\circ \vdash \vec{\alpha}$ sofort $G^\circ \vdash \vec{\alpha}^\circ$. Letzteres hat $G \vdash \vec{\alpha}$ zur Folge. $\dashv$

Wir weisen darauf hin, daß auch die Bedingung, daß in einer Produktion die linke Seite ein Nichtterminalsymbol enthält, keine wirkliche Einschränkung darstellt. Denn sei $G = \langle \mathtt{S}, N, A, R\rangle$ eine Grammatik, bei der dies nicht so ist. Dann sei für jedes Terminalsymbol a a^1 ein neues Symbol, und $A^1 := \{a^1 : a \in A\}$. Schließlich sei für jede Regel $\rho = \vec{\alpha} \to \vec{\beta}$ die Regel ρ^1 das Ergebnis der Ersetzung von jedem Vorkommen eines $a \in A$ durch a^1 (auf beiden Seiten der Produktion). Nun setze man $\mathtt{S}' := \mathtt{S}$, falls $\mathtt{S} \notin A$, und $\mathtt{S}' := \mathtt{S}^1$ andernfalls, und $R' := \{\rho^1 : \rho \in R\} \cup \{a^1 \to a : a \in A\}$. Endlich sei

$$G' := \langle \mathtt{S}', N \cup A^1, A, R'\rangle$$

Es ist nicht schwer zu zeigen, daß $L(G') = L(G)$. Damit ist die Definition einer Grammatik um einiges einfacher geworden.

Als nächstes wollen wir noch eine Verallgemeinerung für kontextsensitive Sprachen beweisen. Eine Grammatik heiße **nichtkontrahierend**, falls entweder keine Regel kontrahierend ist oder alle Regeln nichtkontrahierend sind bis auf die Regel $S \to \varepsilon$, und in diesem Fall ist das Symbol S niemals auf der rechten Seite einer Produktion.

Theorem 1.5.6 *Genau dann ist eine Sprache kontextsensitiv, wenn sie durch eine nichtkontrahierende Grammatik erzeugt werden kann.*

Beweis. Gewiß sind kontextsensitive Grammatiken nichtkontrahierend. Sei also G nichtkontrahierend. Wir wollen ein $G^\spadesuit$ finden, welches kontextsensitiv ist und $L(G^\spadesuit) = L(G)$ erfüllt. Sei dazu $\rho : X_0X_1 \dots X_{m-1} \to Y_0Y_1 \dots Y_{n-1}$ eine Regel. (Wie oben gesehen, können wir uns auf solche Regeln und Regeln der Form $X \to a$ beschränken. Letztere sind aber gewiß nicht kontrahierend.) Es ist $m \leq n$. Wir nehmen uns m neue Symbole, Z_0, Z_1, …, Z_{m-1} und anstelle der alten Regel die folgenden:

$$\begin{array}{lcl}
X_0X_1 \dots X_{m-1} & \to & Z_0X_1 \dots X_{m-1} \\
Z_0X_1X_2 \dots X_{m-1} & \to & Z_0Z_1X_2 \dots X_{m-1} \\
 & \dots & \\
Z_0Z_1 \dots Z_{m-2}X_{m-1} & \to & Z_0Z_1 \dots Z_{m-1} \\
Z_0Z_1 \dots Z_{m-1} & \to & Y_0Z_1 \dots Z_{m-1} \\
Y_0Z_1Z_2 \dots Z_{m-1} & \to & Y_0Y_1Z_2 \dots Z_{m-1} \\
 & \dots & \\
Y_0Y_1 \dots Y_{m-2}Z_{m-1} & \to & Y_0Y_1 \dots Y_{n-1}
\end{array}$$

Die Menge dieser Regeln werde mit $\rho^\spadesuit$ bezeichnet. Es sei $G^\spadesuit$ das Resultat der Ersetzung aller nicht kontextsensitiven Regeln ρ durch $\rho^\spadesuit$. Gewiß sind alle diese Regeln kontextsensitiv. Nun sei eine Ableitung in G gegeben. Dann ersetze eine Anwendung der Regel ρ durch die angegeben Sequenz von Regeln in $\rho^\spadesuit$. Dies ergibt eine Ableitung der Zeichenkette in $G^\spadesuit$. Sei umgekehrt eine Ableitung in $G^\spadesuit$ gegeben. Man überlegt sich zunächst dies: wird irgendwo eine Regel aus $\rho^\spadesuit$ angwedent, und darauf eine Regel aus $\rho_1^\spadesuit$, so vertauschen diese, sofern nicht $\rho_1 = \rho$ und die zweite Regelanwendung gerade die auf die erste innerhalb $\rho^\spadesuit$ folgende ist. Dann ist nach geeigneter Umordnung der Beweis eine Folge von Segmenten, wo jedes Segment, welches eine Anwendung einer Regel aus $\rho^\spadesuit$ enthält, von der oben gegebenen Form ist, wobei es also mit $\vec{X}$ beginnt und mit $\vec{Y}$ endet. Es kann also durch ρ ersetzt werden. Man führe alle diese Ersetzungen durch. Dies ergibt eine Ableitung in G. ⊣

Der folgende Satz zeigt, daß die Sprachen vom Typ 1 nicht unter beliebigen Homomorphismen abgeschlossen sind (unter Benützung von Satz 1.7.8).

Theorem 1.5.7 *Es sei* $\mathtt{a}, \mathtt{b} \notin A$. *Zu jeder Sprache S über A vom Typ 0 existiert eine Sprache T über $A \cup \{\mathtt{a}, \mathtt{b}\}$ vom Typ 1 derart, daß zu jedem $\vec{x} \in S$ ein i existiert mit $\mathtt{a}^i\mathtt{b}\vec{x} \in T$ und jedes $\vec{y} \in T$ die Form $\mathtt{a}^i\mathtt{b}\vec{x}$ mit $\vec{x} \in S$.*

Beweis. Wir setzen $N^\clubsuit := N \cup \{\mathtt{A}, \mathtt{B}, \mathtt{S}^\clubsuit\}$. Es sei $\rho : X_0X_1 \dots X_{m-1} \to Y_0Y_1 \dots Y_{n-1}$ eine kontrahierende Regel. Dann sei $\rho^\clubsuit := X_0X_1 \dots X_{m-1} \to \mathtt{A}^{m-n}Y_0Y_1 \dots Y_{n-1}$. $\rho^\clubsuit$ ist gewiß nicht kontrahierend. Ist ρ nicht kontrahierend, so sei $\rho^\clubsuit := \rho$. Es bestehe

$R^{\clubsuit}$ aus allen Regeln $\rho^{\clubsuit}$ mit $\rho \in R$, sowie folgenden Regeln:

$$\begin{array}{lcll} \mathtt{S}^{\clubsuit} & \to & \mathtt{BS}\,, & \\ X\mathtt{A} & \to & \mathtt{A}X, & X \in N^{\clubsuit}, \\ \mathtt{BA} & \to & \mathtt{aB}\,, & \\ \mathtt{B} & \to & \mathtt{b}\,. & \end{array}$$

Es sei $T := L(G^{\clubsuit})$. Gewiß ist $\vec{y} \in T$ nur dann, wenn $\vec{y} = \mathtt{a}^i\mathtt{b}\vec{x}$ für ein $\vec{x} \in A^*$. Zunächst enthalten Worte nur einmal $\mathtt{B}$ (oder nur einmal $\mathtt{b}$). Weiter kann man $\mathtt{A}$ in $\mathtt{a}$ nur dann umwandeln, wenn es direkt vor $\mathtt{B}$ steht. Danach ist $\mathtt{B}$ hinter $\mathtt{a}$. Folglich muß $\mathtt{b}$ hinter allen $\mathtt{a}$ sein, aber vor allen anderen Symbolen, da man $\mathtt{B}$ nicht vor diese bewegen kann. Nun betrachte die Abbildung v definiert durch $v : \mathtt{A}, \mathtt{a}, \mathtt{B}, \mathtt{b}, \mathtt{S}^{\clubsuit} \mapsto \varepsilon$ und $v : X \mapsto X$ für $X \in N$, $a \mapsto a$ für $a \in A$. Ist $\langle \vec{\alpha}_i : i < n \rangle$ eine Ableitung in $G^{\clubsuit}$, so ist $\langle \overline{v}(\vec{\alpha}_i) : 0 < i < n \rangle$ eine Ableitung in G (von Wiederholungen abgesehen). So zeigt man, daß aus $\mathtt{a}^i\mathtt{b}\vec{x} \in T$ folgt $\vec{x} \in L(G)$. Nun sei $\vec{x} \in L(G)$. Es sei $\langle \vec{\alpha}_i : i < n \rangle$ eine Ableitung von $\vec{x}$ in G. Dann verfahre man so. Es sei $\vec{\beta}_0 := S^{\clubsuit}$ und $\vec{\beta}_1 = \mathtt{BS}$. Ferner sei $\vec{\beta}_{i+1}$ von der Form $\mathtt{BA}^{k_i}\vec{\alpha}_i$ für gewisses k_i, welches man induktiv berechnet. Es ist leicht zu sehen, daß $\vec{\beta}_{i+1} \vdash_{G^{\clubsuit}} \vec{\beta}_{i+2}$, sodaß man die Folge $\langle \vec{\beta}_i : i < n+1 \rangle$ leicht zu einer Ableitung auffüllen kann. Aus $\mathtt{BA}^{k_n}\vec{x}$ läßt sich nun $\mathtt{a}^{k_n}\mathtt{b}\vec{x}$ herleiten. Dies zeigt, daß $\mathtt{a}^{k_n}\mathtt{b}\vec{x} \in T$. $\dashv$

Es sei $v : A \to B^*$ eine Abbildung. v (sowie der Homomorphismus $\overline{v}$) heißt ε–**frei**, falls $v(a) \neq \varepsilon$ für alle $a \in A$. Wir können nun Folgendes feststellen.

Theorem 1.5.8 *Es seien L_1 und L_2 Sprachen vom Typ i, $0 \leq i \leq 3$. Dann sind auch folgende Sprachen vom Typ i:*

1. $L_1 \cup L_2$
2. $L_1 \cdot L_2$
3. L_1^*
4. $\overline{v}[L_1]$, *wobei v ε–frei ist.*

Ist $i \neq 1$, so ist $\overline{v}[L_1]$ vom Typ i auch wenn v nicht ε–frei ist.

Beweis. Die erste Behauptung ist bereits gezeigt. Zu der zweiten bemerken wir, daß wir eine neues Startsymbol $\mathtt{S}^{\times}$ einführen können, mit den Regeln $\mathtt{S}^{\times} \to \mathtt{S}_1\mathtt{S}_2$, wo $\mathtt{S}_1$ das Startsymbol von G_1 und G_2 das Startsymbol von G_2 ist. Dies ergibt eine Grammatik vom Typ i, außer wenn $i = 3$. In diesem Falle ergibt sich die Tatsache aus den Ergebnissen von Abschnitt 2.1. Es ist jedoch nicht schwer eine Grammatik zu konstruieren, welche regulär ist und $L_1 \cdot L_2$ erzeugt. Nun zu L_1^*. Sei $\mathtt{S}$ das Startsymbol

für eine reguläre Grammatik G, welche L_1 erzeugt. Hier führe man ein neues Symbol $\mathtt{S}^+$ sowie ein neues Startsymbol $\mathtt{S}^*$ ein und Regeln

$$\begin{array}{lcl} \mathtt{S}^* & \to & \varepsilon \mid \mathtt{S} \mid \mathtt{S}\mathtt{S}^+, \\ \mathtt{S}^+ & \to & \mathtt{S} \mid \mathtt{S}\mathtt{S}^+ \end{array}$$

Diese Grammatik ist vom Typ i und erzeugt L_1^*. (Wiederum ist $i = 3$ eine Ausnahme, die leicht zu beheben ist.) Als letztes sei v in Homomorphismus. Dann ersetzen wir alle Regeln $\rho : \vec{\alpha} \to \vec{\beta}$ durch Regeln $\overline{v}(\rho) : \overline{v}(\vec{\alpha}) \to \overline{v}(\vec{\beta})$. (Wobei wir v fortsetzen, indem wir $v(X) := X$ für jedes Nichtterminalsymbol X setzen.) Ist $i = 0, 2$, so bleibt der Typ erhalten. Ist $i = 1$, so muß man verlangen, daß v ε–frei ist. Denn wenn $\vec{\gamma}X\vec{\delta} \to \vec{\gamma}\vec{\alpha}\vec{\delta}$ eine Regel ist, und $\vec{\alpha}$ eine terminale Zeichenkette, so kann durchaus $\overline{v}(\alpha) = \varepsilon$ sein. Dies ist jedoch niemals so, wenn v ε–frei ist. Ist $i = 3$, so muß man wiederum extra Vorkehrungen treffen. Denn nun haben wir Regeln der Form $X \to \vec{x}Y$ und $X \to \vec{x}$, $\vec{x} = x_0x_1 \dots x_{n-1}$. Diese sind durch Regeln der Form $X \to x_0Z_0$, $Z_i \to x_iZ_{i+1}$ und $Z_{n-2} \to x_{n-1}Y$ beziehungsweise $Z_{n-2} \to x_{n-1}$ zu ersetzen. $\dashv$

Definition 1.5.9 *Es sei A eine Menge. Eine nichtleere Menge $\mathcal{S} \subseteq \wp(A^*)$ heißt* ***abstrakte Sprachfamilie über*** *A, falls Folgendes gilt.*

1. *Für jedes $S \in \mathcal{S}$ existiert ein endliches $B \subseteq A$ mit $S \subseteq B^*$.*
2. *Ist $h : A^* \to A^*$ ein Homomorphismus und $S \in \mathcal{S}$, so ist auch $h[S] \in \mathcal{S}$.*
3. *Ist $h : A^* \to A^*$ ein Homomorphismus und $S \in \mathcal{S}$, $B \subseteq A$ endlich, so ist auch $h^{-1}[S] \cap B^* \in \mathcal{S}$.*
4. *Ist $S \in \mathcal{S}$ und R eine reguläre Sprache, so ist $S \cap R \in \mathcal{S}$.*
5. *Ist $S_1, S_2 \in \mathcal{S}$, so auch $S_1 \cup S_2 \in \mathcal{S}$ und $S_1 \cdot S_2 \in \mathcal{S}$.*

Die regulären Sprachen, die kontextfreien Sprachen und die Typ 0 Sprachen bilden abstrakte Sprachfamilien. Die kontextsensitiven Sprachen erfüllen alle Kriterien, nur sind sie lediglich unter ε–freien Homomorphismen abgeschlossen. Man zeigt im Übrigen leicht, daß die regulären Sprachen über A die kleinste abstrakte Sprachfamilie bilden. Mehr zu diesem Thema findet man in Ginsburg [18].

Übung 12. Es sei T ein Semi–Thue–System über A und $A \subseteq B$. Es sei $\Rightarrow_T^* \subseteq A^* \times A^*$ bekannt ist. Man fasse T als Semi–Thue System T' über B auf und charakterisiere $\Rightarrow_{T'}^*$ mittels $\Rightarrow_T^*$. *Bemerkung.* Diese Übung zeigt, daß man streng genommen zu T auch noch das Alphabet benennen muß, über welches der Semi–Thue–Prozeß definiert ist, da ja nicht alle Symbole in T auftreten müssen.

Übung 13. Beweisen Sie das Regelvertauschungslemma.

Übung 14. Zeigen Sie, daß jede endliche Sprache regulär ist.

Übung 15. Es sei G eine Grammatik mit Regeln der Form $X \to \vec{\alpha}$. Zeigen Sie, $L(G)$ ist kontextfrei. Desgleichen zeige man, daß $L(G)$ regulär ist, wenn alle Regeln die Form $X \to \alpha_0 \cdot \alpha_1$ haben, wo $\alpha_0 \in A \cup \{\varepsilon\}$ und $\alpha_1 \in N \cup \{\varepsilon\}$.

Übung 16. Es sei G eine Grammatik, in der jede Regel verschieden von $X \to a$ strikt expandierend ist. Zeigen Sie, daß eine Ableitung eines Wortes der Länge n höchstens $2n$ Schritte benötigt.

Übung 17. Zeigen Sie, daß die Sprache $\{\mathtt{a}^n\mathtt{b}^n : n \in \omega\}$ kontextfrei ist aber nicht regulär.

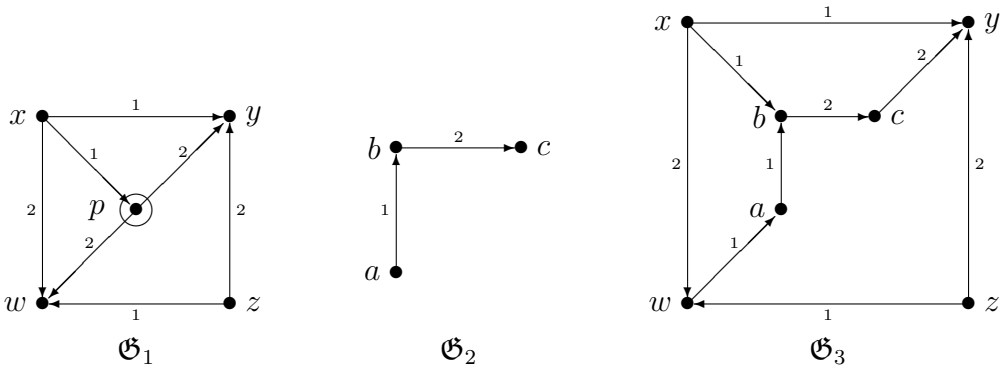
Übung 18. Schreiben Sie eine Typ 1 Grammatik für die Sprachen $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \in \omega\}$ und eine für $\{\vec{x} \cdot \vec{x} : \vec{x} \in A^*\}$.

1.6 Grammatik und Strukturbeschreibung

Im Unterschied zur bloßen Termersetzung kann man auch *Strukturersetzung* betrachten. Dabei wird analog zum Termfall ein System definiert, das schrittweise Strukturen durch Strukturen ersetzt. Dadurch erzeugt man direkt einen Strukturbaum. Man kann allerdings auch die Ersetzungssysteme selbst auffassen als — allerdings indirekte — Systeme zur Erzeugung von Strukturen. Dabei wollen wir als Strukturen in der Regel dat–Graphen mit Wurzel nehmen. Wir wollen beide Wege vorstellen.

Zunächst einmal wollen wir das Konzept eines Graphen erweitern. Wir betrachten jetzt statt Graphen sogenannte *Multigraphen*. Ein **gerichteter Multigraph** ist eine Struktur $\langle E, \langle K_i : i < n\rangle\rangle$, wo E eine Menge ist, die Menge der **Ecken**, und $K_i \subseteq E \times E$ eine Menge, die Menge der **Kanten** des Typs i. Kanten sind in unserem Fall stets gerichtet; deswegen werden wir das in Zukunft nicht besonders erwähnen. Geordnete Bäume sind nur ein Beispiel unter vielen von einem Multigraphen. Wir haben im Gegensatz zu Bäumen nunmehr zwei binäre Relationen. Aus technischen Gründen erlauben wir, daß $\langle \varnothing, \langle \varnothing : i < n\rangle\rangle$ auch ein Multigraph ist. Wir führen nun auf der Menge der Ecken eine Färbung ein. Dies ist eine Funktion $\mu_E : E \to F_E$, wo F_E eine nichtleere Menge ist, die Menge der **Eckenfarben**. Zum Beispiel ist die Menge der Kantentypen eine Teilmenge von F_E, aber Gleichheit muß nicht gelten, da wir unter Umständen einen großen Vorrat an Hilfssymbolen brauchen und deswegen gewisse Farben nicht verwendet werden. Genauso werden wir Kanten färben. Hier gibt es allerdings einen Unterschied. Zunächst einmal haben wir bereits eine Einteilung der Kantenmenge in n möglicherweise verschiedene Teilmengen. Wir können also einer Kante aus K_i jeweils die Zahl i als Farbe zuordnen. Allerdings kann eine gegebene Kante in verschiedenen K_i auftreten. Daher wollen

Abbildung 1.3: Beispiel einer Graph–Ersetzung



wir es anders einrichten. Wir bestimmen eine Menge F_K von **Kantenfarben** (etwa die Menge $n = \{0, 1, \ldots, n-1\}$) und betrachten einen Multigraphen als einen Graphen mit Kantenfärbung auf der Kantenmenge $K = \bigcup_{i \in F_K} K_i$ setzen, wobei $\mu_K : K \to \wp(F_K)$ definiert ist durch

$$\mu_K(\langle x, y \rangle) = \{i : \langle x, y \rangle \in K_i\}$$

Man beachte, daß $\mu_K(\langle x, y \rangle) \neq \varnothing$. Wir können auch $K = E \times E$ setzen, aber dann ist nicht notwendigerweise $\mu_K(\langle x, y \rangle) \neq \varnothing$. Ist $f \in F_K$ eine Farbe, so bezeichen wir mit $\mu_K(f)$ auch die Menge derjenigen Kanten, welche die Farbe f tragen.

Definition 1.6.1 *Ein $\langle F_E, F_K \rangle$–**gefärbter Multigraph** oder schlicht γ–**Graph** (über F_E und F_K) ist ein Tripel $\langle E, \mu_E, \mu_K \rangle$, wo E eine (möglicherweise leere) Menge und $\mu_E : E \to F_E$ sowie $\mu_K : E \times E \to \wp(F_K)$ Funktionen sind.*

Analog zum Termersetzungsmodell wollen wir terminale und nichtterminale Farben unterscheiden. Die Ersetzung verläuft stets so, daß eine Ecke ausgewählt wird und durch einen Graphen ersetzt wird. Da die Ecke in den Graphen eingebettet ist, muß man angeben, in welcher Weise der ersetzende Graph in die umgebende Struktur eingebettet sein soll. Dies macht die Definition um einiges aufwendiger. Bevor wir mit der Definition einer Grammatik beginnen, wollen wir klären, wie man eine Graphersetzung vornimmt. Zum besseren Verständnis geben wir dem Leser die Abbildung 1.3 an die Hand. Der Graph $\mathfrak{G}_3$ ist das Resultat der Ersetzung in $\mathfrak{G}_1$ des eingekreisten Punktes durch $\mathfrak{G}_2$. Die Farben sind hier $\{1, 2\}$. Dazu eine Vorüberlegung.

Es sei $\mathfrak{G} = \langle E, \mu_E, \mu_K \rangle$ ein γ–Graph. Ferner seien M_1 und M_2 Teilmengen von E mit $M_1 \cap M_2 = \varnothing$ und $M_1 \cup M_2 = E$. Dann definieren M_1 und M_2 Teilgraphen $\mathfrak{M}_i = \langle M_i, \mu_E^i, \mu_K^i \rangle$, nämlich durch $\mu_E^i := \mu_E \restriction M_i$ und $\mu_K^i := \mu_K \restriction M_i \times M_i$.

Diese Teilgraphen bestimmen $\mathfrak{G}$ nicht vollständig, da sie keinerlei Information über die Kanten zwischen M_1 und M_2 bereitstellen. Wir definieren deshalb Funktionen $in, out : M_2 \times F_K \to \wp(M_1)$, welche zu jeder Ecke in M_2 und jeder Kantenfarbe die Menge aller Ecken in M_1 benennt, welche eine in M_1 ein– bzw. auslaufende Kante der gewählten Farbe mit der gewählten Ecke haben.

$$\begin{array}{lcl} in(x,f) & := & \{y \in M_1 : f \in \mu_K(\langle y, x\rangle)\} \\ out(x,f) & := & \{y \in M_1 : f \in \mu_K(\langle x, y\rangle)\} \end{array}$$

Es ist klar, daß $\mathfrak{M}_1$, $\mathfrak{M}_2$, sowie die Funktionen in und out den Graphen $\mathfrak{G}$ eindeutig bestimmen. In unserem Beispiel ist

$$\begin{array}{lcllcl} in(p,1) & = & \{x\}, & in(p,2) & = & \varnothing \\ out(p,1) & = & \varnothing, & out(p,2) & = & \{w,y\} \end{array}$$

Nehmen wir nun an, wir wollen $\mathfrak{M}_2$ durch einen anderen Graphen $\mathfrak{H}$ ersetzen. Dazu müssen wir nicht nur $\mathfrak{H}$ haben, sondern auch Funktionen $in, out : H \times F_K \to \wp(M_1)$. Dies ist allerdings nicht der Weg, den wir hier gehen werden. Wir wollen nämlich Ersetzungsregeln formulieren, die so allgemein sind, daß sie nichts über die Form des Kontextes voraussetzen, in dem der zu ersetzende Graph eingebettet ist. Dazu wollen wir annehmen, daß die Mengen $in(x,f)$ und $out(x,f)$, $x \in H$, in systematischer Weise aus den Mengen $in(y,g)$, $out(y,g)$, $y \in M_2$, hervorgehen. Wir erlauben daher nur Angaben, wie die ersteren Mengen aus den letzteren hervorgehen. Wir definieren dazu vier sogenannte *Farbfunktionale*. Ein **Farbfunktional** von $\mathfrak{H}$ nach $\mathfrak{M}_2$ ist eine Abbildung

$$\mathfrak{F} : H \times F_K \to \wp(M_2 \times F_K)$$

In unserem Beispiel ist ein Funktional eine Funktion von $\{a,b,c\} \times \{1,2\}$ nach $\wp(\{p\} \times \{1,2\})$. Wir können dies vereinfachen zu Funktionen von $\{a,b,c\} \times \{1,2\}$ nach $\wp(\{1,2\})$. Die Farbfunktionale heißen $\mathfrak{II}$, $\mathfrak{IO}$, $\mathfrak{OI}$ und $\mathfrak{OO}$. Diese ordnen wir in eine 2×2–Matrix:

$$\mathfrak{F} := \begin{pmatrix} \mathfrak{II} & \mathfrak{IO} \\ \mathfrak{OI} & \mathfrak{OO} \end{pmatrix}$$

Dies ergibt für das Beispiel von Figur 1.3 das Folgende (wir notieren nur, wo die Funktion nicht $\varnothing$ ergibt).

$$\begin{array}{lcl} \mathfrak{II} & : & \langle b,1\rangle \mapsto 1, \\ \mathfrak{IO} & : & \\ \mathfrak{OI} & : & \langle a,1\rangle \mapsto 2, \\ \mathfrak{OO} & : & \langle c,2\rangle \mapsto 2. \end{array}$$

Das Ergebnis der Substitution von $\mathfrak{M}_2$ durch $\mathfrak{H}$ unter Benutzung der Farbfunktionale aus $\mathfrak{F}$ wird notiert durch $\mathfrak{G}[\mathfrak{H}/\mathfrak{M}_2 : \mathfrak{F}]$. Dieser Graph ist die Vereinigung von $\mathfrak{M}_1$ und $\mathfrak{H}$ mit den Funktionen in^+, out^+, die wie folgt definiert sind.

$$\begin{array}{lcl} in^+(x,f) & := & \bigcup\langle in(y,g) : \langle y,g\rangle \in \mathfrak{II}(x,f)\rangle \cup \bigcup\langle out(y,g) : \langle y,g\rangle \in \mathfrak{OI}(x,f)\rangle \\ out^+(x,f) & := & \bigcup\langle out(y,g) : \langle y,g\rangle \in \mathfrak{OO}(x,f)\rangle \cup \bigcup\langle in(y,g) : \langle y,g\rangle \in \mathfrak{IO}(x,f)\rangle \end{array}$$

Falls $\langle y, g\rangle \in \mathfrak{II}(x, f)$, so sagen wir, daß die mit der Farbe g in y einlaufende Kante als einlaufende Kante der Farbe f an x weitergereicht wird. Falls $\langle y, g\rangle \in \mathfrak{OI}(x, f)$, so sagen wir, daß die mit der Farbe g aus y auslaufende Kante an x als einlaufende Kante der Farbe f weitergereicht wird. Analog für $\mathfrak{IO}$ und $\mathfrak{OO}$. Es ist also möglich, daß die Kanten ihre Richtung wechseln, wenn sie weitergereicht werden. Falls dies nicht der Fall ist, benötigen wir nur die Funktionale $\mathfrak{II}$, und $\mathfrak{OO}$, die wir dann auch schlicht mit $\mathfrak{I}$ und $\mathfrak{O}$ bezeichnen. Wir betrachten nun den Spezialfall, wo M_2 aus einem einzigen Element, x, besteht. In diesem Fall ist ein Farbfunktional schlicht eine Funktion $\mathfrak{F} : H \times F_K \to \wp(F_K)$.

Definition 1.6.2 *Eine **kontextfreie Graph–Grammatik mit Ecken–Ersetzung** — kurz eine **kontextfreie** γ**–Grammatik** genannt — ist ein Quintupel* $\Gamma = \langle \mathfrak{S}, F_E, F_E^T, F_K, R\rangle$, *bei dem F_E eine nichtleere, endliche Menge von Eckenfarben, F_K eine nichtleere Menge von Kantenfarben, $F_E^T \subseteq F_E$ eine Menge von **terminalen Eckenfarben**, $\mathfrak{S}$ ein γ–Graph über F_E und F_K, der sogenannte **Startgraph**, und schließlich R eine endliche Menge von Tripeln $\langle X, \mathfrak{H}, \mathfrak{F}\rangle$ derart, daß $X \in F_E - F_E^T$ eine nichtterminale Eckenfarbe ist, $\mathfrak{H}$ ein γ–Graph über F_E und F_K ist, und $\mathfrak{F}$ eine Matrix von Farbfunktionalen.*

Eine Ableitung in einer γ–Grammatik Γ ist wie folgt. Wir definieren auf γ–Graphen $\mathfrak{G}$ und $\mathfrak{H}$ über den Farben F_E und F_K $\mathfrak{G} \Rightarrow_R^1 \mathfrak{H}$, falls es ein $\langle X, \mathfrak{M}, \mathbb{F}\rangle \in R$ gibt, dergestalt, daß $\mathfrak{H} = \mathfrak{G}[\mathfrak{M}/F_E : \mathbb{F}]$ wo F_E ein Teilgraph ist bestehend aus nur einer Ecke x, welche die Farbe Farbe X hat. Weiter definieren wir $\Rightarrow_R$ als die reflexiv, transitive Hülle von $\Rightarrow_R^1$ und schließlich sei $\Gamma \vdash \mathfrak{G}$, falls $\mathfrak{S} \Rightarrow_R \mathfrak{G}$. Eine Ableitung ist beendet, wenn es keine Ecke mit nichtterminaler Farbe gibt. Wir schreiben $L_\gamma(\Gamma)$ für die von Γ erzeugte Menge von γ–Graphen. Man beachte, daß auf der Kantenmenge keinerlei Unterschied besteht zwischen terminalen und nichtterminalen Farben. Die Ableitung wird lediglich durch die Eckenfarben gesteuert.

Analog zu Zeichenkettengrammatiken definieren wir die Produktivität einer Regel als die Differenz zwischen der Kardinalität der ersetzenden Graphen und der Kardinalität des ersetzten Graphen. Letztere ist hier immer gleich 1. Daher ist die Produktivität stets ≥ -1. Sie ist immer dann $= -1$, wenn der ersetzende Graph der leere Graph ist. In einer Übungsaufgabe soll nachgewiesen werden, daß solche Regeln stets eliminierbar sind. Eine Regel hat die Produktivität $= 0$, falls der ersetzende Graph auch wieder aus einem Punkt besteht. Daß auch solche Regeln verzichtbar sind, soll in den Übungen ebenfalls nachgewiesen werden.

Wir geben im Folgenden zwei besonders wichtige Typen von γ–Grammatiken an. Beide sind kontextfrei als γ–Grammatiken, die zweite erzeugt aber nachweislich auch nicht–kontextfreie Sprachen. Dies zeigt, daß das Konzept der γ–Grammatiken allgemeiner ist. Wir beginnen aber mit kontextfreien Grammatiken. Man kann diese entweder als Kettenersetzungsgrammatiken auffassen, oder aber als Grammatiken,

welche Bäume schrittweise erzeugen. Damit letzteres gelingt, nehmen wir hier an, es gebe keine Regeln der Form $X \to \varepsilon$. (Denn diese erzeugen Bäume, deren Marken auf den Blättern nicht notwendig aus A sind. Dieser Sonderfall kann behandelt werden, indem wir ε als Marke zulassen, wollen dies aber im Folgenden nicht tun, um die Darstellung einfacher zu gestalten.) Sei $G = \langle S, A, N, R\rangle$ eine solche Grammatik. Wir setzen $F_E := A \cup (N \times 2)$. Wir schreiben X^0 für $\langle X, 0\rangle$ und X^1 für $\langle X, 1\rangle$. $F_E^T = A \cup N \times \{0\}$. $F_K := \{<, \sqsubset\}$. Ferner besteht der Startgraph aus nur einer Ecke, mit Farbe S^1, und keiner Kante. Die Ersetzungsregeln sind wie folgt. Sei $\rho = X \to \alpha_0\alpha_1 \ldots \alpha_{n-1}$ eine Regel aus G, keines der α_i ist ε. Dann definieren wir einen γ–Graphen $\mathfrak{H}_\rho$ wie folgt. $H_\rho := \{y_i : i < n\} \cup \{x\}$. $\mu_E(x) = X^0$, $\mu_E(y_i) = \alpha_i$, falls $\alpha_i \in A$ und $\mu_E(y_i) = \alpha_i^1$, falls $\alpha_i \in N$. Wir bezeichnen mit $\mu_K(f)$, $f \in F_K$, auch die Menge der Kanten, welche die Farbe f tragen. Damit ist.

$$\begin{array}{lcl} \mu_K(<) & := & \{\langle y_i, x\rangle : i < n\} \\ \mu_K(\sqsubset) & := & \{\langle y_i, y_j\rangle : i < j < n\} \end{array}$$

Dies definiert $\mathfrak{H}_\rho$. Nun legen wir die Farbfunktionale fest. Für alle $u \in n$ ist

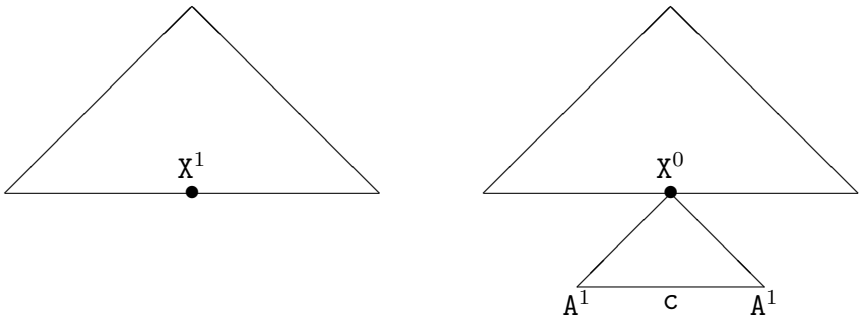
$$\begin{array}{lcllcl} \mathfrak{I}_\rho(u, \sqsubset) & := & \{\sqsubset\} & \mathfrak{O}_\rho(u, \sqsubset) & := & \{\sqsubset\} \\ \mathfrak{I}_\rho(u, <) & := & \{<\} & \mathfrak{O}_\rho(u, <) & := & \{<\} \end{array}$$

Schließlich setzen wir $\rho^\gamma := \langle X, \mathfrak{H}_\rho, \{\mathfrak{I}_\rho, \mathfrak{O}_\rho\}\rangle$. $R^\gamma := \{\rho^\gamma : \rho \in R\}$.

$$\gamma G := \langle \mathfrak{S}, F_E, F_E^T, F_T, R^\gamma\rangle$$

Wir werden zeigen, daß diese Grammatik tatsächlich genau diejenigen Bäume erzeugt, welche wir mit der Grammatik G assoziieren. Bevor wir dies tun, einige Bemerkungen. Die ursprünglichen Nichtterminalsymbole sind jetzt, technisch gesehen, auch Terminalsymbole, d. h. terminale Farben, denn sie sind ja Bestandteil der zu erzeugenden Struktur. Damit wir überhaupt nichttriviale Ableitungen bekommen, müssen wir bei den Nichtterminalsymbolen von G unterscheiden zwischen solchen, bei welchen die Strukturersetzung noch zu geschehen hat, und solchen, bei denen sie schon erfolgt ist. Definitionsgemäß sind die ersteren diejenigen, die Blätter im Baum sind. Die Ableitung ist also genau dann nicht beendet, wenn der Baum Blätter mit G–Nichtterminalen trägt. An diesen — und nur an diesen — darf man weiterarbeiten. Dieser Effekt wird erreicht, indem wir bei G–Nichtterminalen eine Unterscheidung einführen in **aktive** (Superskript 1) und **passive** (Superskript 0). Eine Anwendung einer Ersetzung an einem aktiven Knoten besteht in der Ersetzung des Knotens durch einen Baum, dessen Blätter aktiv sind, nicht jedoch die Wurzel. Ein Ableitungsschritt wird in Figur 1.4 gezeigt. Darin wird die Regel $\texttt{X} \to \texttt{AcA}$ auf den Baum auf der linken Seite angewendet. Das Ergebnis ist rechts zu sehen. Man kann leicht zeigen, daß in jeder Ableitung nur Blätter aktiv sind. Dies ist aber genau die Art und Weise, in welcher eine kontextfreie Grammatik Bäume erzeugt. Wir setzen dann $L_B(G)$ gerade die Menge der von γG erzeugten Bäume unter Identifikation von den Marken X^0 mit X. Die Regeln von G können dann als Bedingungen

Abbildung 1.4: Ersetzung in einer kontextfreien Grammatik



an Bäume wie folgt uminterpretiert werden. Es heiße ein Teilbaum $\mathfrak{C}$ von $\mathfrak{B}$ **lokaler Teilbaum**, falls (i) er die Höhe 1 hat, das heißt keine inneren Knoten besitzt und (ii) er zusammen mit der Wurzel sämtliche Töchter dieser Wurzel in $\mathfrak{B}$ enthält. Zu einer Regel $\rho = X \to Y_0 Y_1 \dots Y_{n-1}$ definieren wir $L_\rho := \{y_i : i < n\} \cup \{x\}$, $<_\rho := \{\langle y_i, x\rangle : i < n\}$, $\sqsubset_\rho := \{\langle y_i, y_j\rangle : i < j < n\}$, und schließlich $\ell_\rho(x) := X$, $\ell(y_i) := Y_i$. $\mathfrak{L}_\rho := \langle L_\rho, <_\rho, \sqsubset_\rho, \ell_\rho\rangle$. Dabei ist ein **Isomorphismus** zwischen geordneten Bäumen $\mathfrak{B} = \langle B, <_\mathfrak{B}, \sqsubset_\mathfrak{B}, \ell_\mathfrak{B}\rangle$, $\mathfrak{C} = \langle C, <_\mathfrak{C}, \sqsubset_\mathfrak{C}, \ell_\mathfrak{C}\rangle$ mit Marken in M eine bijektive Abbildung $h : B \to C$ derart, daß $h(<_\mathfrak{B}) = <_\mathfrak{C}$, $h(\sqsubset_\mathfrak{B}) = h(\sqsubset_\mathfrak{C})$ und $\ell_\mathfrak{C}(h(x)) = \ell_\mathfrak{B}(x)$ für alle $x \in B$.

Proposition 1.6.3 *Sei $G = \langle S, N, A, R\rangle$. $\mathfrak{B} \in L_B(G)$ genau dann, wenn jeder lokale Teilbaum von $\mathfrak{B}$ isomorph ist zu einem $\mathfrak{L}_\rho$ mit $\rho \in R$.*

Theorem 1.6.4 *Es sei $\boldsymbol{B}$ eine Menge von Bäumen über dem Alphabet $A \cup N$. Genau dann ist $\boldsymbol{B} = L_B(G)$ für eine kontextfreie Grammatik G, wenn es eine endliche Menge $\{\mathfrak{L}_i : i < n\}$ von Bäumen der Höhe 1 gibt, und ein S derart, daß $\mathfrak{B} \in \boldsymbol{B}$ genau dann, wenn (i.) die Wurzel die Marke S trägt, (ii.) eine Marke terminal ist genau dann, wenn der Knoten ein Blatt ist, und (iii.) jeder lokale Teilbaum zu einem $\mathfrak{L}_i$ isomorph ist.*

Wir wollen ein paar nützliche Folgerungen aus dieser Betrachtung von kontextfreien Grammatiken ziehen. Es ist klar, daß γG Bäume generiert, die nicht notwendig Blätter mit Terminalsymbolen tragen. Aber wir wissen, daß die Blätter entweder Marken aus A oder aus $N^1 := N \times \{1\}$ tragen, während alle anderen Knoten Marken aus $N^0 := N \times \{0\}$ tragen. Definiere nun zu einem markierten Baum wie gehabt die assoziierte Zeichenkette $k(\mathfrak{B})$. Dies ist ein Element aus $(A \cup N^1)^*$. Sei $v : A \cup N \times 2 \to A \cup N$ definiert durch $v(a) = a$, $a \in A$ und $v(X^0) = v(X^1) = X$ für $X \in N$.

Lemma 1.6.5 *Es sei $\gamma G \vdash \mathfrak{B}$ und $\vec{\alpha} = k(\mathfrak{B})$. Dann gilt $\vec{\alpha} \in (A \cup N^1)^*$ und $G \vdash \overline{v}(\vec{\alpha})$.*

Beweis. Induktion über die Länge der Ableitung. Ist diese gleich 0, so ist $\vec{\alpha} = S^1$, und $\overline{v}(S^1) = S$. Da $G \vdash S$, ist dieser Fall erledigt. Nun sei $\mathfrak{B}$ das Ergebnis der Anwendung einer Regel ρ^γ aus $\mathfrak{A}$, mit $\rho = X \to \vec{\beta}$. Wir haben dann $k(\mathfrak{C}) \in (A \cup N^1)^*$. Die Regel ρ^γ wurde auf ein Blatt angewandt; dieses entspricht einem Vorkommen von X^1. Wir haben daher $k(\mathfrak{C}) = \vec{\delta}_1 \cdot X^1 \cdot \vec{\delta}_2$. Dann ist $k(\mathfrak{B}) = \vec{\delta}_1 \cdot \vec{\beta} \cdot \vec{\delta}_2$. Es ist $k(\mathfrak{B})$ das Ergebnis einer einmaligen Anwendung von ρ. $\dashv$

Definition 1.6.6 *Es sei $\mathfrak{B}$ ein Baum. Ein **Schnitt** durch $\mathfrak{B}$ ist eine maximale Menge S dergestalt, daß keine zwei Elemente vergleichbar sind. Falls $\mathfrak{B}$ ein erschöpfend geordneter markierter Baum ist, so ist ein Schnitt linear geordnet und markiert, und dann sei ein **Schnitt** auch die zu einem Schnitt gehörende Zeichenkette.*

Proposition 1.6.7 *Es sei $\gamma G \vdash \mathfrak{B}$, und $\vec{\alpha}$ ein Schnitt von $\mathfrak{B}$. Dann ist $G \vdash \overline{v}(\vec{\alpha})$.*

Dieser Satz zeigt, daß der Baum alle wesentlichen Informationen bereitstellt. Hat man den Baum, läßt sich eine Ableitung sofort angeben. Nun sei ein Baum $\mathfrak{B}$ gegeben, und $\vec{\alpha}$ ein Schnitt. Wir sagen, ein Vorkommen C von $\vec{\gamma}$ von $\vec{\alpha}$ sei eine **Konstituente vom Typ** X, falls dieses Vorkommen von $\vec{\gamma}$ der durch $\vec{\alpha}$ in einer Konstituente $\downarrow x$ definierte Schnitt ist, und x die Marke X trägt. Dies bedeutet, daß $\vec{\alpha} = \vec{\delta}_1 \cdot \vec{\gamma} \cdot \vec{\delta}_2$, $C = \langle \vec{\delta}_1, \vec{\delta}_2 \rangle$, und $\downarrow x$ genau diejenigen Knoten, die nicht zu $\vec{\delta}_1$ oder $\vec{\delta}_2$ gehören, enthält. Ferner sei ein Teilwortvorkommen von $\vec{\gamma}$ eine Konstituente vom Typ X in $\vec{\alpha}$, falls es einen Baum gibt, für den ein Schnitt $\vec{\alpha}$ existiert, in dem das Teilwortvorkommen $\vec{\gamma}$ eine Konstituente vom Typ X ist.

Lemma 1.6.8 *Es sei $\mathfrak{B}$ ein ein γG–Baum und $\vec{\alpha}$ ein Schnitt von $\mathfrak{B}$. Dann existiert ein Baum $\mathfrak{C}$ mit assoziierter Zeichenkette $\vec{\beta}$ und $\overline{v}(\vec{\beta}) = \overline{v}(\vec{\alpha})$.*

Lemma 1.6.9 *Es sei $G \vdash \vec{\alpha}_1 \cdot \vec{\gamma} \cdot \vec{\alpha}_2$ und das Teilwort $\vec{\gamma}$ sei eine Konstituente vom Typ X in $C(\vec{\beta})$. Dann ist $G \vdash \vec{\alpha}_1 \cdot X \cdot \vec{\alpha}_2 = C(X)$, wobei $C = \langle \vec{\alpha}_1, \vec{\alpha}_2 \rangle$.*

Als Beweis bemerke man, daß wenn $\vec{\alpha}_1 \cdot \vec{\gamma} \cdot \vec{\alpha}_2$ ein Schnitt ist und $\vec{\gamma}$ darin eine Konstituente vom Typ X, so ist $\vec{\alpha}_1 \cdot X \cdot \vec{\alpha}_2$ auch ein Schnitt.

Theorem 1.6.10 (Konstituentensubstitution) *Es sei $G \vdash \vec{\alpha}_1 \cdot \vec{\beta} \cdot \vec{\alpha}_2 = C(\vec{\beta})$ und $\vec{\beta}$ eine Konstituente vom Typ X in $C(\vec{\beta})$. Ferner gelte $X \vdash_G \vec{\gamma}$. Dann ist $G \vdash C(\vec{\gamma}) = \vec{\alpha}_1 \cdot \vec{\gamma} \cdot \vec{\alpha}_2$ und das Teilwort $\vec{\gamma}$ ist eine Konstituente vom Typ X in $C(\vec{\gamma})$.*

Beweis. Nach Voraussetzung existiert ein Baum, in welchem $\vec{\beta}$ eine Konstituente vom Typ X ist in $\vec{\alpha}_1 \cdot \vec{\beta} \cdot \vec{\alpha}_2$. Dann existiert ein Schnitt $\vec{\alpha}_1 \cdot X \cdot \vec{\alpha}_2$ durch diesen Baum und nach Lemma 1.6.8 ein Baum mit assoziierter Zeichenkette $\vec{\alpha}_1 \cdot X \cdot \vec{\alpha}_2$. Gewiß ist X in diesem Baum eine Konstituente. Dann aber kann man eine Derivation $X \vdash_G \vec{\gamma}$ ausweiten zu einer γG–Derivation von $\vec{\alpha}_1 \cdot \vec{\gamma} \cdot \vec{\alpha}_2$, in welcher $\vec{\gamma}$ eine Konstituente ist. $\dashv$

Wir wollen daraus ein wichtiges Theorem herleiten. Angenommen, G ist gegeben. Dann existiert eine Zahl k derart, daß jede Zeichenkette der Länge mindestens k zwei echt ineinander enthaltene Teilworte desselben Typs enthält. Genauer, wenn $|\vec{x}| \geq k$ ist, so existiert ein Baum $\mathfrak{B}$ und ein $X \in N$ derart, daß $\vec{x} = \vec{u} \cdot \vec{x} \cdot \vec{v} \cdot \vec{y} \cdot \vec{w}$ die assoziierte Kette ist und $\vec{x} \cdot \vec{v} \cdot \vec{y}$ sowie $\vec{v}$ Konstituenten vom Typ X, und schließlich $\vec{x} \neq \varepsilon$ oder $\vec{y} \neq \varepsilon$. Zur Vorbereitung beweisen wir das folgende

Lemma 1.6.11 *Es sei G eine kontextfreie Grammatik. Dann existiert eine Zahl k_G derart, daß zu jedem Ableitungsbaum einer Zeichenkette der Länge $\geq k_G$ zwei Konstituenten $\downarrow y$ und $\downarrow z$ gleichen Typs mit $y \leq z$ oder $z \leq y$, welche verschiedene assoziierte Zeichenketten haben.*

Beweis. Zunächst einmal muß man sich klarmachen, daß nichts sich an unserer Behauptung ändert, wenn wir unproduktive Regeln eliminieren. Dies geschieht ja unter Wahrung der Konstituentstruktur. Es sei π das Maximum aller Produktivitäten von Regeln in G und $\nu := |N|$. Dann ist $k_G := (1 + \pi)^\nu + 1$ die gesuchte Zahl. (Wir können annehmen, daß $\pi > 0$. Andernfalls erzeugt G nur Worte der Länge 1, und $k_G := 2$ erfüllt unsere Behauptung.) Denn sei ein $\vec{x}$ gegeben mit $|\vec{x}| \geq k_G$. Dann existiert in jedem Ableitungsbaum ein Zweig der Länge $> \nu$. (Falls nicht, so kann es nicht mehr als κ^ν Blätter geben.) Auf diesem Zweig haben zwei Nichtterminalsymbole die gleiche Marke. Deren assoziierte Zeichenkette sind verschieden, da keine unproduktiven Regeln existieren. $\dashv$

Wir sagen, ein Vorkommen von $\vec{u}$ sei eine **linke Konstituentenhälfte**, falls $\vec{u}$ Präfixvorkommen einer Konstituente ist. Ebenso ist eine **rechte Konstituentenhälfte** definiert. $\vec{x}$ enthält eine linke Konstituentenhälfte $\vec{z}$, falls ein Suffix von $\vec{u}$ eine linke Konstituente ist. Wir bemerken noch Folgendes. Ist $\vec{u}$ eine Konstituente und echtes Teilwort von $\vec{x}$, so ist $\vec{x} = \vec{v}\vec{u}\vec{v}_1$ und $\vec{v}_1$ ist linke Konstituenthälfte und $\vec{v}$ rechte Konstituentenhälfte. Dies ist im Folgenden bedeutsam.

Lemma 1.6.12 *Es sei G eine kontextfreie Grammatik. Dann existiert eine Zahl k'_G derart, daß zu jedem Ableitungsbaum einer Zeichenkette $\vec{x}$ und jedem Zeichenkettenvorkommen in $\vec{x}$ einer Zeichenkette $\vec{z}$ der Länge $\geq k'_G$ zwei linke oder zwei rechte Konstituentenhälften $\vec{y}$ und $\vec{y}_1$ von $\vec{z}$ gleichen Typs existieren mit $\vec{y} \neq \vec{y}_1$. Es ist dann notwendigerweise $\vec{y}$ ein Präfix von $\vec{y}_1$ oder $\vec{y}_1$ ein Präfix von $\vec{y}$, im Falle,*

daß beide linke Hälften sind, oder $\vec{y}$ Suffix von $\vec{y}_1$ oder $\vec{y}_1$ Suffix von $\vec{y}$ im Falle, daß beide rechte Hälften sind.

Beweis. Es sei $\nu := |N|$ und π die maximale Produktivität von G. Wir können sicherlich $\pi \geq 2$ annehmen. $k'_G := (2 + 2\pi)^\nu$. Wir zeigen induktiv über die Zahl m, daß eine Zeichenkette der Länge $\geq (2 + 2\pi)^m$ mindestens m linke oder mindestens m rechte ineinander enthaltene Hälften enthält. Ist $m = 1$, so ist die Behauptung gewiß wahr. Sei sie nun für $m \geq 1$ gezeigt. Wir zeigen sie jetzt für $m+1$. Es sei $\vec{z}$ von der Länge $\geq (2+2\pi)^{m+1}$. Es sei $\vec{x} = \prod_{i<2\pi+2} \vec{x}_i$ für gewisse $\vec{x}_i$ der Länge mindestens $(2 + 2\pi)^m$. Nach Induktionsvoraussetzung enthält jedes $\vec{x}_i$ mindestens m Konstituentenhälften. Nun haben wir nicht unbedingt $(2\pi+2)m$ Konstituentenhälften; denn enthält $\vec{x}_i$ eine linke Hälfte, so kann ein $\vec{x}_j$ mit $j > i$ die zugehörige rechte Hälfte enthalten. Zu einer linken Hälfte wird höchstens eine rechte Hälfte gezählt. Insgesamt haben wir also mindestens $(1+\pi)m \geq m+1$ Konstituentenhälften. Allerdings müssen wir noch verifizieren, daß mindestens $m+1$ davon ineinander enthalten sind. Sei dies für kein $\vec{x}_i$ der Fall. Dann haben alle $\vec{x}_i$, $i < 2\pi + 2$, stets genau m linke oder genau m rechte Hälften. Fall 1. $\vec{x}_0$ hat m ineinanderliegende linke Hälften. Hat dann $\vec{x}_1$ auch m ineindanderliegende linke Hälften, so sind wir fertig. Nehmen wir an, dies sei nicht der Fall. Dann hat $\vec{x}_1$ hat m ineinanderliegende rechte Hälften. Dadurch bekommen wir jetzt m ineinanderliegende Konstituenten. Wiederum wären wir fertig, wenn $\vec{x}_2$ m rechte Hälften besitzt (dies erfordert einen kurzen Beweis). Falls dies nicht der Fall ist, hat $\vec{x}_2$ genau m linke Hälften. Wiederum sind wir fertig, wenn dies nicht die zu den linken Hälften von $\vec{x}_3$ passenden rechten Hälften wären. Nehmen wir dies also an. Wir argumentieren so weiter. Wir bekommen schließlich eine mindestens π lange Folge von Konstituenten, die jeweils m ineinanderliegende Konstituenten besitzen. Eine Teilfolge von diesen bildet eine Konstituente, welche nun die Höhe $m + 1$ hat. Dies zeigt unsere Hilfsbehauptung. Mit $m := \nu + 1$ folgt nun die Behauptung des Satzes. $\dashv$

Theorem 1.6.13 (Pumplemma) *Zu einer kontextfreien Sprache S existiert ein p_S derart, daß für jede Zeichenkette $\vec{z} \in S$ der Länge $\geq p_S$ und ein Vorkommen $\vec{r}$ einer Zeichenkette der Länge $\geq p_S$ in $\vec{z}$ eine Zerlegung*

$$\vec{z} = \vec{u} \cdot \vec{x} \cdot \vec{v} \cdot \vec{y} \cdot \vec{w}$$

existiert, sodaß gilt

1. $\vec{x}\,\vec{y} \neq \varepsilon$,
2. $\vec{u}\,\vec{w} \neq \varepsilon$,
3. *$\vec{x}$ oder $\vec{y}$ sind in dem Vorkommen von $\vec{r}$ enthalten.*
4. $\{\vec{u} \cdot \vec{x}^i \cdot \vec{v} \cdot \vec{y}^i \cdot \vec{w} : i \in \omega\} \subseteq S$.

Alternativ zu 3. kann man verlangen, daß $|\vec{v}| \leq p_S$. Ferner läßt sich p_S noch so wählen, daß sich jede beliebige ableitbare Zeichenkette $\vec{\gamma}$ mit vorgegebenem Vorkommen einer Zeichenkette $\vec{\alpha}$ der Länge p_S auf diese Weise zerlegen läßt.

Beweis. Es sei G eine Grammatik, welche S erzeugt. Sei p_S die in Lemma 1.6.12 definierte Konstante. Wir betrachten einen G–Ableitungsbaum von $\vec{z}$ sowie das darin definierte Teilwortvorkommen von $\vec{r}$. Angenommen, $\vec{r}$ habe die Länge mindestens p_S. Dann existieren zwei linke oder zwei rechte Konstituenthälften gleichen Typs in $\vec{r}$. Ohne Beschränkung der Allgemeinheit nehmen wir an, $\vec{r}$ besitze zwei linke Hälften gleichen Typs. Nehmen wir an, diese seien beide nicht ganz in $\vec{r}$ enthalten. Also ist $\vec{r} = \vec{s}\vec{x}\vec{s}_1$, wo $\vec{x}\vec{s}_1$ und $\vec{s}_1$ linke Konstituentenhälften gleichen Typs, etwa X, sind. Es ist $|\vec{x}| > 0$. Es existieren jetzt $\vec{s}_2$ und $\vec{y}$ derart, daß $\vec{v} := \vec{s}_1\vec{s}_2$ und $\vec{x}\vec{s}_1\vec{s}_2\vec{y}$ Konstituenten des Typs X sind.

Also existiert eine Zerlegung

$$\vec{z} = \vec{u} \cdot \vec{x} \cdot \vec{v} \cdot \vec{y} \cdot \vec{w}$$

wo $\vec{v}$ eine Konstituente des gleichen Typs ist wie $\vec{x}\vec{v}\vec{y}$ mit den Eigenschaften 1., 2. und 3. Aufgrund des Konstituentensubstitutionslemmas kann man dann $\vec{x}\vec{v}\vec{y}$ durch $\vec{v}$ ersetzen, wie auch $\vec{v}$ durch $\vec{x}\vec{v}\vec{y}$. Dies ergibt, durch eine einfache Induktion, die Behauptung 4. Sei nun die kleinere Hälfte in $\vec{r}$ enthalten, die größere nicht. Dann existiert eine Zerlegung $\vec{r} = \vec{s}\vec{x}\vec{v}\vec{s}_1$ derart, daß $\vec{v}$ eine Konstituente vom Typ X ist, und $\vec{x}\vec{v}\vec{s}_1$ linke Hälfte einer Konstituente vom Typ X. Dann existiert ein $\vec{s}_2$ derart, daß auch $\vec{x}\vec{v}\vec{s}_1\vec{s}_2$ Konstituente vom Typ X ist. Setze dann $\vec{y} := \vec{s}_1\vec{s}_2$. Es ist dann auch $\vec{y} \neq \varepsilon$. Der dritte Fall ist, wenn beide Hälften echte Teilkonstituenten sind. Auch hier findet man eine geforderte Zerlegung. Will man anstelle von 3. haben, daß $\vec{v}$ möglichst klein ist, so überlege man sich, daß in unserer Zerlegung $\vec{v}$ bereits eine Konstituente ist. Hat sie die Länge $\geq (1+\pi)^\nu$, so existiert eine Zerlegung von $\vec{v}$ derart, daß darin pumpbare Teilworte vorhanden sind. Also kann statt 3. fordern, daß $|\vec{v}| \leq p_G$. $\dashv$

Das Pumplemma kann mit Kontexten kompakter so formuliert werden. Für jede genügend große Zeichenkette $\vec{x}$ existieren Kontexte C, D, $D \neq \langle \varepsilon, \varepsilon \rangle$, und eine Zeichenkette $\vec{y}$ mit $\vec{x} = D(C(\vec{x}))$, und es ist $D(C^k(\vec{y})) \in S$ für jedes $k \in \omega$.

Man beachte, daß man auch $i = 0$ wählen kann. Dies bedeutet, daß man nicht nur die Zeichenkette 'hochpumpen' kann, so daß sie länger wird (außer wenn $i = 1$, aber dieser Fall ist ja nicht interessant), sondern man kann sie auch schrumpfen lassen. Eine Anwendung dieses Lemmas besteht in dem Beweis, daß die Sprache $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n : n \in \omega\}$ nicht kontextfrei sein kann. Denn angenommen, sie sei kontextfrei. Dann müßte ein m existieren derart, daß für genügend großes k die Kette $\mathsf{a}^k\mathsf{b}^k\mathsf{c}^k$ zerlegt werden kann in

$$\mathsf{a}^k\mathsf{b}^k\mathsf{c}^k = \vec{u} \cdot \vec{v} \cdot \vec{w} \cdot \vec{x} \cdot \vec{y}$$

Außerdem muß ein $\ell > k$ existieren mit

$$\mathtt{a}^\ell\mathtt{b}^\ell\mathtt{c}^\ell = \vec{u}\cdot\vec{v}^2\cdot\vec{w}\cdot\vec{x}^2\cdot\vec{y}$$

Dann enthält $\vec{v}\cdot\vec{x}$ je genau $\ell - k$ mal die Buchstaben $\mathtt{a}$, $\mathtt{b}$ und $\mathtt{c}$. Man überlegt sich nun, daß $\vec{v} \subseteq \mathtt{a}^*\cup\mathtt{b}^*\cup\mathtt{c}^*$. Denn enthalte $\vec{v}$ zwei verschiedene Buchstaben, etwa $\mathtt{b}$ und $\mathtt{c}$ so enthält $\vec{v}$ ein Vorkommen von $\mathtt{b}$ vor einem Vorkommen von $\mathtt{c}$. $\vec{v}^2$ enthält dann ein Vorkommen von $\mathtt{c}$ vor einem Vorkommen von $\mathtt{b}$, und das darf nicht sein. Analog zeigt man, daß $\vec{y} \in \mathtt{a}^* \cup \mathtt{b}^* \cup \mathtt{c}^*$. Dies ist aber ein Widerspruch. Dieses Beispiel einer nicht kontextfreien Sprache wird uns noch öfter begegnen.

Als zweites Beispiel wollen wir die sogenannten *Baumadjunktionsgrammatiken* vorstellen. Wir nehmen ein Alphabet A und eine Menge N von nichtterminalen Symbolen. Sei $S \in N$ das Startsymbol. Ein **Zentralbaum** ist ein geordneter Baum über $A \cup N$, bei dem alle Blätter Marken aus A tragen, alle übrigen Knoten Marken aus N, und die Wurzel die Marke S. Ein **Adjunktionsbaum** ist ein geordneter Baum über $A \cup N$, bei dem die Wurzel und mindestens ein Blatt eine Marke $X \in N$ trägt und alle anderen Blätter Marken aus A, und alle übrigen Knoten Marken aus N. Es wird verlangt, daß ein Adjunktionsbaum mindestens ein Blatt mit einem Terminalsymbol trägt. Eine **unregulierte Baumadjunktionsgrammatik**, abgekürzt UBAG, ist ein Paar $\langle \mathfrak{C}, \mathfrak{A}\rangle$, wo $\mathfrak{C}$ eine endliche Menge von Zentralbäumen und $\mathfrak{A}$ eine endliche Menge von Adjunktionsbäumen ist. Ein Beispiel einer Baumadjunktion ist in Figur 1.5 gegeben. Dabei wird in den Baum zur Linken ein Zentralbaum mit Wurzel X und assoziierter Zeichenkette $\mathtt{bXb}$ adjungiert; das Ergebnis ist rechts zu sehen. Unabhängig von dem gegebenen Kontext kann man eine Baumadjunktion mit einem Adjunktionsbaum wie folgt beschreiben. Es sei $\mathfrak{B} = \langle B, <, \sqsubset, \ell\rangle$ ein Baum und $\mathfrak{A} = \langle A, <, \sqsubset, m\rangle$ ein Adjunktionsbaum. Wir nehmen an, es sei r die Wurzel von $\mathfrak{A}$, sowie s dasjenige Blatt mit $m(r) = m(s)$. Nun sei x ein Knoten in B mit $\ell(x) = m(r)$. Dann ist die Ersetzung von x durch $\mathfrak{B}$ definiert durch Angabe der Farbfunktionale. Diese sind

$$\mathfrak{II}_\rho(y,\sqsubset) := \begin{cases} \{\sqsubset,<\} & \text{wenn } s \sqsubset y \\ \{\sqsubset\} & \text{sonst} \end{cases} \qquad \mathfrak{OI}_\rho(y,\sqsubset) := \varnothing$$

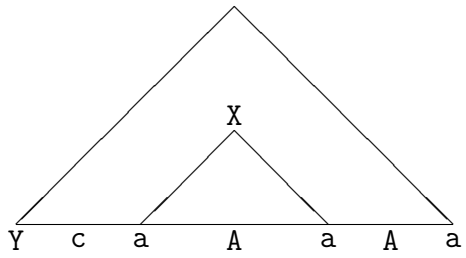
$$\mathfrak{IO}_\rho(y,\sqsubset) := \begin{cases} \{<\} & \text{wenn } y \sqsubset s \\ \varnothing & \text{sonst} \end{cases} \qquad \mathfrak{OO}_\rho(y,\sqsubset) := \{\sqsubset\}$$

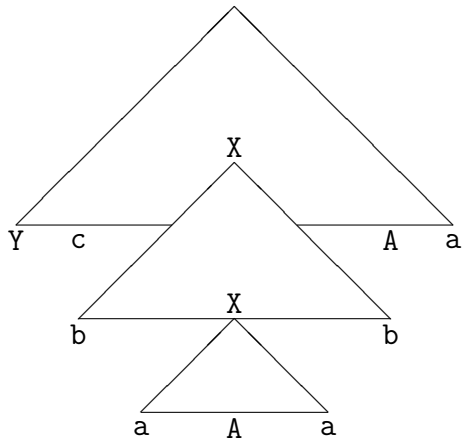
$$\mathfrak{II}_\rho(y,<) := \{<\} \qquad \mathfrak{IO}_\rho(y,<) := \varnothing$$

$$\mathfrak{OI}\rho(y,<) := \varnothing \qquad \mathfrak{OO}_\rho(y,<) := \{<\}$$

Zwei Punkte sind zu bemerken. Erstens, wir haben statt eines einzigen Startgraphen eine endliche Menge. Dies ist behebbar. Zweitens sind im alle Farben sowohl terminal als auch nichtterminal. Man kann also die Ableitung an jeder Stelle abbrechen. Wie wir im vorhergehenden Abschnitt überlegt haben, ist dies behebbar, indem man noch einige zusätzliche Symbole einführt. Wir wollen dies jedoch nicht tun, und uns mit dem Hinweis begnügen, daß wir es strenggenommen mit einer quasi–Grammatik zu tun haben.

Abbildung 1.5: Baumadjunktion





Zuletzt wollen wir eine Graphgrammatik angeben, welche alle Zeichenketten der Form $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, $n > 0$, erzeugen kann. Die Idee dazu stammt von Uwe Mönnich. Dabei wollen wir ausnützen, daß man ja auch Terme als Strukturen auffassen kann, und deshalb können wir die Grammatik relativ bündig aufschreiben. Dazu nehmen wir ein dreistelliges Symbol, F, welches nichtterminal ist, ein dreistelliges Symbol, f, welches terminal ist, und ein zweistelliges terminales Symbol $\frown$. Die Regeln sind wie folgt. (Zur einfacheren Lesbarkeit schreiben wir den Ausdruck nicht in Polnischer Notation, sondern mit Hilfe von Klammern.)

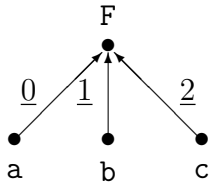
$$\begin{array}{lcl} \mathtt{F}(x,y,z) & \to & \mathtt{F}(\mathtt{a}\frown x, \mathtt{b}\frown y, \mathtt{c}\frown z), \\ \mathtt{F}(x,y,z) & \to & \mathtt{f}(x,y,z), \end{array}$$

Diese Regeln bilden ein sogenanntes **Termersetzungssystem**. Der Start wird von dem Term $\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c})$ gebildet. Wann immer man einen Term t hat, in dem ein Teilterm s auftritt, welche Ergebnis der Substitution in eine linke Seite ist, so kann man anstelle von s in t die entsprechende rechte Seite setzen. Damit bekommt man zum Beispiel folgende Ableitungen:

$$\begin{array}{lll} \mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c}) & \to \mathtt{f}(\mathtt{a},\mathtt{b},\mathtt{c}), & \\ \mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c}) & \to \mathtt{F}(\mathtt{a}\frown\mathtt{a}, \mathtt{b}\frown\mathtt{b}, \mathtt{c}\frown\mathtt{c}) & \to \mathtt{f}(\mathtt{a}\frown\mathtt{a}, \mathtt{b}\frown\mathtt{b}, \mathtt{c}\frown\mathtt{c}) \\ \mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c}) & \to \mathtt{F}(\mathtt{a}\frown\mathtt{a}, \mathtt{b}\frown\mathtt{b}, \mathtt{c}\frown\mathtt{c}) & \to \mathtt{F}(\mathtt{a}\frown(\mathtt{a}\frown\mathtt{a}), \mathtt{b}\frown(\mathtt{b}\frown\mathtt{b}), \mathtt{c}\frown(\mathtt{c}\frown\mathtt{c})) \\ & & \to \mathtt{f}(\mathtt{a}\frown(\mathtt{a}\frown\mathtt{a}), \mathtt{b}\frown(\mathtt{b}\frown\mathtt{b}), \mathtt{c}\frown(\mathtt{c}\frown\mathtt{c})) \end{array}$$

Man beachte, daß die Terme hier für Graphen stehen. Wir verwenden dabei die Dependenzkodierung. Daher sind die zu diesen Termen assoziierten Zeichenketten gerade abc, aabbcc und aaabbbccc.

Um nun eine Graphgrammatik zu schreiben, die diese Graphmengen erzeugt, muß man Hilfsfarben für Kanten einführen. Wir setzen $F_E := \{\mathtt{F},\mathtt{f},\mathtt{a},\mathtt{b},\mathtt{c}\}$, $F_E^T := \{\mathtt{f},\mathtt{a},\mathtt{b},\mathtt{c}\}$ und $F_K := \{\underline{0},\underline{1},\underline{2},\sqsubset,<\}$. Der Startgraph ist wie folgt. Er hat drei Punkte, p, q, r und s. ($<$ ist leer (!), und $q \sqsubset r \sqsubset s$.) Die Markierung ist $p \mapsto \mathtt{F}$, $q \mapsto \mathtt{a}$, $r \mapsto \mathtt{b}$ und $s \mapsto \mathtt{c}$.



Es gibt zwei Ersetzungsregeln. Die erste kann man schematisch so beschreiben. Die Wurzel, x, trägt den Namen F und hat drei einlaufende Kanten; deren Namen sind $\underline{0}$, $\underline{1}$ und $\underline{2}$. Daran hängen die Teilgraphen $\mathfrak{G}_0$, $\mathfrak{G}_1$ und $\mathfrak{G}_2$, welche Bäume sind in Bezug auf $<$ und $\sqsubset$ und in denen es keine Kanten mit Namen $\underline{0}$, $\underline{1}$ und $\underline{2}$ gibt. Im Ersetzungsschritt wird x ersetzt durch einen Graphen mit sieben Knoten, p, q_i, r_i

und s_i, $i \in \{0,1\}$, wobei $q_i \sqsubset r_j \sqsubset s_k$, $i,j,k \in \{0,1\}$, und $q \xrightarrow{\underline{0}} p$, $r \xrightarrow{\underline{1}} p$ und $s \xrightarrow{\underline{2}} p$. $<= \{\langle q_1, q_0\rangle, \langle r_1, r_0\rangle, \langle s_1, s_0\rangle\}$. Die Färbung ist

$$\begin{array}{llll} p & \mapsto & \mathtt{F}, & \\ q_0 & \mapsto & \frown, & q_1 \mapsto \mathtt{a}, \\ r_0 & \mapsto & \frown, & r_1 \mapsto \mathtt{b}, \\ s_0 & \mapsto & \frown, & s_1 \mapsto \mathtt{c}. \end{array}$$

(Wir reproduzieren damit auf $\{p, q_0, r_0, s_0\}$ die Anfangssituation.) Der Baum $\mathfrak{G}_0$ wird nun an q_0 rechts von q_1 angehängt, der Baum $\mathfrak{G}_1$ an r_0 rechts von r_1 und der Baum $\mathfrak{G}_2$ an s_0 rechts von s_1. Wir setzen zusätzlich alle Knoten von den $\mathfrak{G}_i$ $<$ als p. Damit wird erreicht, daß in jedem Schritt die Vereinigung der Relationen $<$, $\underline{0}$, $\underline{1}$ und $\underline{2}$ die intendierte Baumordnung ist und jeweils eine Kante mit Namen $\underline{0}$, $\underline{1}$ und $\underline{2}$ existiert, welche in die Wurzel einläuft.

Die zweite Ersetzungsregel ersetzt den Wurzelpunkt durch einen einpunktigen Graphen, dessen Knoten den Namen `f` trägt. Dadurch terminiert die Ableitung. Die Kanten mit Namen $\underline{0}$, $\underline{1}$ und $\underline{2}$ werden nun unter dem Namen $<$ weitergereicht. Damit entsteht ein Baum. Dieser hat die gewünschte Form.

Übung 19. Man kann auch Zeichenketten als Multigraphen auffassen. Dabei braucht man natürlich nur eine Kantenfarbe. Zeigen Sie, daß eine kontextfreie Grammatik für Zeichenketten auch formuliert werden kann als kontextfreie γ–Grammatik. Es wird sich im nächsten Kapitel herausstellen, daß kontextfreie Sprachen auch durch unregulierte Baumadjunktionsgrammatiken erzeugen lassen, jedoch die Umkehrung nicht gilt.

Übung 20. Zeigen Sie, daß zu jeder kontextfreien γ–Grammatik Γ eine kontextfreie Grammatik Δ existiert, welche keine Regeln der Produktivität -1 hat und die gleiche Menge von γ–Graphen erzeugt.

Übung 21. Zeigen Sie, daß zu einer kontextfreien γ–Grammatik eine erzeugungsgleiche kontextfreie γ–Grammatik gibt, in der alle Regeln strikt produktiv sind.

Übung 22. Formulieren Sie in Analogie zu unregulierten Baumadjunktionsgrammatiken Kettenadjunktionsgrammatiken. Beachten Sie, daß diese quasi–Grammatiken sind. Geben Sie eine Charakterisierung der erzeugten Bäume in Form von Regeln in Kettenersetzungsgrammatiken.

Übung 23. Zeigen Sie, daß die Sprache $\{\vec{w} \cdot \vec{w} : \vec{w} \in A^*\}$ nicht kontextfrei ist, daß sie aber das Pumplemma erfüllt.

1.7 Turingmaschinen

Alan Turing verdanken wir das Konzept einer Maschine, welche denkbar einfach ist und dennoch alle uns bekannten berechenbaren Funktionen berechnen kann. Es würde zu weit führen, den Begriff der Berechenbarkeit zu problematisieren. Stattdessen benützen wir den Begriff der *Turingberechenbarkeit*, welcher nur auf die Turingmaschine bezugnimmt. Der Clou an Turings Konzept war es, daß Zahlen mit Hilfe von Zeichenketten dargestellt wurden und zwar auf denkbar einfache Weise. Die Zahl n wurde durch $n+1$ Striche notiert. (Dadurch entsprach der 0 der einzelne Strich. Diese Konvention ist unverzichtbar.) Zusätzlich gab es noch das Leerzeichen, welches als Trennhilfe eingesetzt wird. In dieser Konzeption sind Turingmaschinen denkbar schwerfällig. Wir erlauben daher, daß das Alphabet der Maschine beliebig groß ist. Der Leser wird anschließend aufgefordert zu zeigen, daß schon ein Alphabet aus zwei Buchstaben zuzüglich einem Leerzeichen ausreicht. Wie also funktioniert eine Turingmaschine? Eine Turingmaschine operiert auf einem Band, welches unendlich viele Felder hat. Wir nehmen an, daß das Band nach beiden Seiten unendlich ist. Jedes dieser Felder kann einen Buchstaben tragen oder leer sein, d. h. das Leerzeichen tragen. Die Maschine hat einen Kopf, der sowohl lesen wie auch schreiben kann und der in jedem Moment genau ein Feld bearbeitet. Die Maschine kann den Kopf jeweils ein Feld nach rechts oder ein Feld nach links schieben und sich so zu jeder beliebigen Stelle des Bandes bewegen. Schließlich hat die Maschine endlich viele Zustände, und diese Zustände bestimmen, was die Maschine zu welchem Zeitpunkt machen soll. Wir können im Übrigen anstelle eines unendlich langen Bandes auch annehmen, daß die Maschine auf einer endlichen Zeichenkette entlangläuft, welche sie falls nötig links oder rechts verlängern kann. Wir geben der letzteren Interpretation deshalb den Vorzug, weil sie näher an dem von uns betrachteten Gegenstand liegt.

Definition 1.7.1 *Eine **(nichtdeterministische) Turingmaschine** ist ein Quintupel $\langle A, \mathtt{L}, Q, q_0, f\rangle$, wo A eine endliche Menge ist, das **Alphabet**, $\mathtt{L} \notin A$ das sogenannte **Leerzeichen**, Q eine endliche Menge, die Menge der **Zustände**, $q_0 \in Q$ das sogenannte **Startsymbol** und $f : (A \cup \{\mathtt{L}\}) \times Q \to \wp((A \cup \{\mathtt{L}\}) \times \{-1, 0, 1\} \times Q)$, genannt die **Übergangsfunktion**. Ist $|f(\langle b, q\rangle)| \leq 1$, so heißt die Maschine **deterministisch**.*

Wir schreiben im Folgenden A_L anstelle von $A \cup \{\mathtt{L}\}$. Die Arbeitsweise einer Turingmaschine können wir bequem über Zeichenketten veranschaulichen. Eine **Konfiguration** ist ein Tripel $\langle \vec{x}, i, q\rangle$, wo $\vec{x} \in A_L^*$, $i < |\vec{x}|$, und $q \in Q$. Dabei bezeichnet i die Stelle in der Zeichenkette, an welcher der Lesekopf ist und q den Zustand, in welchem sich die Maschine befindet. Eine **Z–Konfiguration** ist eine Zeichenkette aus $A_L^* \times Q \times A_L^*$. Ist dann $\vec{x} \cdot q \cdot \vec{y}$ eine Z–Konfiguration, und $|\vec{x}| = j$, so entspricht ihr

die Konfiguration $\langle \vec{x} \cdot \vec{y}, j, q \rangle$. Wir haben also in der Zeichenkette ein Symbol links von dem Lesekopf eingeschoben, welches den Zustand der Maschine bezeichnet.

Wir wollen nun die Arbeitsweise der Maschine als Übergänge zwischen Z–Konfigurationen angeben. Wir sagen, $\vec{x} \cdot q \cdot \vec{y}$ gehe durch T in einem Schritt in $\vec{x}' \cdot q' \cdot \vec{y}'$ über, falls eines der Folgenden gilt:

1. $\vec{x}' = \vec{x}$, und für ein gewisses $\vec{v}$ sowie b und c ist $\vec{y} = b \cdot \vec{v}$ und $\vec{y}' = c \cdot \vec{v}$, und $\langle c, 0, q' \rangle \in f(\langle b, q \rangle)$.
2. Es ist $\vec{x}' = \vec{x} \cdot c$ und $\vec{y} = b \cdot \vec{y}'$ sowie $\langle c, 1, q' \rangle \in f(\langle b, q \rangle)$.
3. Es ist $\vec{x} = \vec{x}' \cdot c$ und $\vec{y}' = b \cdot \vec{y}$ sowie $\langle c, -1, q' \rangle \in f(\langle b, q \rangle)$.

Ist also $\langle c, 0, q' \rangle \in f(\langle q, b \rangle)$ und der Lesekopf liest b, während die Maschine in dem Zustand q ist, wird auf das gelesene Feld der Buchstabe c geschrieben und der Kopf nicht verschoben. Ist stattdessen $\langle c, 1, q' \rangle \in f(\langle q, b \rangle)$, so wird c auf das Feld geschrieben und der Kopf nach rechts gerückt. Steht anstelle der 1 eine -1, so wird der Kopf ein Feld nach links geschoben. Wir schreiben $Z \Rightarrow_T Z'$ ($Z \Rightarrow_T^k Z'$), falls Z' aus Z in einem Schritt (in k Schritten) hervorgeht, und $Z \Rightarrow_T^* Z'$, falls Z' aus Z in endlich vielen Schritten hervorgeht.

Definition 1.7.2 *Es sei T eine Turingmaschine, Z eine Konfiguration und $\vec{x} \in A^*$. Z ist eine **Endkonfiguration**, falls es keine Konfiguration Z' gibt mit $Z \Rightarrow_T Z'$. T **akzeptiert** $\vec{x}$, falls es eine Endkonfiguration Z gibt mit $q_0 \cdot \vec{x} \Rightarrow_T^* Z$. Die von T akzeptierte Sprache, $L(T)$, ist die Menge aller Zeichenketten aus A^*, welche T akzeptiert.*

Es bedarf einiger Gewöhnung, bis man das Konzept einer Turingmaschine sowie der von ihr akzeptierten Sprache verstanden hat. Dem interessierten Leser sei empfohlen, ein paar einfache Funktionen auf einer Turingmaschine zu programmieren. Wir wollen im Folgenden ein paar allgemeine Sätze beweisen und so bleibt uns wenig Zeit für konkrete Beispiele; solche werden allerdings auf dem Weg zu diesen Sätzen ohnehin anfallen. Ein erstes Beispiel ist die Nachfolgermaschine. Angenommen, unser Alphabet sei $\{\mathtt{a}_i : i < p\}$. Wir wollen eine Maschine bauen, welche zu $\vec{x}$ das in der numerischen Ordnung nächste Wort ausgibt. (Siehe dazu Abschnitt 1.2.) Dies soll heißen, daß falls die Maschine mit $q_0 \cdot \vec{x}$ beginnt, so hält sie in der Konfiguration $q_0 \cdot \vec{y}$, wo $\vec{y}$ das jeweils nächste Wort ist. Wenn wir im Folgenden von Zahlen reden, meinen wir stillschweigend oft statt der Zahl n die $n$te Zeichenkette in der numerischen Ordnung.

Wie baut man nun die Maschine, welche jedem Wort seinen Nachfolger zuordnet? Wir benötigen 3 Zustände, q_0, q_1 und q_2. Ist die Maschine in q_0 und liest ein Symbol

a_j, wobei $j < n-1$ ist, so schreibt sie a_{j+1} und geht in den Zustand q_1 über. Liest sie a_{n-1}, so schreibt sie a_0, geht nach rechts und verbleibt im Zustand q_0. Liest sie hingegen L, so schreibt sie a_1 und geht in den Zustand q_1 über. Ist die Maschine im Zustand q_1 und liest $a \neq L$, so schreibt sie a, geht nach links und bleibt im Zustand q_1. Liest sie aber L, so geht sie nach rechts und geht in den Zustand q_2 über. Im Zustand q_2 sind keine Aktionen definiert. Initialisiert man diese Maschine bei $q_0 \cdot \vec{x}$, so wird sie von links her kommend entweder das nächstbeste a_j mit $j < n-1$ durch a_{j+1} ersetzen (bzw. L durch a_1). Ist ihr dies geglückt (und es glückt immer, da die Maschine andernfalls an das Ende gerät und dann a_1 schreibt), so geht sie anschließend nach links zurück. Damit der Übertrag korrekt ausgeführt wird, muß auf dem Weg nach rechts a_{n-1} stets durch a_0 ersetzt werden. Es ist auch nicht schwer, eine Vorgängermaschine zu bauen (dies ist eine Übung), und man kann auch Folgendes zeigen. (Hierbei ist *berechenbar* im Sinne der Definition weiter unter zu verstehen.)

Lemma 1.7.3 *Es seien A und B endliche Alphabete. Es gibt berechenbare Bijektionen $f : A^* \to B^*$ und $g : B^* \to A^*$ mit $f = g^{-1}$.*

In der Rekursionstheorie wird der Begriff der Berechenbarkeit üblicherweise für Funktionen auf den natürlichen Zahlen definiert. Mit Hilfe der Abbildung Z, welche ja bijektiv ist, lassen sich diese Begriffe ebenso auch für Funktionen auf Zeichenketten definieren, was wir hier tun werden.

Definition 1.7.4 *Es seien A und B Alphabete. Es sei $f : A^* \to B^*$ eine beliebige Funktion. f heißt **berechenbar**, falls es eine deterministische Turingmaschine T gibt derart, daß für jedes $\vec{x} \in A^*$ gilt $q_0 \cdot \vec{x} \Rightarrow_T^* q_t \cdot f(\vec{x})$ und $q_t \cdot f(\vec{x})$ ist eine Endkonfiguration. Es sei $S \subseteq A^*$. S heißt **aufzählbar**, falls $S = \varnothing$ oder es eine berechenbare Funktion $f : \{\mathtt{0},\mathtt{1}\}^* \to A^*$ gibt mit $f[\{\mathtt{0},\mathtt{1}\}^*] = S$. S heißt **entscheidbar**, falls sowohl S als auch $A^* - S$ aufzählbar sind.*

In diesem Abschnitt werden wir zeigen, daß die aufzählbaren Mengen genau diejenigen sind, welche von einer Turingmaschine akzeptiert werden. Ferner zeigen wir, daß es genau die Typ 0 Sprachen sind, womit eine erste Korrespondenz zwischen Typen von Automaten und Typen von Sprachen hergestellt ist. Anschließend werden wir zeigen, daß das Erkennungsproblem für Typ 0 Sprachen nicht entscheidbar ist. Die Beweise folgen letztlich aus einer Reihe von Reduktionssätzen für Turingmaschinen. Zunächst werden wir das Konzept einer Turingmaschine verallgemeinern. Eine k–Band Turingmaschine ist ein Quintupel $\langle A, \mathtt{L}, Q, q_0, f\rangle$, wobei A, $\mathtt{L}$, Q, q_0 wie vorher sind, aber

$$f : A_L^k \times Q \to \wp(A_L^k \times \{-1, 0, 1\} \times Q)$$

Anschaulich bedeutet dies, daß die Turingmaschine jetzt anstelle des einen Bandes k viele Bänder hat. Auf jedem Band ist ein Kopf, und in jedem Schritt kann

die Maschine einen dieser Köpfe unabhängig von den anderen bewegen, wobei der ausgeführte Schritt von den gelesenen k Symbolen und dem Zustand abhängt. Die Startkonfiguration ist dabei diese: alle Bänder bis auf das erste sind leer, der Kopf irgendwo. Auf dem ersten Band ist der Kopf links von der Eingabe. Die k–Band Maschine hat also im Gegensatz zur 1–Bandmaschine $k-1$ Bänder zum Speichern von Zwischenergebnissen. Der Leser möge sich vergewissern, daß wir auch solche Konfigurationen als Startkonfiguration zulassen können, bei denen auf jedem Band eine Zeichenkette ist, an derem linken Ende sich der Kopf befindet. Denn diese können wir auch als k Zeichenketten auf dem ersten Band interpretieren. Dazu lassen wir auch zu, daß sich rechts von den Köpfen endliche Folgen von Zeichenketten befinden, jeweils getrennt durch ein einzelnes Leerzeichen. (Das Leerzeichen dient der Trennung der Ketten. Damit die Maschine allerdings weiß, wann diese Folge zu Ende ist, muß man verlangen, daß es immer genau ein Leerzeichen ist.) Dies ändert nichts an den bisherigen Konzepten, erlaubt allerdings, die Maschine auf mehr Eingaben rechnen zu lassen als bisher, nämlich Zahlenfolgen anstelle von Zahlen. Ohne uns in Einzelheiten zu verlieren, wollen wir plausibel machen, daß k–Band Turingmaschinen nicht mehr Funktionen berechnen können wie 1–Band Maschinen. Dazu überlege man sich folgende Kodierung der k Bänder auf einem. Wir teilen das eine Band in Sektoren mit je $2k$ Feldern auf. Das Feld $2kp+2m$, $m<k$, enthält den Eintrag von Feld p auf Band m. Das Feld $2kp+2m+1$ enthält dagegen lediglich die Information, ob der Kopf m auf diesem Feld zu finden ist. (Also ist jedes zweite Feld nur mit, sagen wir, `0` oder `1` beschriftet, je nachdem einer der Köpfe auf einem entsprechende Feld zu finden ist.) Hat man nun eine k–Band Turingmaschine T, so konstruiert man eine 1–Bandmaschine U wie folgt. Immer wenn die Maschine U einen Schritt ausführen soll, muß sie das Wort von links nach rechts ablaufen und die relevanten Stellen der Leseköpfe registrieren. Ist sie beim Ende angekommen, weiß sie, welcher Zustand von T repräsentiert wird, und kann nun, nach links laufend, die Leseköpfe neu positionieren und die Einträge auf den relevanten Feldern abändern. Dann geht sie an den Anfang zurück, und beginnt mit dem nächsten Schritt von T. Man überlegt sich, daß somit jeder Schritt der k Band Maschine $|\vec{x}|$ Zeit kostet, wobei $\vec{x}$ die Summe der Wörter der k Bänder ist. Ist also ein Algorithmus auf einer k Band Maschine in $f(n)$ Zeit lösbar, so benötigt man höchstens $(f(n)+n)^2$ Zeit auf einer 1–Band Maschine, da ja in $f(n)$ Zeitschritten das Band höchstens ein Wort der Länge $f(n)+n$ schreiben kann.

Wir werden dies benützen, um zunächst zu zeigen, daß eine nichtdeterministische Turingmaschine nicht mehr berechnen kann als eine deterministische.

Proposition 1.7.5 *Ist $S=L(T)$ für eine Turingmaschine, so ist auch $S=L(U)$ für eine deterministische Turingmaschine.*

Beweis. Sei $S=L(T)$. Es sei $|f(\langle q,x\rangle)|<b$ für ein b. Wir fixieren eine Ordnung auf $f(\langle q,x\rangle)$. Wir nehmen eine 3–Band–Maschine V wie folgt. Auf das erste Band

schreiben wir die Eingabe $\vec{x}$. Auf dem zweiten Band generieren wir alle Sequenzen $\vec{p}$ von Zahlen $< b$. Diese Sequenzen beschreiben die Aktionssequenzen von T. Mit jeder neuen Sequenz $\vec{p} = a_0 a_1 \dots a_q$ schreiben wir auf das dritte Band $\vec{x}$ (welches wir von Band 1 nur kopieren müssen) und lassen V nun wie folgt arbeiten. Der Kopf auf Band 2 ist links von der Sequenz $\vec{a}$. Im ersten Schritt befolgen wir auf Band 3 die $a_0$te Alternative für die Maschine T in der Übergangsmenge, rücken den Kopf 2 nach rechts, im zweiten Schritt befolgen wir die $a_1$te Alternative in der Übergangsmenge für T auf Band 3, und so weiter. Dabei liest der Kopf 2 also jeweils, welche der Alternativen der Übergangsmenge von T auf Band 3 genommen werden soll. Falls für ein i die $a_i$te Alternative nicht existiert, hält die Berechnung auf Band 3. Damit wird T also künstlich deterministisch gemacht. Die Maschine V hält jedoch nur dann an, wenn sie für ein k bei allen Sequenzen der Länge k auf Band 3 anhält. Der Leser mag sich überlegen, wie die Maschine dies überprüfen kann. ⊣

Lemma 1.7.6 *Genau dann ist S aufzählbar, wenn $S = L(T)$ für eine Turingmaschine T ist.*

Beweis. Der Fall $S = \varnothing$ muß gesondert betrachtet werden. Es ist einfach, eine Maschine zu konstruieren, die auf keinem Wort anhält. Diese beweist die Äquivalenz in diesem Fall. Nun sei also $S \neq \varnothing$. Es sei S aufzählbar. Dann existiert eine Funktion $f : \{0,1\}^* \to A^*$ mit $f[\{\mathtt{0},\mathtt{1}\}^*] = S$ und Turingmaschine U, welche die Funktion f berechnet. Nun konstruieren wir eine 3–Band–Maschine V wie folgt. Die Eingabe ist $\vec{x}$, und sie wird auf dem ersten Band notiert. Auf dem zweiten Band wird V alle Worte $\vec{y} \in \{\mathtt{0},\mathtt{1}\}^*$ erzeugen, beginnend mit ε, in der lexikographischen Ordnung. Dazu nehmen wir die Nachfolgermaschine. Ist ein Wort $\vec{y}$ neu berechnet, beginnt die Maschine, auf dem dritten Band den Wert $f(\vec{y})$ zu berechnen. Da f berechenbar ist, wird sie irgendwann damit fertig sein. Anschließend vergleicht sie $f(\vec{y})$ mit $\vec{x}$. Sind diese gleich, so hält sie an. Andernfalls aber wird sie den Nachfolger von $\vec{y}$ ausrechnen und weitermachen. Es ist leicht zu sehen, daß $S = L(V)$. Nach den vorangegangenen Überlegungen existert nun eine 1–Band–Maschine T mit $S = L(T)$. Nun sei umgekehrt $S = L(T)$ für eine Turingmaschine T. Wir wollen zeigen, daß S aufzählbar ist. Wir dürfen annehmen, daß T deterministisch ist. Nun konstruieren wir eine Turingmaschine U, welche eine Funktion $f : \{\mathtt{0},\mathtt{1}\}^* \to A^*$ berechnet, deren Bild S ist. Dazu sei V eine 4–Band–Maschine, die wie folgt arbeitet. Wir beginnen mit $\vec{u}$ auf Band 1. Auf Band 2 schreibt die Maschine das Wort, `0`. Nun berechnet sie auf dem vierten Band einen T–Schritt für 0 und kopiert das Ergebnis auf das dritte Band. In jedem folgenden Schritt wird die Maschine dieses tun: auf Band 1 einen Nachfolger konstruieren, und an das letzte Wort auf Band 3 anfügen (durch `L` getrennt!). Nun nimmt sie das erste Wort auf Band 3 und berechnet eine T–Schritt auf Band 4. (Also: auf Band 4 kopieren, einen T–Schritt ausrechnen, und das Ergebnis auf Band 3 an das Ende schreiben, durch `L` getrennt.) Falls auf ein Wort kein T–Schritt anwendbar ist, vermindert die Maschine die Sequenz auf Band 1 um eins,

erhöht gleichzeitig aber die Sequenz auf Band 2 um eins und kopiert den Inhalt von Band 2 ans Ende von Band 3 (durch `L` getrennt). Ist sie auf Band 1 bei ε angekommen, kopiert sie das Wort von Band 4 auf Band 1 und hält an. Da es mindestens ein Wort gibt, bei dem T anhält, wird die Maschine für jede Eingangssequenz $\vec{u}$ ein Ergebnis berechnen. Denn sei $\vec{x} \in S$, so wird $\vec{x}$ auf Band 3 immer wieder an den Anfang kommen, und die Maschine wird immer wieder feststellen, daß T hier anhält, und so den Zähler auf Band 1 verringern. Daher hält die Maschine gewiß an. Daß sie auch wirklich alle T–Berechnungen simuliert, liegt an der speziellen Verwaltung. Denn sie bildet eine Liste auf Band 3, bei der das jeweils erste Wort bearbeitet und dann an das Ende geschoben wird. Ist die Liste zu diesem Zeitpunkt k Einträge lang, so wird das Ergebnis spätestens nach k Berechnungen von T–Schritten an die Reihe kommen. So wird die Maschine an jedem Wort beliebig tief rechnen. Falls T auf diesem Wort anhält wird V dieses Wort also für ein genügend großes $\vec{u}$ ausgeben. $\dashv$

Theorem 1.7.7 *Es sind äquivalent:*

1. *S ist vom Typ 0.*
2. *S ist aufzählbar.*
3. *$S = L(T)$ für eine Turingmaschine T.*

Beweis. Wir zeigen (1) $\Rightarrow$ (2) und (3) $\Rightarrow$ (1). Der Satz folgt dann mit Lemma 1.7.6. Sei also S vom Typ 0. Dann existiert eine Grammatik $\langle A, N, \mathtt{S}, R\rangle$, welche S erzeugt. Man muß nun lediglich eine Turingmaschine konstruieren, welche alle aus `S` erzeugbaren Zeichenketten auflistet und solche wieder entfernt, welche nicht in A^* sind. Dazu bauen wir eine Maschine mit drei Bändern, deren erstes die Zahl n enthält, und welche auf Band 2 zuerst `S` schreibt, und dann auf `S` jede mögliche Regel anwendet, und diese Ergebnisse der Reihe nach auf dem Band notiert. Wann immer eine Zeichenkette terminal ist, wird sie auf Band 2 geschrieben. Hat Band 2 schließlich n Ketten, so hört die Maschine auf. Die letzte Zeichenkette ist das Ergebnis. Man muß nun sicherstellen, daß auf Band 2 für jedes Wort jede mögliche Regelanwendung aufgelistet wird. Dazu nimmt die Maschine das Wort und fügt links ein neues Zeichen $\flat$ an. $\flat$ notiert das linke Ende des Anwendungsbereichs einer Regel. Falls unmittelbar rechts von $\flat$ Regeln anwendbar sind, so werden alle diese Regeln nacheinander angewendet, und $\flat$ wird ein Feld nach rechts geschoben. Diese Maschine generiert also alle Ableitungen auf Band 2, und zählt so S auf. Nun sei $S = L(T)$ für eine Turingmaschine. Man wähle folgende Grammatik G: zuzüglich zu A sei `X` das Startsymbol, `0` und `1` zwei Nichtterminalsymbole, und $\mathtt{Y}_q$ ein Nichtterminalsymbol

für jedes $q \in Q$. Folgendes seien die Regeln.

$$\begin{array}{lcll}
\mathtt{X} & \to & \mathtt{X0} \mid \mathtt{X1} \mid \mathtt{Y}_{q_0} & \\
\mathtt{Y}_q b & \to & c\mathtt{Y}_r & \text{falls } \langle c, 1, r\rangle \in f(\langle b, q\rangle) \\
\mathtt{Y}_q b & \to & \mathtt{Y}_r c & \text{falls } \langle c, 0, r\rangle \in f(\langle b, q\rangle) \\
a\mathtt{Y}_q b & \to & \mathtt{Y}_r cb & \text{falls } \langle c, -1, r\rangle \in f(\langle b, q\rangle) \\
\mathtt{Y}_q b & \to & b & \text{falls } f(\langle b, q\rangle) = \varnothing
\end{array}$$

Ausgehend von $\mathtt{X}$ erzeugt die Grammatik nun Ketten der Form $\mathtt{Y}_{q_0}\vec{x}$, wobei $\vec{x}$ eine Binärfolge ist. Diese kodiert die Eingabe für T. Die weiteren Regeln kodieren in anschaulicher Weise die Berechnung von T auf der Zeichenkette. Falls die Berechnung anhält, so ist es erlaubt, $\mathtt{Y}_q$ zu eliminieren. Ist die Zeichenkette terminal, so wird sie daher auch von G erzeugt. So sieht man, daß $L(G) = L(T)$. $\dashv$

Nun wollen wir daraus einen wichtigen Satz ableiten, nämlich, daß es unentscheidbare Sprachen gibt. Dazu überlegen wir uns, daß man Turingmaschinen auf einfache Weise als Zeichenketten darstellen kann. Nun kann man eine Maschine U bauen, die zwei Eingaben bekommt, nämlich eine Zeichenkette $\vec{x}$ und einen Kode für eine Turingmaschine T, und die nun die Aktion von T auf $\vec{x}$ berechnet. Eine solche Maschine heißt **universelle Turingmaschine**. Die Kodierung von Turingmaschinen ist wie folgt. Wir benützen nur die Buchstaben $\mathtt{a}$, $\mathtt{b}$ und $\mathtt{c}$, welche auch in A enthalten sein sollen. Es sei $A = \{\mathtt{a}_i : i < n\}$. Dann sei $\gamma(a_i)$ gerade die Zahl i in Binärdarstellung (über $\{\mathtt{a}, \mathtt{b}\}$, mit $\mathtt{a}$ anstelle von $\mathtt{0}$ und $\mathtt{b}$ anstelle von $\mathtt{1}$). Die Zahl Null werde mit $\mathtt{a}$ kodiert, um sie von ε zu unterscheiden. Ferner sei $\mathtt{L}$ der Zahl n zugeordnet. Die Zustände werden ebenso kodiert; wir nehmen dabei an, daß $Q = \{0, 1, \ldots, n-1\}$ für ein n ist, und daß $q_0 = 0$. Nun müssen wir nur noch f aufschreiben. f ist eine Teilmenge von

$$A_L \times Q \times A_L \times \{-1, 0, 1\} \times Q$$

Jedes Element $\langle a, q, b, m, r\rangle$ von f kann wie folgt notiert werden

$$\vec{x} \cdot \mathtt{c} \cdot \vec{u} \cdot \mathtt{c} \cdot \vec{\mu} \cdot \mathtt{c} \cdot \vec{y} \cdot \mathtt{c} \cdot \vec{v} \cdot \mathtt{c}$$

wobei $\vec{x} = \gamma(a)$ ist, $\vec{u} = Z^{-1}(q)$, $\vec{y} = \gamma(b)$, $\vec{v} = Z^{-1}(r)$. Ferner ist $\vec{\mu} = \mathtt{a}$, falls $m = -1$, $\vec{\mu} = \mathtt{b}$, falls $m = 0$, und $\vec{\mu} = \mathtt{ab}$, falls $m = 1$. Nun notieren wir f einfach als Liste, die Einträge jeweils durch $\mathtt{c}$ getrennt. Je zwei Quintupel werden also durch $\mathtt{cc}$ getrennt. (Das muß nicht sein, ist aber praktischer.) Wir nennen diese Kodewort von T einfach $T^{\spadesuit}$. Die Menge aller Kodewörter von Turingmaschinen ist entscheidbar. (Das ist wesentlich, ist aber leicht zu sehen.) Es sollte nun nicht schwer sein, sich davon zu überzeugen, daß es eine Maschine U mit zwei Bändern gibt, welche für zwei Ketten $\vec{x}$ und $\vec{y}$ Folgendes tut. Ist $\vec{y} = T^{\spadesuit}$, so berechnet U genau das, was T mit $\vec{x}$ berechnet. Ist $\vec{y}$ kein Kode einer Maschine, so geht U in einen speziellen Zustand und hält an.

Wir nehmen zunächst an, es gebe eine Turingmaschine V, welche für $\vec{x}$ und $T^{\spadesuit}$ entscheidet, ob $\vec{x} \in L(T)$ ist oder nicht. Nun konstruieren wir eine 2–Band–Maschine W wie folgt. Die Eingabe ist das Wort $\vec{x}$ auf beiden Bändern. Ist $\vec{x} = T^{\spadesuit}$, so berechnet W, ob T auf $\vec{x}$ anhält. (Dazu benütze man V.) Wenn ja, so wird W in eine unendliche Schleife geschickt und hört deshalb nicht mehr auf zu rechnen. Wenn nein, hält W an. (Ist $\vec{x}$ kein Kodewort, so hält die Maschine an.) Nun gilt: $W^{\spadesuit} \in L(W)$ genau dann, wenn $W^{\spadesuit} \notin L(W)$. Denn $W^{\spadesuit} \in L(W)$ genau dann, wenn W, auf $W^{\spadesuit}$ losgelassen, anhält. Ist dies aber der Fall, so soll W gerade nicht anhalten. Ist nun $W^{\spadesuit} \notin L(W)$, so hält W bei $W^{\spadesuit}$ nicht an, was wir aber mit Hilfe von V entscheiden können, und deswegen hält W bei $W^{\spadesuit}$ eben doch an. Widerspruch. V kann also nicht existieren. Es gibt also kein allgemeines Verfahren. Wenn es dieses also nicht gibt, dann kann dennoch für jede Turingmaschine entscheidbar sein, ob $\vec{x}$ von ihr akzeptiert wird oder nicht. (Nur wissen wir dann nicht, wie wir das Entscheidungsverfahren aus $T^{\spadesuit}$ berechnen können.) Daß auch Letzteres im Prinzip nicht gehen kann, dafür setzen wir unsere universelle Turingmaschine ein, und zwar in der 1–Band–Version. Sie heiße U. Ist $L(U)$ entscheidbar, so können wir zu jedem $\vec{x}$ und jedem T (universell) entscheiden, ob U auf $\vec{x}\mathbb{L}T^{\spadesuit}$ anhält. Da U universell ist, heißt das, daß wir mittels U entscheiden können, ob T auf $\vec{x}$ anhält. Dies können wir jedoch nicht.

Theorem 1.7.8 *Es existiert eine aufzählbare Menge, welche nicht entscheidbar ist. Es ist also* $\mathrm{CSS} \subsetneq \mathrm{GS}$.

Hiermit haben wir auch gezeigt, daß die Typ 1 Sprachen echt in den Typ 0 enthalten sind. Denn Typ 1 Sprachen sind entscheidbar.

Theorem 1.7.9 *Jede Typ 1 Sprache ist entscheidbar.*

Beweis. Es sei G vom Typ 1 und sei $\vec{x}$ gegeben. Dann erzeuge alle G–Ableitungen bis zur Länge $|N \cup A||\vec{x}|$. Tritt $\vec{x}$ darin auf, so ist $\vec{x} \in L(G)$. Andernfalls ist aber $\vec{x} \notin L(G)$. Ist $n = |\vec{x}|$, $\alpha = |A \cup N|$ und k die Anzahl der Regeln in G, so gibt es $k^{\alpha n}$ viele Ableitungen der Länge αn. $\dashv$

Bevor wir dieses Kapitel beenden, wollen wir ein paar Maße für die Schwierigkeit einer Berechnung einführen. Im Folgenden wird es oft darum gehen, den Platz– und Zeitverbrauch einer Turingmaschine zu messen. Dies wird als das Maß der von ihr erkannten Sprache dienen. Dazu sei $f : \omega \to \omega$ eine Funktion. Sei T eine Turingmaschine, welche eine Funktion $g : A^* \to A^*$ berechnet. Wir sagen, T brauche $O(f)$–**Platz**, falls es eine Konstante c gibt, sodaß für fast jedes $\vec{x} \in A^*$ gilt: es gibt eine Berechnung von $q_0 \cdot g(\vec{x})$ aus $q_0 \cdot \vec{x}$, in welcher jede Konfiguration die Länge $\leq c \times f(|\vec{x}|)$ hat. (Hierbei heißt *fast jedes* $\vec{x}$: nur endlich viele $\vec{x}$ erfüllen dies

nicht.) Bei einer Mehrbandmaschine summieren wir einfach die Längen der entstehenden Worte auf den Bändern. T braucht $O(f)$–**Zeit**, falls für fast alle $\vec{x} \in A^*$ gilt: $q_0 \cdot \vec{x} \Rightarrow_T^k q_0 \cdot g(\vec{x})$ für ein $k \leq c \times f(|\vec{x}|)$. Es ist $DSPACE(f)$ ($DTIME(f)$) die Menge aller Funktionen, welche für ein k von einer deterministischen k–Band Turingmaschine in $O(f)$–Platz ($O(f)$–Zeit) berechnet werden können. Analog die Notation $NSPACE(f)$ und $NTIME(f)$ für nichtdeterministische Maschinen. Es gilt stets:

$$DTIME(f) \subseteq NTIME(f) \subseteq NSPACE(f)$$

sowie

$$DSPACE(f) \subseteq NSPACE(f)$$

Denn eine Maschine kann in k Schritten höchstens k Felder bearbeiten, egal ob sie deterministisch ist oder nicht. Dies gilt auch für Maschinen mit mehreren Bändern, da ein Schritt jeweils nur das Bearbeiten eines Feldes beinhaltet.

Der Grund, warum man keinen Unterschied macht zwischen dem Zeitbedarf $f(n)$ und dem Zeitbedarf $cf(n)$, c eine Konstante, liegt in dem folgenden Satz.

Theorem 1.7.10 (Speed Up Theorem) *Es sei f eine berechenbare Funktion, und es sei T eine Turingmaschine, welche $f(\vec{x})$ in höchstens $g(|\vec{x}|)$ Zeitschritten (mit höchstens $h(|\vec{x}|)$ Platz) berechnet, wobei $\inf_{n\to\infty} g(n)/n = \infty$. Ferner sei c eine beliebige reelle Zahl > 0. Dann existiert eine Turingmaschine U, welche f in höchstens $c \cdot g(|\vec{x}|)$ Zeitschritten (mit höchstens $c \cdot h(|\vec{x}|)$ Platz) berechnet.*

Der Beweis ergibt sich aus folgender Tatsache. Wir können anstelle des Alphabets $A_L = A \cup \{\mathtt{L}\}$ ein Alphabet $B_L = B \cup \{\mathtt{L}_1\}$ betrachten, wo jedem Symbol aus B genau eine Zeichenkette der Länge k aus $A \cup \{\mathtt{L}\}$ entspricht. Dem Symbol $\mathtt{L}_1$ entspricht dann $\mathtt{L}^k$. Wir kodieren nun die Eingabe um und rechnen nun in B_L anstelle von A_L. Da einem Einzelzeichen ein Block der Länge k entspricht, sinkt der Platzbedarf um den Faktor k. (Falls wir die Eingabe über A_L betrachten, so kann der Platzbedarf allerdings niemals unter $|\vec{x}|$ sinken. Diesen Fall muß man gesondert betrachten.) Ebenso sinkt bei der Berechnung der Zeitbedarf, weil es möglich ist, viele Bewegungen der Maschine T in eine zusammenzufassen. Allerdings ist die Zeitreduktion nicht so einfach zu bestimmen wie die Platzreduktion. Auch hier muß man den Zeitbedarf bei der Kodierung der Eingabe und der Rekodierung der Ausgabe mitrechnen, was jedoch wegen der eingebauten Bedingung an f nicht nötig ist. Die Details kann man im Buch von Hopcroft und Ullman [25] nachlesen.

Übung 24. Man konstruiere eine Turingmaschine, die zu jeder Zeichenkette den lexikographischen Vorgänger berechnet und zu ε wieder ε.

Übung 25. Man konstruiere eine Turingmaschine, die bei einer Liste von Zeichenketten die erste Zeichenkette an das Ende verschiebt.

Übung 26. Es sei T eine Turingmaschine über dem Alphabet A. Zeigen Sie, wie

man eine Turingmaschine über $\{\mathtt{0},\mathtt{1}\}$, schreibt, welche dieselbe partielle Funktion wie A berechnet unter der Kodierung, daß jedem Zeichen aus A ein Block der Länge β für ein festes β zugeordnet wird.

Übung 27. In vielen Büchern ist das Band einer Turingmaschine nach links nicht offen. Dann muß man entweder ein spezielles Symbol, etwa $\sharp$, einführen, welches den linken Rand des Bandes markiert, oder aber ein Prädikat left-end, welches wahr wird, wenn man sich am linken Rand befindet. Die Übergänge können nun verschieden sein, je nachdem, ob left-end wahr ist oder nicht. (Die Alternative, die Berechnung abzubrechen, wenn die Maschine nach links rücken soll und dies nicht tun kann, ist denkbar schlecht. Eine solche Maschine kann wesentlich weniger berechnen als eine Turingmaschine, da das linke Ende der Eingabe unmarkiert bleibt.) Man zeige, daß sich Maschinen mit beidseitig offenem Band in Maschinen mit halbseitig offenem Band überführen lassen (und umgekehrt) unter Wahrung der akzeptierten Sprache.

Übung 28. Beweisen Sie Lemma 1.7.3. *Hinweis.* Zeigen Sie zuerst, daß es genügt, den Fall $|A| = 1$ zu betrachten.

Übung 29. Eine Menge $S \subseteq A^*$ möge **berechenbar** heißen, wenn die charakteristische Funktion $\chi_S : A^* \to \{\mathtt{0},\mathtt{1}\}$ berechenbar ist. Dabei ist $\chi_S(\vec{x}) = 1$ genau dann, wenn $\vec{x} \in S$. Zeigen Sie, daß S genau dann entscheidbar ist, wenn S berechenbar ist.

Kapitel 2

Kontextfreie Sprachen

2.1 Reguläre Sprachen

Typ 3 Grammatiken sind die einfachsten in der Chomsky–Hierarchie. Es gibt eine mehrfache Charakterisierung von regulären Sprachen, nämlich über endliche Automaten, über Lösungen von Gleichungen über Zeichenketten, sowie über reguläre Ausdrücke. Bevor wir beginnen, wollen wir uns eine möglichst einfache Form für reguläre Grammatiken verschaffen. Zunächst lassen sich alle Regeln von der Form $X \to Y$ eliminieren. Dazu setzen wir als neue Regelmenge

$$\begin{aligned} R^{\heartsuit} := \quad & \{X \to aY : \text{ für ein } Z \in N : X \to aZ \in R \text{ und } Z \Rightarrow_R Y\} \\ \cup \quad & \{X \to \vec{x} : \text{ für ein } Z \in N : X \Rightarrow_R Z \text{ und } Z \to \vec{x} \in R, \vec{x} \in A_\varepsilon\} \end{aligned}$$

Man zeigt leicht, daß mit $R^{\heartsuit}$ die gleichen Zeichenketten ableitbar sind. Nun wollen wir noch eine zweite Vereinfachung einführen. Wir führen neue nichtterminale Symbole U_a ein für jedes $a \in A$. Anstatt der Regeln $X \to a$ nehmen wir die Regeln $X \to aU_a$ und zusätzlich die Regel $U_a \to \varepsilon$. Damit ist jede Regel strikt expandierend mit Ausnahme der Regeln $Y \to \varepsilon$, welche sogar kontrahierend sind. (Diese Grammatik ist nicht regulär, aber sie erzeugt die gleiche Sprache.) Allerdings kann man zeigen, daß in einer Ableitung nur die letzte Regelanwendung eine Anwendung einer kontrahierenden Regel ist. Denn man kann mit den expandierenden Regeln nur Zeichenketten der Form $\vec{x} \cdot Y$ ableiten, wo $\vec{x} \in A^*$ und $Y \in N$. Wendet man eine Regel $Y \to \varepsilon$ an, so verschwindet das Nichtterminalsymbol Y und die Ableitung ist beendet. Wir nennen eine reguläre Grammatik **strikt binär**, falls es nur Regeln der Form $X \to aY$ sowie $X \to \varepsilon$ gibt.

Definition 2.1.1 *Sei A ein Alphabet. Ein **(partieller) endlicher Automat über** A ist ein Quadrupel $\mathfrak{A} = \langle Q, i_0, F, \delta \rangle$, sodaß $i_0 \in Q$, $F \subseteq Q$ und $\delta : Q \times A \to \wp(Q)$.*

*Q ist die Menge der **Zustände**, i_0 heißt der **Anfangszustand**, F die Menge der **Endzustände** und δ die Übergangsfunktion. $\mathfrak{A}$ heißt **deterministisch**, falls $\delta(q,a)$ genau ein Element hat für jedes $q \in Q$ und $a \in A$.*

δ läßt sich auf Mengen von Zuständen sowie Zeichenketten wie folgt fortsetzen.

$$\begin{array}{lcl} \delta(S,\varepsilon) & := & S \\ \delta(S,a) & := & \bigcup\langle \delta(q,a) : q \in S\rangle \\ \delta(S,\vec{x}\cdot\vec{y}) & := & \delta(\delta(S,\vec{x}),\vec{y}) \end{array}$$

Damit können wir die Menge der akzeptierten Zeichenketten definieren.

$$L(\mathfrak{A}) = \{\vec{x} : \delta(\{i_0\},\vec{x}) \cap F \neq \varnothing\}$$

Ein Automat ist echt partiell, falls es einen Zustand q gibt und ein $a \in A$ derart, daß $\delta(q,a) = \varnothing$. Man kann einen Automaten stets so umformen, daß er nicht mehr partiell ist. Dazu muß man nur einen neuen Zustand $q_\varnothing$ einführen mit folgender Setzung für die Übergangsfunktion δ^+.

$$\delta^+(q,a) := \begin{cases} \delta(q,a) & \text{falls} \quad \delta(q,a) \neq \varnothing, q \neq q_\varnothing, \\ q_\varnothing & \text{falls} \quad \delta(q,a) = \varnothing, q \neq q_\varnothing, \\ q_\varnothing & \text{falls} \quad q = q_\varnothing. \end{cases}$$

Ferner soll $q_\varnothing$ kein akzeptierender Zustand sein. Im Falle eines deterministischen Automaten ist $\delta(q,\vec{x}) = \{q'\}$ für ein gewisses q'. In diesem Falle lassen wir die Mengenklammern weg und schreiben $\delta(q,\vec{x}) = q'$. Damit läßt sich die Definition der akzeptierten Sprache umwandeln in

$$L(\mathfrak{A}) = \{\vec{x} : \delta(i_0,\vec{x}) \in F\}\ .$$

Man kann einen nichtdeterministischen Automaten in einen deterministischen Automaten umformen, der die gleiche Sprache akzeptiert. Setze nämlich

$$\mathfrak{A}^d := \langle \wp(Q), \{i_0\}, F^d, \delta\rangle$$

wo $F^d := \{G \subseteq Q : G \cap F \neq \varnothing\}$ und δ die auf Mengen fortgesetzte Funktion aus $\mathfrak{A}$ ist.

Proposition 2.1.2 *$\mathfrak{A}^d$ ist deterministisch und $L(\mathfrak{A}^d) = L(\mathfrak{A})$. Also ist jede durch einen endlichen Automaten akzeptierte Sprache auch eine durch einen endlichen deterministischen Automaten akzeptierte Sprache.*

Der Beweis ist einfach und als Übung überlassen. Wir wollen nun als erstes überlegen, daß eine reguläre Sprache tatsächlich eine Sprache ist, welche von einem endlichen

Automaten erkannt wird. Wir können annehmen, daß G strikt binär ist. Sei $G = \langle S, N, A, R\rangle$. Wir setzen dann $Q_G := N$, $i_0 := S$, $F_G := \{X : X \to \varepsilon \in R\}$ sowie

$$\delta_G(X, a) := \{Y : X \to aY \in R\}$$

Setze $\mathfrak{A}_G := \langle Q_G, i_0, F_G, \delta_G\rangle$.

Lemma 2.1.3 *Für alle $X, Y \in N$ und $\vec{x}$ der Länge > 0 gilt: $Y \in \delta(X, \vec{x})$ genau dann, wenn $X \Rightarrow_R \vec{x} \cdot Y$.*

Beweis. Induktion über die Länge von $\vec{x}$. Der Fall $|\vec{x}| = \varepsilon$ ist klar. Sei $\vec{x} = a \in A$. Dann ist $Y \in \delta_G(X, a)$ nach Definition genau dann, wenn $X \to aY \in R$, und daraus folgt $X \Rightarrow_R aY$. Umgekehrt folgt aus $X \Rightarrow_R aY$, daß $X \to aY$ eine Regel aus R sein muß. Denn da in der Ableitung nur strikt expandierende Regeln angewandt worden sind, kann es sich bei der Ableitung von aY aus X nur um die einfache Anwendung einer Regel handeln. Nun sei $\vec{x} = \vec{y} \cdot a$. Nach Definition von δ_G ist dann

$$\delta_G(X, \vec{x}) = \delta_G(\delta_G(X, \vec{y}), a)$$

Also existiert ein Z dergestalt, daß $Z \in \delta_G(X, \vec{y})$ und $Y \in \delta_G(Z, a)$. Nach Induktionsannahme ist dies äquivalent mit $X \Rightarrow_R \vec{y} \cdot Z$ und $Z \Rightarrow_R aY$. Daraus folgt nun $X \Rightarrow_R \vec{y} \cdot a \cdot Y = \vec{x} \cdot Y$. Umgekehrt folgt aus $X \Rightarrow_R \vec{x} \cdot Y$ die Existenz eines Z mit $X \Rightarrow_R \vec{y} \cdot Z$ und $Z \Rightarrow_R aY$. Dies liegt daran, daß die Grammatik regulär ist. Jetzt folgt mit Induktionsannahme $Z \in \delta_G(X, \vec{y})$ und $Y \in \delta_G(Z, a)$, und so $Y \in \delta_G(\vec{x}, X)$. $\dashv$

Proposition 2.1.4 $L(\mathfrak{A}_G) = L(G)$.

Beweis. Es ist leicht einzusehen, daß $L(G) = \{\vec{x} : G \vdash \vec{x} \cdot Y, Y \to \varepsilon \in R\}$. Es gilt gemäß obenstehendem Lemma $\vec{x} \cdot Y \in L(G)$ genau dann, wenn $S \Rightarrow_R \vec{x} \cdot Y$. Letzteres ist gleichbedeutend mit $Y \in \delta_G(S, \vec{x})$. Dies ist nichts anderes als $\vec{x} \in L(\mathfrak{A}_G)$. Daher ist $L(G) = L(\mathfrak{A}_G)$. $\dashv$

Falls wir nun einen endlichen Automaten $\mathfrak{A} = \langle Q, i_0, F, \delta\rangle$ haben, so können wir wie folgt eine Grammatik definieren. Wir setzen $N := Q$, $S := i_0$. R enthält alle Regeln der Form $X \to aY$, wo $Y \in \delta(X, a)$ sowie alle Regeln der Form $X \to \varepsilon$ für $X \in F$. Dies ist die Grammatik $G_{\mathfrak{A}}$. Sie ist strikt binär und es gilt $\mathfrak{A}_{G_{\mathfrak{A}}} = \mathfrak{A}$. Deswegen haben wir $L(G_{\mathfrak{A}}) = L(\mathfrak{A})$.

Theorem 2.1.5 *Die regulären Sprachen sind genau die Sprachen, welche von einem endlichen deterministischen Automaten akzeptiert werden.* $\dashv$

Wir wollen uns nun einer weiteren Charakterisierung zuwenden. Ein **regulärer Term über** A ist ein Term, welcher aus A mit Hilfe von 0, ε, $\cdot$, $\cup$ und * aufgebaut werden kann. Ein regulärer Term definiert eine Sprache über A wie folgt

$$\begin{array}{lcl} L(0) & := & \varnothing \\ L(\varepsilon) & := & \{\varepsilon\} \\ L(a) & := & \{a\} \\ L(R \cdot S) & := & L(R) \cdot L(S) \\ L(R \cup S) & := & L(R) \cup L(S) \\ L(R^*) & := & L(R)^* \end{array}$$

Sprachen, welche durch reguläre Terme definiert sind, können auch als kleinste Lösungen von relativ einfachen Gleichungssystemen erhalten werden. Wir führen dazu Variablen ein (etwa X, Y und Z), welche für Sprachen stehen und formulieren mit Hilfe der Operationen $\cdot$, $\cup$ und * gewisse Gleichungen, welche eine oder mehrere Lösungen zulassen.

Lemma 2.1.6 *Es sei $\varepsilon \notin L(R)$. Dann ist R^* die eindeutig bestimmte Lösung von*

$$X = \varepsilon \cup R \cdot X$$

Beweis. (Man überlege sich, daß der Satz auch dann gilt, wenn $R = 0$. Dies sei im Folgenden daher ausgeschlossen.) Der Beweis erfolgt durch Induktion über die Länge von $\vec{x}$. $\vec{x} \in X$ bedeutet nach Definition, daß $\vec{x} \in \varepsilon \cup R \cdot X$. Falls $\vec{x} = \varepsilon$, so ist $\vec{x} \in R^*$. Sei also $\vec{x} \neq \varepsilon$; so ist $\vec{x} \in R \cdot X$, also von der Form $\vec{u}_0 \cdot \vec{x}_0$ mit $\vec{u}_0 \in R$ und $\vec{x}_0 \in X$. Da $\vec{u}_0 \neq \varepsilon$, hat $\vec{x}_0$ kleinere Länge als $\vec{x}$. Nach Induktionsannahme ist also $\vec{x}_0 \in R^*$. Also $\vec{x} \in R^*$. Die andere Richtung ist ebenso leicht. $\dashv$

Lemma 2.1.7 *Es seien C, D reguläre Terme und $\varepsilon \notin L(D)$. Die Gleichung $X = C \cup D \cdot X$ hat genau eine Lösung, nämlich $D^* \cdot C$.* $\dashv$

Wir zeigen nun, daß sich reguläre Sprachen ganz allgemein als Lösungen von Gleichungssystemen darstellen lassen. Ein allgemeines Gleichungssystem besteht aus Gleichungen von der Form $X_j = Q \cup \bigcup_{i<m} T^i$, wo Q ein regulärer Term ist und die T^i wiederum die Form $R \cdot X_k$ haben, R ein regulärer Term. Hier ist ein Beispiel.

$$\begin{array}{lcl} X_0 & = & \mathtt{a}^* \cup \mathtt{c} \cdot \mathtt{a} \cdot \mathtt{b} \cdot X_1 \\ X_1 & = & \mathtt{c} \cup \mathtt{c} \cdot \mathtt{b}^3 \cdot X_0 \end{array}$$

Man beachte, daß wie in anderen Gleichungssystemen auch nicht jede Variable rechts auftaucht. Außerdem soll ein Gleichungssystem jede Variable höchstens einmal links enthalten. Das Gleichungssystem heißt **echt**, falls für kein T^i_j gilt $\varepsilon \in L(T^i_j)$. Ein

Gleichungssystem wollen wir **einfach** nennen, falls es echt ist und Q und T_j^i nur aus Elementen von A und ε mit Hilfe von $\cup$ zusammengesetzt sind. Das obenstehende Gleichungssystem ist echt, aber nicht einfach.

Sei nun $\langle S, N, A, R\rangle$ eine strikt binäre reguläre Grammatik. Führe für jedes Nichtterminalsymbol X eine Variable Q_X ein. Die Variable Q_X stehe für die Menge aller Zeichenketten, welche die Grammatik als Konstituenten von der Kategorie X analysiert, das heißt, alle Zeichenketten $\vec{x}$, für die gilt $X \Rightarrow_R \vec{x}$. Letztere Menge bezeichnen wir mit $[X]$. Wir behaupten, daß diese Q_X folgenden Gleichungen genügen.

$$\begin{aligned} Q_Y = \quad & \bigcup\{\varepsilon : Y \to \varepsilon \in R\} \\ \cup \quad & \bigcup\{a \cdot Q_X : Y \to aX \in R\} \end{aligned}$$

Dieses Gleichungssystem ist einfach. Wir zeigen $Q_Y = [Y]$ für alle $Y \in N$. Dieser Beweis wird induktiv über die Länge der Zeichenketten erbracht. Zunächst $Q_Y \subseteq [Y]$. Denn sei $\vec{y} \in Q_Y$, so ist entweder $\vec{y} = \varepsilon$ und $Y \to \varepsilon \in R$, oder aber $\vec{y} = a \cdot \vec{x}$ mit $\vec{x} \in Q_X$ und $Y \to a \cdot \vec{x} \in R$. Im ersten Fall ist $Y \to \varepsilon \in R$, also $\varepsilon \in [Y]$. Im zweiten Fall haben wir $|\vec{x}| < |\vec{y}|$ und so nach Induktionsannahme $\vec{x} \in [X]$, also $X \Rightarrow_R \vec{x}$. Dann ist aber $Y \Rightarrow_R a \cdot \vec{x} = \vec{y}$, also $\vec{y} \in [Y]$. Dies zeigt die erste Inklusion. Nun zeigen wir noch $[Y] \subseteq Q_Y$. Dazu sei $Y \Rightarrow_R \vec{y}$. Dann ist entweder $\vec{y} = \varepsilon$ und so $Y \to \varepsilon \in R$, oder $\vec{y} = a \cdot \vec{x}$ für ein $\vec{x}$. In dem Fall ist $\vec{y} \in Q_Y$ nach Definition. Im zweiten Fall muß es ein X geben derart, daß $Y \to aX \in R$ und $X \Rightarrow_R \vec{x}$. Dann ist $|\vec{x}| < |\vec{y}|$ und deshalb nach Induktionsannahme $\vec{x} \in Q_X$ und so nach Definition von Q_Y schließlich $\vec{y} \in Q_Y$, was zu zeigen war.

Eine reguläre Sprache ist also Lösung eines einfachen Gleichungssystems. Umgekehrt kann jedes einfache Gleichungssystem sofort in eine reguläre Grammatik umgeschrieben werden, deren Sprache genau die Lösung des Gleichungssystems ist. Uns bleibt als Letztes zu zeigen, daß auch reguläre Terme nichts anderes beschreiben als reguläre Sprachen. Wir zeigen in der Tat allgemeiner, daß jedes echte Gleichungssystem mit ebenso viel Gleichungen wie Variablen eine reguläre Lösung besitzt für jede Variable. Dazu sei ein echtes Gleichungssystem vorgegeben mit den Gleichungen $X_j = \bigcup_{i<m_j} T_j^i$. Wir beginnen, indem wir X_0 aus sämtlichen Gleichungen eliminieren. Dazu sind zwei Fälle zu unterscheiden. (1.) X_0 taucht in der Gleichung $X_0 = \bigcup_{i<m_0} T_j^i$ nur links auf. Dann genügt einfaches Ersetzen von X_0 durch die rechte Seite. (2.) Die Gleichung ist von der Form $X_0 = C \cup D \cdot X_0$, C ein regulärer Term, der X_0 nicht enthält, D variablenfrei und $\varepsilon \notin L(D)$. Dann ist $X_0 = D^* \cdot C$ gemäß obigem Lemma. Nun taucht X_0 rechts nicht mehr auf, und wir können X_0 in den anderen Gleichungen wie in (1.) angegeben ersetzen. Das so erhaltene Gleichungssystem ist nicht mehr einfach. Wir können dennoch weitermachen und schrittweise alle Variablen eliminieren, bis wir die letzte Gleichung erreichen. Die Lösung für X_{n-1} enthält keine Variablen mehr und ist ein regulärer Term. Die Lösung kann man dann iterativ einsetzen und bekommt so reguläre Terme für X_{n-2}, X_{n-3}, ...

Als Beispiel diene folgendes Gleichungssystem.

$$
\begin{array}{llllll}
(I) & X_0 = & \mathtt{a} & \cup\, \mathtt{a}\cdot X_0 & \cup\, \mathtt{b}\cdot X_1 & \cup\, \mathtt{c}\cdot X_2 \\
 & X_1 = & & \cup\, \mathtt{c}\cdot X_0 & & \cup\, \mathtt{a}\cdot X_2 \\
 & X_2 = & \mathtt{b} & \cup\, \mathtt{a}\cdot X_0 & \cup\, \mathtt{b}\cdot X_1 & \\
\\
(II) & X_0 = & \mathtt{a}^+ & \cup\, \mathtt{a}^*\mathtt{b}\cdot X_1 & \cup\, \mathtt{a}^*\mathtt{c}\cdot X_2 & \\
 & X_1 = & \mathtt{ca}^+ & \cup\, \mathtt{ca}^*\mathtt{b}\cdot X_1 & \cup\, (\mathtt{ca}^*\mathtt{c}\cup\mathtt{a})\cdot X_2 & \\
 & X_2 = & \mathtt{b}\cup\mathtt{aa}^+ & \cup\, (\mathtt{a}^+\mathtt{b}\cup\mathtt{b})\cdot X_1 & \cup\, \mathtt{a}^*\mathtt{c}\cdot X_2 & \\
\\
(III) & X_1 = & (\mathtt{ca}^*\mathtt{b})^*\mathtt{ca}^+ & \cup\, (\mathtt{ca}^*\mathtt{b})^*(\mathtt{ca}^*\mathtt{c}\cup\mathtt{a})\cdot X_2 & & \\
 & X_2 = & (\mathtt{b}\cup\mathtt{aa}^+) & \cup\, [(\mathtt{a}^+\mathtt{b}\cup\mathtt{b})(\mathtt{ca}^*\mathtt{b})^*(\mathtt{ca}^*\mathtt{c}\cup\mathtt{a})\cup\mathtt{a}^*\mathtt{c}]\cdot X_2 & & \\
\\
(IV) & X_2 = & [(\mathtt{a}^+\mathtt{b}\cup\mathtt{b})(\mathtt{ca}^*\mathtt{b})^*(\mathtt{ca}^*\mathtt{c}\cup\mathtt{a})\cup\mathtt{a}^*\mathtt{c}]^*(\mathtt{b}\cup\mathtt{aa}^+) & & &
\end{array}
$$

Theorem 2.1.8 (Kleene) *Es sei S eine Sprache über A. Dann sind äquivalent:*

1. *S ist regulär.*
2. *$S = L(\mathfrak{A})$ für einen endlichen, deterministischen Automaten $\mathfrak{A}$ über A.*
3. *$S = L(R)$ für einen regulären Term R über A.*
4. *S ist die Lösung für X_0 eines einfachen Gleichungssystems über A mit Variablen X_i, $i < m$.*

Ferner existieren Algorithmen, welche zu einem Automaten $\mathfrak{A}$ einen regulären Ausdruck R berechnen mit $L(\mathfrak{A}) = L(R)$; welche zu einem regulären Ausdruck R ein einfaches Gleichungssystem Σ über $\vec{X}$ berechnen, dessen Lösung für eine gegebene Variable X_0 gerade $L(R)$ ist; und welche zu einem einfachen Gleichungssystem Σ über $\vec{X}$ einen Automaten $\mathfrak{A}$ berechnen derart, daß $\vec{X}$ die Zustandsmenge ist, und die Lösung für X_i gerade diejenige Menge von Zeichenketten ist, welche den Automaten von X_0 in X_i überführen. $\dashv$

Dies ist der wichtigste Satz über reguläre Sprachen. Wir wollen ein paar Folgerungen ziehen. Man beachte, daß ein endlicher Automat auch eine Turingmaschine ist. Und zwar ist eine Turingmaschine genau dann ein endlicher Automat, wenn sie nur nachs rechts gehen darf und Symbole nicht überschreiben kann. Deswegen ist das Erkennungsproblem für reguläre Sprachen in $DTIME(n)$ und $DSPACE(n)$. Dies gilt aber auch für das Analyseproblem, wie man sich leicht überlegt.

Korollar 2.1.9 *Das Erkennungsproblem und das Analyseproblem sind in $DTIME(n)$ und $DSPACE(n)$.*

Korollar 2.1.10 *Die Menge der regulären Sprachen über A ist abgeschlossen unter Schnitt und relativem Komplement. Ferner lassen sich zu gegebenen regulären Termen R und S Terme U und V bestimmen mit $L(U) = A^* - L(R)$ und $L(V) = L(R) \cap L(S)$.*

Beweis. Es genügt, diese Konstruktion für Automaten durchzuführen. Mit Hilfe von Satz 2.1.8 folgt, daß wir sie auch für die entsprechenden regulären Terme durchführen können. Sei $\mathfrak{A} = \langle Q, i_0, F, \delta\rangle$. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß $\mathfrak{A}$ deterministisch ist. Dann sei $\mathfrak{A}^- := \langle Q, i_0, Q - F, \delta\rangle$. Es gilt dann $L(\mathfrak{A}^-) = A^* - L(\mathfrak{A})$. Dies zeigt, daß wir zu gegebenem $\mathfrak{A}$ einen Automaten bauen können, der das Komplement von $L(\mathfrak{A})$ akzeptiert. Nun sei $\mathfrak{A}' = \langle Q', i_0', F', \delta'\rangle$. Setze $\mathfrak{A} \times \mathfrak{A}' = \langle Q \times Q', \langle i_0, i_0'\rangle, F \times F', \delta \times \delta'\rangle$, wobei

$$(\delta \times \delta')(\langle q, q'\rangle, a) := \{\langle r, r'\rangle : r \in \delta(q, a), r' \in \delta'(q', a)\}$$

Es ist leicht zu sehen, daß $L(\mathfrak{A} \times \mathfrak{A}') = L(\mathfrak{A}) \cap L(\mathfrak{A}')$ ist. $\dashv$

Der nächste Satz ist eine Übung.

Theorem 2.1.11 *Sind R und S reguläre Sprachen, so auch S/R und $R\backslash S$. Ebenso ist mit S auch S^T, $S^P := S/A^*$ und $S^S := A^*\backslash S$ regulär.*

Ferner kann man die folgende wichtige Folgerung ziehen.

Theorem 2.1.12 *Es seien $\mathfrak{A}$ und $\mathfrak{B}$ endliche Automaten. Dann ist das Problem '$L(\mathfrak{A}) = L(\mathfrak{B})$' entscheidbar.*

Beweis. Seien $\mathfrak{A}$ und $\mathfrak{B}$ gegeben. Aufgrund von Satz 2.1.8 kann man einen regulären Ausdruck R mit $L(R) = L(\mathfrak{A})$ sowie einen Ausdruck S mit $L(S) = L(\mathfrak{B})$ berechnen. Es ist $L(\mathfrak{A}) = L(\mathfrak{B})$ genau dann, wenn $L(R) = L(S)$ genau dann, wenn $(L(R) - L(S)) \cup (L(S) - L(R)) = \varnothing$. Nun läßt sich nach Korollar 2.1.10 ein regulärer Ausdruck U berechnen, für den $L(U) = (L(R) - L(S)) \cup (L(S) - L(R))$. Genau dann ist $L(\mathfrak{A}) = L(\mathfrak{B})$, wenn $L(U) = \varnothing$. Dies kann man nach dem Lemma 2.1.13 entscheiden. $\dashv$

Lemma 2.1.13 *Sei R ein regulärer Ausdruck. Es ist entscheidbar, ob $L(R) = \varnothing$.*

Beweis. Induktion über den Aufbau von R. Ist $R = \varepsilon$ oder $R = a$, so $L(R) \neq \varnothing$. Ist $R = 0$, so $L(R) = \varnothing$. Seien nun die Probleme '$L(R) = \varnothing$' und '$L(S) = \varnothing$' entscheidbar. Es ist genau dann $L(R \cup S) = \varnothing$, wenn $L(R) = \varnothing$ und $L(S) = \varnothing$; dies ist entscheidbar. Ferner ist genau dann $L(R \cdot S) = \varnothing$, wenn $L(R) = \varnothing$ oder $L(S) = \varnothing$. Dies ist entscheidbar. Endlich ist $L(R^*) = \varnothing$ genau dann, wenn $L(R) = \varnothing$. Auch dies ist entscheidbar. $\dashv$

Zum Schluß beweisen wir noch einen wichtigen Satz.

Theorem 2.1.14 *Es sei S eine kontextfreie und R eine reguläre Sprache. Dann ist $S \cap R$ auch kontextfrei.*

Beweis. Es sei $G = \langle S, N, A, R\rangle$ eine kontextfreie Grammatik mit $L(G) = S$ und $\mathfrak{A} = \langle n, 0, F, \delta\rangle$ ein aus n Zuständen bestehender deterministischer Automat mit $L(\mathfrak{A}) = R$. Wir können annehmen, daß Regeln in G die Form $X \to a$ oder $X \to \vec{Y}$ haben. Wir definieren nun neue Nichtterminalsymbole, welche alle die Form ${}^iX^j$ haben, wo $i, j < n$ und $X \in N$. Die Interpretation ist wie folgt. X steht für die Menge aller Zeichenketten $\vec{\alpha} \in A^*$ mit $X \vdash_G \vec{\alpha}$. ${}^iX^j$ steht nun für die Menge aller $\vec{\alpha}$ mit $X \vdash_G \vec{\alpha}$ und $\delta(i, \vec{\alpha}) = j$. Als neues Startsymbole fungieren ${}^0S^j$ für alle $j \in F$. Wie wir wissen, dürfen wir eine Menge von Startsymbolen annehmen. Eine Regel der Form $X \to Y_0Y_1 \dots Y_{k-1}$ wird nun ersetzt durch die Menge aller Regeln

$$ {}^iX^j \to {}^iY_0{}^{i_0}\ {}^{i_0}Y_1{}^{i_1}\ \dots\ {}^{i_{k-2}}Y_{k-1}{}^j. $$

Schließlich nehmen wir noch alle Regeln ${}^iX^j \to a$ hinzu, wo $\delta(i, a) = j$ ist. Dies definiert die Grammatik G_r. Wir zeigen: $\vdash_{G^r} \vec{x}$ genau dann, wenn $\vdash_G \vec{x}$ und $\vec{x} \in L(\mathfrak{A})$. ($\Rightarrow$) Sei $\mathfrak{B}$ ein G^r–Baum mitZeichenkette $\vec{x}$. Die Abbildung ${}^iX^j \mapsto X$ macht $\mathfrak{B}$ zu einem G–Baum. Also ist $\vec{x} \in L(G)$. Ferner beweist man leicht, daß $\delta(0, x_0x_1 \dots x_j) = k_j$ ist, wo ${}^{k_{j-1}}X^{k_j}$ der x_j dominierende Knoten ist. Ferner: ist $|\vec{x}| = n$, so ist ${}^0S^{k_n}$ der Startknoten und nach Voraussetzung $k_n \in F$. Also gilt $\delta(\vec{x}, 0) \in F$ und so $\vec{x} \in L(\mathfrak{A})$. ($\Leftarrow$) Sei $\vec{x} \in L(G)$ und $\vec{x} \in L(\mathfrak{A})$. Wir zeigen $\vec{x} \in L(G^r)$. Wir nehmen dazu einen G–Baum $\mathfrak{B}$ für $\vec{x}$. Wir zeigen nun, daß wir die G–Nichtterminalsymbole in $\mathfrak{B}$ so durch G^r–Symbole ersetzen werden können, daß ein G_r–Baum entsteht. Der Beweis erfolgt durch Induktion über die Höhe der Knoten. Zunächst beginnen wir mit Knoten der Höhe 1. Sei $\vec{x} = \prod_{i<n} x_i$; und es sei X_i das Nichtterminalsymbol oberhalb von x_i. Ferner sei $\delta(0, \prod_{i<j} x_i) = j_i$. Es ist $p_0 = 0$ und $p_n \in F$. Wir ersetzen X_i durch ${}^{p_i}X^{p_{i+1}}$. Wir sagen, daß zwei Knoten x und y **verbinden**, falls sie adjazent sind und für die Marke ${}^iX^j$ von x sowie ${}^kY^\ell$ von y gilt $j = k$. Sei x ein Knoten mit Marke X und Mutter von Knoten mit Marken $Y_0Y_1 \dots Y_{n-1}$ in G. Wir nehmen an, unterhalb von x hätten alle Knoten bereits G^r–Marken derart bekommen, daß je zwei adjazente Knoten verbinden. Dann existiert eine Regel in G_r derart, daß X nun seinerseits mit Indizes versehen werden kann, nämlich den linken Index von Y_0

links, und den rechten Index von Y_{n-1} rechts. Damit bleibt die Verbindungseigenschaft erhalten, wie man leicht nachrechnet. Außerdem hat der linke äußere Knoten stets den linken Index 0 sowie der rechte äußere Knoten den rechten Index p_n. So läßt sich fortfahren, bis die Wurzel erreicht ist. Diese trägt dann die Marke ${}^0S^{p_n}$, und somit haben wir einen G_r–Baum. $\dashv$

Übung 30. Beweisen Sie Theorem 2.1.11.

Übung 31. Zeigen Sie, daß eine Sprache genau dann regulär ist, wenn sie von einer Grammatik erzeugt werden kann mit Regeln von der Form $X \to Y$, $X \to Ya$, $X \to a$ und $X \to \varepsilon$. Eine solche Grammatik heißt **linksregulär**, im Gegensatz zu den Grammatiken vom Typ 3, welche wir auch **rechtsregulär** nennen wollen. Zeigen Sie, daß man auch Regeln von der Form $X \to \vec{x}$ und $X \to Y\vec{x}$ zulassen kann.

Übung 32. Zeigen Sie, es gibt eine Grammatik, welche Regeln der Form $X \to a$, $X \to aY$ und $X \to Ya$ hat und eine nichtreguläre Sprache erzeugt. Dies bedeutet, daß man nur einen Typus von Regeln zulassen darf, entweder rechtsreguläre oder linksreguläre.

Übung 33. Man zeige, daß mit S und T auch die Sprachen S/T und $T\backslash S$ regulär sind.

Übung 34. Es sei S eine Sprache über A. Definiere eine Äquivalenzrelation $\sim_S$ über A^* wie folgt. Es sei $\vec{x} \sim_S \vec{y}$ genau dann, wenn für alle $\vec{z} \in A^*$ gilt $\vec{x}\cdot\vec{z} \in S \Leftrightarrow \vec{y}\cdot\vec{z} \in S$. S habe *endlichen Index*, wenn es nur endlich viele Äquivalenzklassen bezüglich $\sim_S$ gibt. Zeigen Sie, daß S genau dann regulär ist, wenn $\sim_S$ endlichen Index hat.

Übung 35. Zeigen Sie, daß die Sprache $\{\mathtt{a}^n\mathtt{b}^n : n \in \omega\}$ keinen endlichen Index hat. Sie ist folglich nicht regulär.

Übung 36. Zeigen Sie, daß der Schnitt einer kontextsensitiven Sprache mit einer regulären Sprache wieder eine kontextsensitive Sprache ist.

2.2 Normalformen für kontextfreie Grammatiken

In den verbleibenden Abschnitten dieses Kapitels werden wir uns mit kontextfreien Grammatiken und Sprachen auseinandersetzen. Angesichts der Fülle des Wissens über diese Sprachen können wir allerdings nur einen groben Überblick geben. Zunächst einmal wollen wir uns in diesem Abschnitt mit Normalformen befassen. Es gibt sehr viele verschiedene Normalformen von kontextfreien Grammatiken; jede dient einem verschiedenen Zweck. Man muß außerdem beachten, daß die Transformation einer Grammatik in eine Normalform gewisse Eigenschaften zerstören kann. Wie einfach die Normalform ist, hängt also wesentlich von den Eigenschaften ab, welche man von der Transformation erhalten wissen will. Kommt es z. B. nur darauf an, daß die erzeugte Sprache gleich bleibt, so kann man immer die sogenannte

Chomsky–Normalform erreichen. Will man aber die erzeugten Konstituentenstrukturen unangetastet lassen, so kann man nur die *Standardform* erreichen.

Bevor wir uns der Reduktion von Grammatiken zuwenden, wollen wir uns mit der Wechselbeziehung zwischen Ableitungen, Bäumen, und Regelmengen befassen. Aus Sicherheitsgründen nehmen wir an, daß jedes Symbol in mindestens einem Baum auftaucht, also die Grammatiken schlank sind im Sinne von Definition 2.2.3. Aus den Überlegungen in Abschnitt 1.6 folgt, daß für zwei kontextfreie Grammatiken $G = \langle S, N, A, R\rangle$ und $G' = \langle S', N', A, R'\rangle$ genau dann $L_B(G) = L_B(G')$ gilt, wenn $\mathbf{der}(G) = \mathbf{der}(G')$. Ebenso sieht man, daß genau dann für alle $X \in N \cup N'$ gilt $der(G, X) = der(H, X)$, wenn $R = R'$. Es sei nun $G = \langle S, N, A, R\rangle$ gegeben und eine Sequenz $\Gamma = \langle \vec{\alpha}_i : i < n\rangle$. Um zu testen, ob Γ eine korrekte G–Kettenfolge ist, muß man für jedes $i < n - 1$ prüfen, ob $\vec{\alpha}_{i+1}$ aus $\vec{\alpha}_i$ durch einmalige Anwendung einer Regel hervorgeht. Dazu müssen wir ein Nichtterminalsymbol aus $\vec{\alpha}_i$ auswählen und eine Regel anwenden, und dann prüfen, ob die so erhaltene Zeichenkette gleich $\vec{\alpha}_{i+1}$ ist. Diese Prüfung verbraucht $a_G \times |\vec{\alpha}_i|$ Zeitschritte, wo a_G eine Konstante ist, welche von G abhängt. Man braucht also $\sum_{i<n} a_G|\vec{\alpha}_i|$ Zeitschritte für die gesamte Ableitung. Dieses kann nach oben durch $a_G \times n \times |\vec{\alpha}_{n-1}|$ abgeschätzt werden, und wenn G strikt expandierend ist auch durch $a_G \times |\vec{\alpha}_{n-1}|^2$. Man kann zeigen, daß es Grammatiken gibt, bei denen man auch nicht weniger Zeit benötigt. Um von einem geordneten Baum zu prüfen, ob er von γG erzeugt wird, benötigt man weniger Zeit. Man muß nämlich an jedem Knoten x prüfen, ob der lokale Baum bei x einer Regel von G genügt. Dies braucht eine gewisse, nur von G abhängige Zeit. Insgesamt hängt die Zeit nur linear von der Größe des Baumes ab.

Zwischen Bäumen und Ableitungen besteht eine enge Beziehung. Zunächst einmal läßt sich aus jeder Ableitung ein Strukturbaum ableiten, indem zu jeder Anwendung einer Regel ρ in G die analoge Regel ρ^γ auf den Baum angewendet wird. Umgekehrt aber bestimmt ein Baum die Ableitung nicht in eindeutiger Weise. Sei nämlich $\mathfrak{B}$ ein Baum. Wir nennen $\lhd \subseteq B^2$ eine **Linearisierung**, falls $\lhd$ ein irreflexive, lineare Ordnung ist und aus $x > y$ folgt $x \lhd y$. Zu einer gegebenen Linearisierung können wir nun eine Ableitung wie folgt gewinnen. Wir beginnen mit dem bezüglich $\lhd$ kleinsten Element. Dies ist die Wurzel, wie man leicht sieht. Die Wurzel trägt die Marke S. Induktiv konstruieren wir daraus Schnitte $\vec{\alpha}_i$ durch $\mathfrak{B}$ derart, daß die Folge $\langle \vec{\alpha}_i : i < n\rangle$ eine Ableitung von der assoziierten Zeichenkette ist. Der Anfang ist gemacht, denn $\vec{\alpha}_0 = S$. Nun sei $\vec{\alpha}_i$ schon konstruiert und nicht gleich der assoziierte Zeichenkette von $\mathfrak{B}$. Dann existiert ein nichtterminales y in $\vec{\alpha}_i$. Wir nehmen das bezüglich $\lhd$ kleinste solche y. Seine Marke sei Y. Da der Baum ein G–Baum ist, ist der lokale Baum von der Form $Y \to \vec{\beta}$ für ein gewisses $\vec{\beta}$. In $\vec{\alpha}_i$ definiert y eindeutig ein Vorkommen von Y. $\vec{\alpha}_{i+1}$ ist dann das Resultat der Ersetzung von diesem Vorkommen von Y durch $\vec{\beta}$. Damit ist die neue Zeichenkette gewiß das Resultat der Anwendung einer Regel aus G, wie gewünscht. Man kann diese Prozedur auch umkehren, und so zu einer Ableitung eine Linearisierung des Baumes bestimmen.

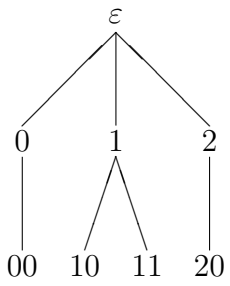
Allerdings ist die Ordnung in Bezug auf die Blätter des Baumes nicht durch die Ableitung eindeutig bestimmt.

Theorem 2.2.1 *Es sei G kontextfrei, und $\mathfrak{B} \in L_B(G)$. Ferner sei $\lhd$ eine Linearisierung von $\mathfrak{B}$. Dann bestimmt $\lhd$ eine G–Ableitung $\boldsymbol{der}(\lhd)$ der zu $\mathfrak{B}$ assoziierten Zeichenkette. Ist dann $\blacktriangleleft$ eine Linearisierung von $\mathfrak{B}$, so ist $\boldsymbol{der}(\blacktriangleleft) = \boldsymbol{der}(\lhd)$ genau dann, wenn $\blacktriangleleft$ und $\lhd$ auf den inneren Knoten von $\mathfrak{B}$ übereinstimmen.* $\dashv$

Linearisierungen können auch als Suchstrategien auf dem Baum betrachtet werden. Wir wollen ein paar davon vorstellen. (Näheres findet man zum Beispiel im Buch von Aigner ([2]).) Die erste ist die in Spezialfall einer sogenannten **depth–first** Suche, die wir **linksseitige Linearisierung** nennen wollen. Hier wird der Baum wie folgt linearisiert: $x \lhd y$ genau dann, wenn $x > y$ oder $x \sqsubset y$. Zu jedem Baum gibt es genau eine linksseitige Linearisierung. Wir wollen die Tatsache, daß es eine linksseitige Ableitung von $\vec{\alpha}$ aus X gibt, mit $X \vdash_G^{\ell} \vec{\alpha}$ bezeichnen. Man kann dies wie folgt verallgemeinern. Es sei $\blacktriangleleft$ eine lineare Ordnung, die auf den lokalen Bäumen uniform ist. Das heißt, falls $\mathfrak{B}$ und $\mathfrak{C}$ lokale Teilbäume sind, die zu der gleichen Regel ρ gehören, dann sei $\mathfrak{B}$ genau wie $\mathfrak{C}$ durch $\lhd$ linearisiert. Im Falle der linksseitigen Ableitung ist die Ordnung gerade die durch $\sqsubset$ gegebene. Man überlege sich, daß jede Linearisierung der lokalen Teilbäume eine Linearisierung des gesamten Baumes induziert — aber nicht umgekehrt. Dann bezeichnet $X \vdash_G^{\blacktriangleleft} \vec{\alpha}$ die Tatsache, daß es eine Ableitung von $\vec{\alpha}$ aus X bestimmt durch $\blacktriangleleft$ gibt. Es heiße nun π eine **Priorisierung** für $G = \langle S, N, A, R\rangle$, falls π eine Linearisierung auf jedem Baum $\mathfrak{H}_\rho$, $\rho \in R$, definiert. Da die Wurzel immer das erste Element einer Linearisierung sein muß, genügt bei der Priorisierung die Angabe, wie die Töchter durch $\blacktriangleleft$ geordnet sein sollen. Wir schreiben $X \vdash_G^{\pi} \vec{\alpha}$, falls $X \vdash_G^{\blacktriangleleft} \vec{\alpha}$ für die durch π definierte Linearisierung $\blacktriangleleft$.

Proposition 2.2.2 *Es sei π eine Priorisierung. Dann gilt $X \vdash_G^{\pi} \vec{x}$ genau dann, wenn $X \vdash_G \vec{x}$.*

Eine andere Strategie ist die **breadth–first** Suche. Diese Strategie sucht den Baum schichtweise ab. Es sei S_n die Menge aller Knoten x mit $t(x) = n$. Jede Schicht ist linear durch $\sqsubset$ geordnet. Die breadth–first Suche geht mit steigendem i die Schicht S_i in Richtung der Ordnung $\sqsubset$ durch. Der Unterschied zwischen depth–first and breadth–first kann auf Baumbereichen augenfällig gemacht werden. Die depth–first Suche geht den Baumbereich in der lexikographischen Ordnung durch, die breadth–first Suche in der numerischen Ordnung. Zum Beispiel sei der folgende Baumbereich gegeben.



Die depth–first Linearisierung ist

$$\varepsilon, 0, 00, 1, 10, 11, 2, 20$$

Die breadth–first Linearisierung ist jedoch

$$\varepsilon, 0, 1, 2, 00, 10, 11, 20$$

Man beachte, daß man mit diesen Linearisierungen den Baumbereich ω^* nicht aufzählen kann. Man bekommt nämlich in der depth–first Linearisierung

$$\varepsilon, 0, 00, 000, 0000, \ldots$$

In der breadth–first Linearisierung bekommt man dagegen

$$\varepsilon, 0, 1, 2, 3, \ldots$$

Da aber ω^* abzählbar ist, gibt es auch eine Linearisierung, nur ist sie wesentlich komplizierter.

Die erste Reduktion, welche wir betrachten, ist die Elimination von überflüssigen Symbolen und Regeln. Sei $G = \langle S, A, N, R\rangle$ eine kontextfreie Grammatik. Es heiße $X \in N$ **erreichbar**, falls $G \vdash \vec{\alpha} \cdot X \cdot \vec{\beta}$ für gewisse $\vec{\alpha}$ und $\vec{\beta}$. X heißt **vollendbar**, falls es ein $\vec{x}$ gibt mit $X \Rightarrow_R \vec{x}$.

S	→	AB	A	→	CB
B	→	AB	A	→	x
D	→	Ay	C	→	y

In der obenstehenden Grammatik sind A, C und D vollendbar. Es sind S, A, B und C erreichbar. Da S, das Startsymbol, nicht vollendbar ist, ist kein Symbol sowohl erreichbar als auch vollendbar. Die Grammatik erzeugt keine terminalen Zeichenketten.

Es sei N' die Menge der nichtterminalen Symbole, welche sowohl erreichbar als auch vollendbar sind. Wir nehmen an, daß S auch vollendbar ist. Falls nicht, ist

$L(G) = \varnothing$. In diesem Fall setze $N' := \{\mathtt{S}\}$ und $R' := \varnothing$. Es sei nun R' die Einschränkung von R auf Symbole aus A oder N'. Dies definiert $G' = \langle \mathtt{S}, N', A, R' \rangle$. Induktiv kann zeigen, daß $G \vdash \vec{\alpha}$ genau dann der Fall ist, wenn $G' \vdash \vec{\alpha}$. Zusätzlich kann man zeigen, daß jede Ableitung in G eine analoge G–Ableitung in G' hat unter Benutzung der gleichen Regeln und umgekehrt.

Definition 2.2.3 *Eine kontextfreie Grammatik heißt **schlank**, falls entweder $L(G) = \varnothing$, und G keine Nichtterminalsymbole außer dem Startsymbol hat und keine Regeln; oder $L(G) \neq \varnothing$ und jedes Nichtterminalsymbol ist sowohl erreichbar wie auch vollendbar.*

Für schlanke Grammatiken gilt offensichtlich, daß ihre Regelmengen genau dann gleich sind, wenn ihre Derivationen gleich sind.

Proposition 2.2.4 *Es seien G und H schlank. Dann ist $G = H$ genau dann, wenn $der(G) = der(H)$.*

Proposition 2.2.5 *Zu jeder kontextfreien Grammatik $G = \langle \mathtt{S}, N, A, R \rangle$ läßt sich eine schlanke kontextfreie Grammatik $G^s = \langle \mathtt{S}, N^s, A, R^s \rangle$ konstruieren mit $N^s \subseteq N$, welche die gleichen Ableitungen hat wie G. Daraus folgt, daß auch $L_B(G^s) = L_B(G)$.* $\dashv$

Als nächstes soll uns die Frage nach dem Beitrag der Nichtterminalsymbole beschäftigen. Da diese Symbole in der erzeugten Sprache $L(G)$ nicht auftauchen, ist ihr Name an sich unbedeutend. Um dies zu präzisieren, führen wir den Begriff einer *Regelsimulation* ein. Es seien G und G' Gramatiken mit Mengen N und N' von Nichtterminalsymbolen. Wir betrachten eine Relation $\sim \subseteq N \times N'$. Diese läßt sich zu einer Relation $\approx \subseteq (N \cup A)^* \times (N' \cup A)^*$ ausdehnen, indem man genau dann $\vec{\alpha} \approx \vec{\beta}$ setzt, wenn $\vec{\alpha}$ und $\vec{\beta}$ die gleiche Länge haben und $\alpha_i \sim \beta_i$ für alle i. Eine Relation $\sim \subseteq N \times N'$ wollen wir eine **Vorwärts–Regelsimulation** oder **R–Simulation** nennen, falls (0.) $\mathtt{S} \sim \mathtt{S}'$, (1.) falls $X \to \vec{\alpha} \in R$ und $X \sim Y$, so existiert ein $\vec{\beta}$ mit $\vec{\alpha} \approx \vec{\beta}$ und $Y \to \vec{\beta} \in R'$, und (2.) falls $Y \to \vec{\beta} \in R'$ und $X \sim Y$, so existiert ein $\vec{\alpha}$ mit $\vec{\alpha} \approx \vec{\beta}$ und $X \to \vec{\alpha} \in R$. Eine **Rückwärts–Simulation** ist wie folgt definiert. (0.) Aus $\mathtt{S} \sim X$ folgt $X = \mathtt{S}'$ und aus $Y \sim \mathtt{S}'$ folgt $Y = \mathtt{S}$, (1.) falls $X \to \vec{\alpha} \in R$ und $\vec{\alpha} \approx \vec{\beta}$, so ist $Y \to \vec{\beta} \in R'$ für ein Y mit $X \sim Y$, (2.) falls $Y \to \vec{\beta} \in R'$ und $\vec{\beta} \approx \vec{\alpha}$, so ist $X \to \vec{\alpha} \in R$ für ein X mit $X \sim Y$.

Wir geben ein Beispiel für eine Vorwärtssimulation. Es seien G und G' die fol-

genden Grammatiken.

S	→	ASB | AB	S	→	ATB | ASC | AC
A	→	b	T	→	ATC | AC
B	→	b	A	→	a
			B	→	b
			C	→	b

Das Startsymbol sei jeweils S. Dann ist die folgende Relation eine R–Simulation.

$$\sim := \{\langle \mathtt{A}, \mathtt{A}\rangle, \langle \mathtt{B}, \mathtt{B}\rangle, \langle \mathtt{S}, \mathtt{S}\rangle, \langle \mathtt{B}, \mathtt{C}\rangle, \langle \mathtt{S}, \mathtt{T}\rangle\}$$

Mit $\sim$ ist auch die konverse Relation $\sim^{\smile} = \{\langle Y, X\rangle : \langle X, Y\rangle \in \sim\}$ eine R–Simulation. Falls nun $\sim$ eine R–Simulation ist und $\langle \vec{\alpha}_i : i < n+1\rangle$ eine G–Ableitung, so existiert eine G'–Ableitung $\langle \vec{\beta}_i : i < n+1\rangle$ mit $\vec{\alpha}_i \approx \vec{\beta}_i$ für alle $i < n+1$. Man kann sogar genauer sagen, daß wenn $\langle \vec{\alpha}_i, C, \vec{\alpha}_{i+1}\rangle$ die Anwendung einer Regel aus G ist mit $C = \langle \kappa_1, \kappa_2\rangle$, so existiert ein Kontext $D = \langle \lambda_1, \lambda_2\rangle$ derart, daß $\langle \vec{\beta}_i, D, \vec{\beta}_{i+1}\rangle$ die Anwendung einer Regel aus G' ist. Auf diese Weise erhalten wir auch die Tatsache, daß zu jedem $\mathfrak{B} = \langle B, <, \sqsubset, \ell\rangle \in L_B(G)$ ein $\mathfrak{C} = \langle B, <, \sqsubset, \mu\rangle \in L_B(G')$ existiert mit $\ell(x) = \mu(x)$ für alle Blätter und $\ell(x) \sim \mu(x)$ für alle Nichtblätter. Man kann nun analog zu einer Regelsimulation eine Derivationssimulation definieren, indem man verlangt, daß zu jeder G–Ableitung Γ eine unter Simulation äquivalente G'–Ableitung Δ existiere.

Proposition 2.2.6 *Es seien $G_1 = \langle \mathtt{S}_1, N_1, A, R_1\rangle$ und $G_2 = \langle \mathtt{S}_2, N_2, A, R_2\rangle$ schlanke kontextfreie Grammatiken und $\sim \subseteq N_1 \times N_2$ eine R–Simulation. Dann existiert zu jeder G_1–Ableitung $\langle \vec{\alpha}_i : i < n\rangle$ eine G_2–Ableitung $\langle \vec{\beta}_i : i < n\rangle$ mit $\vec{\alpha}_i \approx \vec{\beta}_i$, $i < n$.* $\dashv$

Wir wollen zwei Spezialfälle von Simulationen betrachten. Zwei Grammatiken G und G' heißen **äquivalent**, falls es eine Bijektion $b : N \cup A \to N' \cup A$ derart gibt, daß $b(x) = x$ für alle $x \in A$, $b(S) = S'$ und $\overline{b}$ eine Bijektion vermittelt zwischen G–Ableitungen und G'–Ableitungen. Dieser Begriff ist schärfer als derjenige, welcher fordert, daß $\overline{b}$ eine Bijektionen zwischen den Regelmengen stifte. Denn es kann vorkommen, daß gewisse Regeln nie in einer Ableitung eingesetzt werden können. Wir können von zwei kontextfreien Grammatiken ganz einfach entscheiden, ob sie äquivalent sind. Zunächst nämlich bringen wir beide in eine Form, in der alle Regeln in mindestens einer Ableitung gebraucht werden, indem wir nicht vollendbare und nicht erreichbare Symbole eliminieren. Solche Grammatiken sind dann äquivalent, wenn es eine Bijektion b gibt, welche die Regelmengen ineinander überführt. Die Existenz einer solchen Bijektion ist leicht zu prüfen.

Der eben vorgeschlagene Äquivalenzbegriff ist allerdings zu stark in dem folgenden Sinne. Es kann nichtterminale Symbole geben, die sich in nichts voneinander

unterscheiden lassen. Wir sagen, G sei **reduzierbar** auf G', falls es eine surjektive Abbildung $b : N \cup A \to N' \cup A'$ gibt, derart, daß $b(S) = S'$, $b(x) = x$ für alle $x \in A$ und $\overline{b}$ jede G–Ableitung auf eine G'–Ableitung abbildet, während jedes Urbild unter $\overline{b}$ einer G'–Ableitung eine G–Ableitung ist. (Wir verlangen jedoch nicht, daß das Urbild des Startsymbols aus G' notwendig eindeutig ist; lediglich verlangen wir, daß das Startsymbol aus G *ein* Urbild des Startsymbols aus G' sein muß.)

Definition 2.2.7 *G' heißt **reduziert**, wenn jede Grammatik G', auf die G reduzierbar ist, schon zu G äquivalent ist.*

Zu G kann man effektiv eine reduzierte Grammatik konstruieren, auf die G reduzierbar ist. Wir merken an, daß in unserem obigen Beispiel G' nicht auf G reduzierbar ist. Denn obschon $\sim^\smile$ eine Funktion ist (mit $\mathtt{A} \mapsto \mathtt{A}, \mathtt{B} \mapsto \mathtt{B}, \mathtt{C} \mapsto \mathtt{B}, \mathtt{S} \mapsto \mathtt{S}, \mathtt{T} \mapsto \mathtt{S}$) und ASB aus S in einem Schritt herleitbar ist, ist ATB nicht aus S herleitbar. Gegeben G und die Funktion $\sim^\smile$, wird die folgende Grammatik auf G reduziert.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{ASB} \mid \mathtt{ATB} \mid \mathtt{ASC} \mid \mathtt{ATC} \mid \mathtt{AB} \mid \mathtt{AC} \\ \mathtt{T} & \to & \mathtt{ASB} \mid \mathtt{ATB} \mid \mathtt{ASC} \mid \mathtt{ATC} \mid \mathtt{AB} \mid \mathtt{AC} \\ \mathtt{A} & \to & \mathtt{a} \\ \mathtt{B} & \to & \mathtt{b} \\ \mathtt{C} & \to & \mathtt{b} \end{array}$$

Nun sei G eine kontextfreie Grammatik. Wir nehmen zu A noch zwei Symbole hinzu, nämlich (und). Diese sollen nicht schon in A enthalten sein. Anschließend ersetzen wir jede Regel $X \to \vec{\alpha}$ durch die Regel $X \to \mathtt{(} \cdot \vec{\alpha} \cdot \mathtt{)}$. Die so erhaltene Grammatik bezeichnen wir mit G^b.

$$\begin{array}{lcl} & & G \\ \mathtt{S} & \to & \mathtt{AS} \mid \mathtt{SB} \mid \mathtt{AB} \\ \mathtt{A} & \to & \mathtt{a} \\ \mathtt{B} & \to & \mathtt{b} \end{array} \qquad \begin{array}{lcl} & & G^b \\ \mathtt{S} & \to & \mathtt{(AS)} \mid \mathtt{(SB)} \mid \mathtt{(AB)} \\ \mathtt{A} & \to & \mathtt{(a)} \\ \mathtt{B} & \to & \mathtt{(b)} \end{array}$$

G erzeugt die Sprache $\mathtt{a}^+\mathtt{b}^+$. Die Kette aabb hat mehrere Ableitungen, welche unterschiedlichen Bäumen entsprechen, nämlich zum Beispiel $\langle \mathtt{S}, \mathtt{AS}, \mathtt{ASB}, \mathtt{AABB}, \ldots, \mathtt{aabb} \rangle$ und $\langle \mathtt{S}, \mathtt{SB}, \mathtt{ASB}, \mathtt{AABB}, \ldots, \mathtt{aabb} \rangle$. Betrachten wir die analogen Ableitungen in G^b, so bekommen wir die Zeichenketten

$$\mathtt{((a)(((a)(b))(b))),} \qquad \mathtt{(((a)((a)(b)))(b))}$$

Diese sind offensichtlich verschieden. Definiere einen Homomorphismus $\overline{e}$ durch $\overline{e}(a) := a$, falls $a \in A$, $\overline{e} : \mathtt{)} \mapsto \varepsilon$ und $\overline{e} : \mathtt{)} \mapsto \varepsilon$. Dann ist unschwer zu sehen, daß

$$L(G) = \overline{e}[L(G^b)]$$

Nun betrachte man die Baummenge zu $L(G)$ und vergesse dabei die Marken aller Knoten, welche keine Blätter sind. Die so erhaltenen Strukturen wollen wir eine **Klammerungsanalyse** der assoziierten Zeichenkette nennen. Der Grund ist, daß die Klammerungsanalysen in eineindeutiger Beziehung stehen zu den Zeichenketten, welche $L(G^b)$ erzeugt. Wir wollen uns nun fragen, ob für je zwei Grammatiken G und H entscheidbar ist, ob sie die gleichen Klammerungsanalysen erzeugen. Dazu überlegen wir zunächst, wie das Analogon einer Derivation von G in G^b aussieht. Es sei $\vec{\gamma}X\vec{\eta}$ in G ableitbar, und die entsprechende G^b Zeichenkette dieser Ableitung in $\vec{\gamma}^b X\vec{\eta}^b$. Im nächsten Schritt werde X durch α ersetzt. Dann entsteht $\vec{\gamma}\vec{\alpha}\vec{\eta}$, und in G^b die Kette $\vec{\gamma}^b\texttt{(}\vec{\alpha}\texttt{)}\vec{\delta}^b$. Haben wir eine R–Simulation nach H, so ist diese auch eine R–Simulation von G^b nach H^b, wie man leicht bestätigt. Daraus folgt: falls es eine R–Simulation von G nach H gibt, so ist nicht nur $L(G) = L(H)$, sondern auch $L(G^b) = L(H^b)$.

Theorem 2.2.8 *Es gilt $L(G^b) = L(H^b)$, falls es eine R–Simulation von G nach H gibt.*

Die Klammerungsanalysen sind allerdings für viele Zwecke zu strikt. Zunächst ist es nicht notwendig, ein Einzelzeichen in Klammern zu setzen. Ferner macht es keinen Sinn, zwischen $\texttt{((}\vec{x}\texttt{))}$ und $\texttt{(}\vec{x}\texttt{)}$ zu unterscheiden, da beide Ketten nur aussagen, daß $\vec{x}$ eine Konstituente ist. Wir wollen also stattdessen Konstituentenanalysen vornehmen. Diese sind Paare $\langle\vec{x}, \mathfrak{C}\rangle$, bei der $\vec{x}$ eine Zeichenkette, und $\mathfrak{C}$ eine erschöpfende Konstituentenstruktur definiert über $\vec{x}$. Wir wollen mit $L_c(G)$ die Menge der von G erzeugten Konstituentenanalysen bezeichnen. Um von den Klammerungsanalysen zu den Konstituentenanalysen überzugehen, müssen wir lediglich die einstelligen Regeln eliminieren. Dies kann man so erreichen. Zunächst ersetzt man jede Regel $\rho = Y \to \vec{\alpha}$ durch die Menge ρ^2 aller Regeln, in der jedes Y in $\vec{\alpha}$ durch ein Z ersetzt wird mit $Z \Rightarrow Y^+$. $R^2 := \bigcup\langle\rho^2 : \rho \in R\rangle$. $R^>$ sei das Resultat der Streichung von allen Regeln der Form $X \to Y$ aus R^2. Endlich sei $G^> := \langle \mathtt{S}, N, A, R^>\rangle$. Jede Regel ist strikt produktiv, und es ist $L_c(G) = L_c(G^>)$. (Falls nötig, wollen wir zusätzlich annehmen, daß $G^>$ schlank sei.)

Definition 2.2.9 *Eine kontextfreie Grammatik ist in **Standardform**, falls jede Regel $\neq \mathtt{S} \to \varepsilon$ die Form $X \to \vec{Y}$ mit $|\vec{Y}| > 1$ oder die Form $X \to a$ hat. Eine Grammatik ist in **2–Standardform** oder **Chomsky–Normalform**, falls jede Regel von der Form $\mathtt{S} \to \varepsilon$, $X \to Y_0Y_1$ oder $X \to a$ ist.*

(Man beachte, daß nach unseren Konventionen eine kontextfreie Grammatik in Standardform nur die Regel $X \to \varepsilon$ für $X = \mathtt{S}$ anthält, und dies auch nur dann, wenn $\mathtt{S}$ nicht die rechte Seite einer Regel ist.) Wir haben bereits bewiesen, daß Folgendes gilt.

Theorem 2.2.10 *Zu jeder kontextfreien Grammatik G läßt sich eine schlanke kontextfreie Grammatik G^n in Standardform konstruieren, welche die gleichen Konstituentenanalysen erzeugt wie G.*

Theorem 2.2.11 *Zu jeder kontextfreien Grammatik G läßt sich eine schlanke kontextfreie Grammatik G^c in Chomsky–Normalform konstruieren mit $L(G^c) = L(G)$.*

Beweis. Zunächst bringe man G in Standardform $G^n = \langle \mathtt{S}, A, N^n, R^n \rangle$. Sei $\rho = X \to Y_0 Y_1 \dots Y_{n-1}$ eine Regel mit $n > 2$. Es seien $Z_0^\rho, Z_1^\rho, \dots, Z_{n-2}^\rho$ neue Nichtterminalsymbole. Ersetze ρ durch die Regeln

$$\rho_0^c := X \to Y_0 Z_0^\rho, \qquad \rho_1^c := Z_0^\rho \to Y_1 Z_1^\rho, \qquad \dots, \rho_{n-2}^c := Z_{n-3}^\rho \to Y_{n-2} Y_{n-1}$$

Jede Ableitung in G einer Kette $\vec{\alpha}$ läßt sich in eine Ableitung in G^c von $\vec{\alpha}$ übersetzen, indem jede Anwendung von ρ durch die Folge $\rho_0^c, \rho_1^c, \dots, \rho_{n-1}^c$ ersetzt wird. Für die Umkehrung führe folgende Priorisierung π auf den Regeln ein. Es sei Z_i^ρ stets vor Y_i. Aber in $Z_{n-3}^\rho \to Y_{n-2} Y_{n-1}$ wähle die linksseitige Priorisierung. Nun gilt $G \vdash^\ell \vec{x}$ genau dann, wenn $G^c \vdash^\pi \vec{x}$. Ist nämlich $\langle \alpha_i : i < p+1 \rangle$ eine linksseitige Ableitung von $\vec{x}$ in G, so ersetze jede Anwendung einer Regel ρ durch die Sequenz ρ_0^c, ρ_1^c, etc. bis ρ_{n-2}^c. Dies ist eine G^c–Ableitung, wie man leicht prüft. Es ist sogar eine π–Ableitung. Sei umgekehrt $\langle \beta_j : j < q+1 \rangle$ eine G^c–Ableitung, die mit π priorisiert ist. Ist dann β_{i+1} das Ergebnis einer Anwendung der Regel ρ_k^c, für $k < n-2$, so ist $i+2 < q+1$ und β_{i+2} das Ergebnis einer Anwendung von ρ_{k+1}^c auf β_{i+1}, welche genau das von der vorigen Anwendung erzeugte Vorkommen von Z_k ersetzt. Dies bedeutet, daß jedes ρ_k^c in einem Block von Anwendungen von ρ_0^c , ρ_1^c etc. bis ρ_{n-2}^c liegt, der genau einer Anwendung von ρ entspricht. Es existiert somit eine G–Ableitung von $\vec{x}$, welche man durch Rückersetzen der Blöcke gewinnt. Diese ist linksseitig. ⊣

Zum Beispiel ist die rechte Grammatik das Ergebnis der Konversion der linken Grammatik in Chomsky–Standardform.

S	→	ASBBT | ABB	S	→	AX | AV
			V	→	BB
			X	→	SY
			Y	→	BZ
			Z	→	BT
T	→	CTD | CD	T	→	CW | CD
			W	→	TD
A	→	a	A	→	a
B	→	b	B	→	b
C	→	c	C	→	c
D	→	d	D	→	d

Definition 2.2.12 *Eine kontextfreie Grammatik heißt* ***invertierbar****, falls aus* $X \to \vec{\alpha} \in R$ *und* $Y \to \vec{\alpha} \in R$ *folgt* $X = Y$*.*

Für eine invertierbare Grammatik gilt, daß man die Markierung auf den Bäumen schon durch die Markierung auf den Blättern bestimmt ist. In aller Regel wird dies nicht der Fall sein, auch wenn die Grammatik in Standardform ist. Man kann jedoch jede Grammatik so umschreiben, daß die Markierung auf den Blättern ausreicht. Dazu muß man allerdings den Vorrat an Nichtterminalsymbolen vergrößern. Denn es ist anschaulich klar, daß wenn die Markierung eines G–Baumes nicht schon durch die der Blätter bestimmt ist, sondern Alternativen zuläßt, so muß man die lokale Ambiguität durch Hinzunahme neuer Marken kodieren. Dies geschieht, indem wir statt N nunmehr mit der Potenzmenge, $\wp(N)$ arbeiten.

Wir stellen hier einen Algorithmus vor, der aus einer kontextfreien Grammatik eine invertierbare Grammatik macht. Der Einfachheit halber sei eine Regel stets von der Form $X \to \vec{Y}$ oder $X \to \vec{x}$. Dazu wählen wir unsere Nichtterminalsymbole jetzt aus $\wp(N) - \{\varnothing\}$. Die terminalen Regeln sind dann von der Form $\mathsf{X} \to \vec{x}$, wobei $\mathsf{X} = \{X : X \to \vec{x} \in R\}$. Die nichtterminalen Regeln sind von der Form $\mathsf{X} \to \mathsf{Y}_0\mathsf{Y}_1 \ldots \mathsf{Y}_{n-1}$ mit

$$\mathsf{X} = \{X : X \to Y_0Y_1 \ldots Y_{n-1} \in R \text{ für gewisse } Y_i \in \mathsf{Y}_i\}$$

Außerdem wählen wir noch ein Startsymbol, Σ, und nehmen als Regeln $\Sigma \to \vec{\mathsf{X}}$ für alle $\vec{X}$, für die es $X_i \in \mathsf{X}_i$ gibt mit $S \to \vec{X} \in R$. Diese Grammatik nennen wir G^i. Es ist nicht schwer nachzurechnen, daß G^i invertierbar ist. Denn sei $\mathsf{Y}_0\mathsf{Y}_1 \ldots \mathsf{Y}_{n-1}$ die rechte Seite einer Produktion. Dann existieren $Y_i \in \mathsf{Y}_i$ und ein X dergestalt, daß $X \to \vec{Y}$ eine Regel in G ist. Also existiert ein X dergestalt, daß $\mathsf{X} \to \vec{\mathsf{Y}}$ in G^i ist. X ist allerdings auch eindeutig bestimmt. Ferner ist G^i in Standardform (Chomsky–Normalform), falls G es ist.

Theorem 2.2.13 *Es sei* G *eine kontextfreie Grammatik. Dann läßt sich eine invertierbare kontextfreie Grammatik* G^i *konstruieren, welche die gleichen Klammerungsanalysen erzeugt wie* G*.* $\dashv$

Der Vorteil, den eine invertierbare Grammatik bietet, ist der, daß eine Klammerungsstruktur auf einer Zeichenkette schon die Marken auf den inneren Knoten rekonstruieren läßt. Der Leser überlege sich, daß G genau dann invertierbar ist, wenn G^b invertierbar ist.

Definition 2.2.14 *Eine kontextfreie Grammatik heißt* ***perfekt****, falls sie in Standardform ist, schlank, reduziert und invertierbar.*

Es ist instruktiv, ein Beispiel einer Grammatik zu sehen, welche invertierbar ist aber nicht reduziert.

		G			H
S	$\to$	AS | BS | A | B	S	$\to$	CS | C
A	$\to$	a	C	$\to$	a | b
B	$\to$	b			

G ist invertierbar aber nicht reduziert. Dazu betrachte man H und die Abbildung $\mathtt{A}, \mathtt{B} \mapsto \mathtt{C}$, $\mathtt{S} \mapsto \mathtt{S}$. Diese ist eine R–Simulation. H ist reduziert und invertierbar.

Theorem 2.2.15 *Zu jeder kontextfreien Grammatik läßt sich effektiv eine perfekte Grammatik konstruieren, welche dieselben Konstituentenanalysen erzeugt.*

Als letztes wollen wir uns der sogenannten *Greibach–Normalform* zuwenden. Diese ist besonders wichtig für Erkennungsalgorithmen, welche einen Input von links nach rechts einlesen. Solche Algorithmen haben Probleme mit Regeln der Form $X \to Y \cdot \vec{\alpha}$, insbesondere, wenn $Y = X$.

Definition 2.2.16 *Es sei $G = \langle \mathtt{S}, N, A, R\rangle$ eine kontextfreie Grammatik. G ist in **Greibach–(Normal)form**, falls jede Regel von der Form $\mathtt{S} \to \varepsilon$ oder von der Form $X \to x \cdot \vec{Y}$ ist.*

Proposition 2.2.17 *Es sei G eine Grammatik in Greibach–Form und $X \vdash_G \vec{\alpha}$. Genau dann ist $\vec{\alpha}$ linksseitig aus X in G ableitbar, wenn $\vec{\alpha} = \vec{y} \cdot \vec{Y}$ für gewisse $\vec{y} \in A^*$ und $\vec{Y} \in N^*$ und $\vec{y} = \varepsilon$ nur dann, wenn $\vec{Y} = X$.*

Der Beweis dieser Behauptung ist nicht schwer. Es ist ebenso nicht schwer zu sehen, daß diese Eigenschaft die Greibach–Form exakt charakterisiert. Denn falls eine Regel der Form $X \to Y \cdot \vec{\gamma}$ existiert, so ist eben $Y \cdot \vec{\gamma}$ linksseitig aus X herleitbar, aber nicht in der gewünschten Form. Dabei setzen wir voraus, daß es keine Regeln der Form $X \to X$ gibt.

Theorem 2.2.18 (Greibach) *Zu jeder kontextfreien Grammatik G kann man effektiv eine Grammatik G^g in Greibach–Normalform konstruieren mit $L(G^g) = L(G)$.*

Bevor wir den eigentlichen Beweis beginnen, wollen wir zwei Hilfsüberlegungen anstellen. Wir nennen ρ eine X–**Produktion**, falls $\rho = X \to \vec{\alpha}$ für gewisses $\vec{\alpha}$. Eine solche Produktion heiße **linksrekursiv**, falls sie von der Form $X \to X \cdot \vec{\beta}$ ist. Sei $\rho = X \to \vec{\alpha}$ eine Regel; definiere $R^{-\rho}$ wie folgt. Für jede Faktorisierung $\vec{\alpha} = \vec{\alpha}_1 \cdot Y \cdot \vec{\alpha}_2$ von $\vec{\alpha}$ und jede Regel $Y \to \vec{\beta}$ füge die Regel $X \to \vec{\alpha}_1 \cdot \vec{\beta} \cdot \vec{\alpha}_2$ zu R hinzu und nimm am

Ende die Regel ρ weg. Nun sei $G^{-\rho} = \langle \mathtt{S}, N, A, R^{-\rho}\rangle$. Dann ist $L(G^{-\rho}) = L(G)$. Wir sprechen von dieser Konstruktion als dem **Überspringen** der Regel ρ. Der Leser überzeuge sich, daß die Bäume für $G^{-\rho}$ auf sehr einfache Weise aus den Bäumen von G gewonnen werden können, nämlich durch Wegnehmen sämtlicher Knoten x, deren lokaler Baum gerade der Regel ρ entspricht, d. h. isomorph ist zu $\mathfrak{H}_\rho$. (Dies war in Abschnitt 1.6 definiert.) Diese Technik funktioniert allerdings nur, wenn ρ keine $\mathtt{S}$–Produktion ist. In diesem Fall geht man wie folgt vor. Man ersetzt ρ durch alle Regeln der Form $\mathtt{S} \to \vec{\beta}$, wo $\vec{\beta}$ aus $\vec{\alpha}$ durch Anwenden einer Regel abgeleitet ist. Man überlege sich, daß das Überspringen einer Regel nicht unbedingt zu einer neuen Grammatik führt, nämlich dann nicht, wenn es Regeln der Form $X \to Y$ (insbesondere also Regeln der Form $X \to X$) gibt.

Lemma 2.2.19 *Es sei $G = \langle \mathtt{S}, N, A, R\rangle$ eine Grammatik und es seien $\mathtt{X} \to \mathtt{X} \cdot \vec{\alpha}_i$, $i < m$, sämtliche linksrekursiven $\mathtt{X}$–Produktionen, sowie $\mathtt{X} \to \vec{\beta}_j$, $j < n$, sämtliche nicht–linksrekursiven $\mathtt{X}$–Produktionen. Es sei $G^1 := \langle \mathtt{S}, N \cup \{\mathtt{Z}\}, A, R^1\rangle$, wo $\mathtt{Z} \notin N \cup A$ und R^1 aus allen Y–Produktionen aus R besteht mit $Y \neq \mathtt{X}$ sowie aus den Produktionen*

$$\begin{array}{llll} \mathtt{X} \to \vec{\beta}_j, & j < n, & \mathtt{Z} \to \vec{\alpha}_i, & i < m, \\ \mathtt{X} \to \vec{\beta}_j \cdot \mathtt{Z}, & j < n, & \mathtt{Z} \to \vec{\alpha}_i \cdot \mathtt{Z}, & i < m. \end{array}$$

Dann ist $L(G^1) = L(G)$.

Beweis. Wir werden dieses Lemma relativ ausführlich beweisen, da die Methode relativ trickreich ist. Wir betrachten auf G^1 die folgende Priorisierung. In den Regeln $\mathtt{X} \to \vec{\beta}_j$ und $\mathtt{Z} \to \vec{\alpha}_i$ nehmen wir die natürliche Ordnung (d. i. die linksseitige) und in den Regeln $\mathtt{X} \to \vec{\beta}_j \cdot \mathtt{Z}$ sowie $\mathtt{Z} \to \vec{\alpha}_i \cdot \mathtt{Z}$ setzen wir auch die natürliche Ordnung mit der Ausnahme, daß $\mathtt{Z}$ allen Elementen aus $\vec{\alpha}_j$ bzw. $\vec{\beta}_i$ vorausgeht. Dies definiert die Linearisierung $\blacktriangleleft$. Nun zum Beweis. Es sei $M(\mathtt{X})$ die Menge aller $\vec{\gamma}$ derart, daß eine linksseitige Ableitung aus $\mathtt{X}$ in G existiert derart, daß $\vec{\gamma}$ das erste Element nicht von der Form $\mathtt{X} \cdot \vec{\delta}$ ist. Genauso definieren wir $P(\mathtt{X})$ die Menge aller $\vec{\gamma}$, die aus $\mathtt{X}$ mit $\blacktriangleleft$ priorisiert in G^1 herleitbar sind derart, daß $\vec{\gamma}$ das erste Element ist, welches kein $\mathtt{Z}$ enthält. Wir behaupten, daß $P(\mathtt{X}) = M(\mathtt{X})$. Man kann sich überlegen, daß

$$M(\mathtt{X}) = \bigcup_{j<n} \vec{\beta} j \cdot (\bigcup_{i<m} \vec{\alpha}_i)^* = P(\mathtt{X})$$

Daraus folgt nun die gewünschte Behauptung so. Es sei $\vec{x} \in L(G)$. Dann existiert eine linksseitige Ableitung $\Gamma = \langle \vec{\gamma}_i : i < n+1\rangle$ von $\vec{x}$. Diese Ableitung teilen wir in Segmente, Σ_i, $i < \sigma$, der Länge k_i, sodaß

$$\Sigma_i = \langle \vec{\gamma}_j : \sum_{p<i} k_p \leq j < 1 + \sum_{p<i+1} k_i \rangle$$

Diese Einteilung soll so beschaffen sein, daß Σ_i ein maximales Stück in Γ von $\mathtt{X}$–Produktionen ist oder ein maximales Stück von Y–Produktionen mit $Y \neq \mathtt{X}$. Die $\mathtt{X}$–Segmente können wir nach dem Vorangegangen durch eine $\blacktriangleleft$–Ableitung $\widehat{\Sigma}_i$ in G^1 ersetzen. Die Segmente, welche keine X–Produktionen enthalten, sind schon G^1–Ableitungen. Für sie sei $\widehat{\Sigma}_i := \Sigma_i$. Jetzt sei $\widehat{\Gamma}$ die durch Verkettung der $\widehat{\Sigma}_i$ entstehende Ableitung. Dies ist wohldefiniert, da das erste Element von $\widehat{\Sigma}_i$ gleich dem ersten Element von Σ_i, wie auch das letzte Element von $\widehat{\Sigma}_i$ gleich dem letzten Element von Σ_i ist. $\widehat{\Gamma}$ ist eine G^1–Ableitung, priorisiert durch $\blacktriangleleft$. Also $\vec{x} \in L(G^1)$. Die Umkehrung ist analog zu beweisen, indem man von einer mit $\blacktriangleleft$ priorisierten Ableitung beginnt. $\dashv$

Doch nun zum Beweis des Satzes 2.2.18. Wir können von vornherein annehmen, daß G in Chomsky–Normalform ist. Wir wählen eine Aufzählung von N als $N = \{X_i : i < p\}$. Wir behaupten nun als erstes, daß wir es unter Hinzunahme neuer Nichtterminalsymbole erreichen können, daß wir eine Grammatik G^1 bekommen mit $L(G^1) = L(G)$, in welcher die X_i–Produktionen die Form $X_i \to x \cdot \vec{Y}$ oder $X_i \to X_j \cdot \vec{Y}$ mit $j > i$ haben. Dies beweisen wir durch Induktion über i. Sei i_0 das kleinste i derart, daß eine Regel $X_i \to X_j \cdot \vec{Y}$ existiert mit $j \leq i$. Sei j_0 nun noch das größte j derart, daß $X_{i_0} \to X_j \cdot \vec{Y}$ eine Regel ist. Wir können zwei Fälle unterscheiden. Der erste Fall ist $j_0 = i_0$. Nach dem vorigen Lemma können wir durch Einführen eines neuen Symbols Z_{i_0} die Produktion eliminieren. Der zweite Fall ist $j_0 < i_0$. Hier wenden wir die Induktionshypothese auf j_0 an. Wir können die Regel $X_{i_0} \to X_{j_0} \cdot \vec{Y}$ überspringen und so Regeln der Form (a) $X_{i_0} \to X_k \cdot \vec{Y'}$ mit $k > j_0$ einführen. Auf diese Weise wird der zweite Fall entweder aufgelöst oder auf den ersten zurückgespielt.

Es sei $P := \{Z_i : i < p\}$ die Menge der neu hinzugekommenen Symbole. Unter Umständen kommt für gewisse j Z_j überhaupt nicht in der Grammatik vor, aber das macht für den Beweis nichts. Sei noch $P_i := \{Z_j : j < i\}$. Am Ende dieser Reduktion haben wir nun Regeln der Form

$$\begin{array}{lll} (a) & X_i \to X_j \cdot \vec{Y}, & j > i \\ (b) & X_i \to x \cdot \vec{Y}, & x \in A \\ (c) & Z_i \to \vec{W}, & \vec{W} \in (N \cup P_i)^+ \cdot (\varepsilon \cup Z_i) \end{array}$$

Es ist nun klar, daß jede X_{p-1}–Produktion schon die Form $X_{p-1} \to x \cdot \vec{Y}$ hat. Falls eine X_{p-2}–Produktion aber die Form $X_{p-2} \to X_{p-1} \cdot \vec{Y}$ hat, so können wir diese Regel überspringen und bekommen Regeln der Form $X_{p-2} \to \vec{x}\vec{Y'}$. Induktiv kann man also zeigen, daß alle Regeln der Form (a) zugunsten von Regeln der Form (b) übersprungen werden können. Nun noch die Regeln der letzten Zeile. Auch sie können übersprungen werden, und dann bekommen wir Regeln der Form $Z \to x \cdot \vec{Y}$ für ein $x \in A$, wie gewünscht.

Zum Beispiel sei folgende Grammatik gegeben.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{SDA} \mid \mathtt{CC} \\ \mathtt{D} & \to & \mathtt{DC} \mid \mathtt{AB} \\ \mathtt{A} & \to & \mathtt{a} \\ \mathtt{B} & \to & \mathtt{b} \\ \mathtt{C} & \to & \mathtt{c} \end{array}$$

Die Produktion $\mathtt{S} \to \mathtt{SDA}$ ist linksrekursiv. Wir ersetzen sie gemäß dem obenstehenden Lemma durch

$$\mathtt{S} \to \mathtt{CCZ}, \quad \mathtt{Z} \to \mathtt{DA}, \quad \mathtt{Z} \to \mathtt{DAZ}$$

Ebenso ersetzen wir die Produktion $\mathtt{D} \to \mathtt{DC}$ durch

$$\mathtt{D} \to \mathtt{ABY}, \quad \mathtt{Y} \to \mathtt{C}, \quad \mathtt{Y} \to \mathtt{CY}$$

Damit erhalten wir folgende Grammatik

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{CC} \mid \mathtt{CCZ} \\ \mathtt{Z} & \to & \mathtt{DA} \mid \mathtt{DAZ} \\ \mathtt{D} & \to & \mathtt{AB} \mid \mathtt{ABY} \\ \mathtt{Y} & \to & \mathtt{C} \mid \mathtt{CY} \\ \mathtt{A} & \to & \mathtt{a} \\ \mathtt{B} & \to & \mathtt{b} \\ \mathtt{C} & \to & \mathtt{c} \end{array}$$

Anschließend überspringen wir die D–Produktionen.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{CC} \mid \mathtt{CCZ} \\ \mathtt{Z} & \to & \mathtt{ABA} \mid \mathtt{ABYA} \mid \mathtt{ABAZ} \mid \mathtt{ABYZ} \\ \mathtt{D} & \to & \mathtt{AB} \mid \mathtt{ABY} \\ \mathtt{Y} & \to & \mathtt{C} \mid \mathtt{CY} \\ \mathtt{A} & \to & \mathtt{a} \\ \mathtt{B} & \to & \mathtt{b} \\ \mathtt{C} & \to & \mathtt{c} \end{array}$$

Als nächstes kann D eliminiert werden (es ist nicht erreichbar) und wir können in den rechten Seiten der Produktionen die ersten Nichtterminale durch Terminalsymbole ersetzen.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{cC} \mid \mathtt{cCZ} \\ \mathtt{Z} & \to & \mathtt{aBA} \mid \mathtt{aBYA} \mid \mathtt{aBAZ} \mid \mathtt{aBYZ} \\ \mathtt{Y} & \to & \mathtt{c} \mid \mathtt{cY} \end{array}$$

Nun ist die Grammatik in Greibach–Normalform.

Übung 37. Zeigen Sie, daß für eine gegebene kontextfreie Grammatik G entscheidbar ist, (a) ob $L(G) = \varnothing$, (b) ob $L(G)$ endlich ist, (c) ob $L(G)$ unendlich ist.

Übung 38. Es sei G^i die wie oben konstruierte invertierbare Grammatik zu G. Man zeige, daß die Relation $\sim$ mit

$$\mathsf{X} \sim Y \quad \Leftrightarrow \quad Y \in \mathsf{X}$$

eine Rückwärts–Simulation von G^i nach G ist.

Übung 39. Es sei $\langle B, <, \sqsubset, \ell\rangle$ ein geordneter markierter Baum. Falls x ein Blatt ist, so ist $\uparrow x$ ein Zweig, und kann in natürlicher Weise als Zeichenkette $\langle \uparrow x, >, \ell\rangle$ aufgefaßt werden. Da das Blatt x eine Sonderrolle spielt, wollen wir es weglassen. Wir sagen, ein **Zweigausdruck** ist eine Zeichenkette der Form $\langle \uparrow x - \{x\}, >, \ell\rangle$, x ein Blatt. Wir nennen ihn $\zeta(x)$. Man zeige, die Menge

$$Z(G) = \{\zeta(x) : \zeta(x) \text{ Zweigausdruck von } \mathfrak{B}, \mathfrak{B} \in L_B(G)\}$$

ist regulär.

Übung 40. Es sei G in Greibach–Normalform und $\vec{x}$ eine terminale Zeichenkette der Länge $n > 0$. Man zeige, daß jede Ableitung von $\vec{x}$ genau die Länge n hat. Läßt sich auch für beliebige Zeichenketten $\vec{\alpha}$ eine solche Zahl angeben?

2.3 Erkennung und Analyse

Kontextfreie Sprachen können ebenso wie reguläre Sprachen durch gewisse Automaten charakterisiert werden. Ein endlicher Automat ist eine Maschine, welche ihre Handlungen nur über ein begrenztes Gedächtnis steuert. Da es kontextfreie Sprachen gibt, welche nicht regulär sind, müssen die Automaten, welche kontextfreie Sprachen erkennen können, über ein unendliches Gedächtnis verfügen. Die spezielle Art, wie ein solches Gedächtnis aufgebaut ist und manipuliert werden kann, differenziert einzig die verschiedenen Klassen nichtregulärer Sprachen. Kontextfreie Sprachen werden durch sogenannte *Kellerautomaten* charakterisiert. Diese haben ein Gedächtnis in Form eines Stapels, auf welchem sie Symbole ablegen und je nach Bedarf wieder entfernen können. Allerdings hat der Automat nur Zugang zu dem zuletzt abgelegten Symbol. Alle anderen Symbole sind erst zugänglich, nachdem die nach ihnen abgelegten Symbole entfernt worden sind. Ein *Stapel* über einem Alphabet D ist schlicht eine Zeichenkette über D. Wir wollen vereinbaren, daß in einer Zeichenkette über D der erste Buchstabe dem höchsten Eintrag im Stapel entspricht, und der letzte Buchstabe dem untersten Element. Analog zu den endlichen Automaten benötigen wir ein Startsymbol, welches wir in der Regel mit # bezeichnen und mit welchem die Rechnung beginnt.

Ein *Kellerautomat* bestimmt seine Handlungen aus dem Inhalt des obersten Elements im Stapel und dem aktuellen Zustand; seine Handlungen bestehen aus drei

Schritten: (1.) das Ablegen auf den Stapel oder Entfernen vom Stapel, (2.) das Bewegen oder Stillstehen auf der zu lesenden Zeichenkette und (3.) das Übergehen in einen neuen Zustand. Falls der Automat in (2.) nicht wirklich bewegt, so nennen wir diese Handlung eine ε–**Bewegung**. Wir schreiben A_ε anstelle von $A \cup \{\varepsilon\}$.

Definition 2.3.1 *Ein **Kellerautomat** über A ist ein Septupel*

$$\mathfrak{K} = \langle Q, i_0, A, F, D, \#, \delta \rangle \quad ,$$

*wobei Q und D endliche, nichtleere Menge sind, $i_0 \in Q$, $\# \in D$ und $F \subseteq Q$, sowie δ eine Funktion $\delta : Q \times D \times A_\varepsilon \to \wp(Q \times D^*)$, dergestalt, daß $\delta(q, a, d)$ stets endlich ist. Wir nennen Q die Menge der **Zustände**, i_0 den **Startzustand**, F die Menge der **akzeptierenden Zustände**, D das **Stapelalphabet**, $\#$ den **Stapelbeginn** und δ die **Handlungsfunktion**.*

Wir nennen $\mathfrak{z} := \langle q, \vec{d} \rangle$, wo $q \in Q$ und $\vec{d} \in D^*$ eine **Konfiguration**. Wir schreiben nun

$$\langle p, \vec{d} \rangle \xrightarrow{x} \langle p', \vec{d'} \rangle$$

falls $\vec{d} = Z \cdot \vec{d_1}$, $\vec{d'} = \vec{e} \cdot \vec{d_1}$ und $\langle p', \vec{e} \rangle \in \delta(p, Z, x)$. Wir nennen dies einen **Übergang**. Wir dehnen stillschweigend die Funktion δ auch auf Konfigurationen aus und schreiben auch $\langle p', \vec{d'} \rangle \in \delta(p, \vec{d}, x)$. Man beachte, daß es im Gegensatz zu endlichen Automaten erlaubt ist, in einen neuen Zustand zu wechseln, ohne ein neues Symbol einzulesen. Dies ist insbesondere dann notwendig, wenn man den Automaten in einem Zustand den Inhalt des Stapels löschen lassen will. Falls der Stapel leer ist, so kann der Automat allerdings nicht weiterarbeiten. Dies bedeutet insbesondere, daß Kellerautomaten notwendig partiell sind. Die Handlungsfunktion kann nun analog den endlichen Automaten auf Zeichenketten ausgedehnt werden. Ebenso können wir statt eines einzelnen Zustandes Mengen von Zuständen als Argumente zulassen. Wir definieren also

$$\mathfrak{z} \xrightarrow{\vec{x} \cdot \vec{y}} \mathfrak{z}' \quad \Leftrightarrow \quad \text{es existiert } \mathfrak{z}'' \text{ mit } \mathfrak{z} \xrightarrow{\vec{x}} \mathfrak{z}'' \xrightarrow{\vec{y}} \mathfrak{z}' \ .$$

Wir sagen im Fall $\mathfrak{z} \xrightarrow{\vec{x}} \mathfrak{z}'$, es existiere eine $\mathfrak{K}$–**Berechnung** für $\vec{x}$ von $\mathfrak{z}$ nach $\mathfrak{z}'$. Nun ist

$$L(\mathfrak{K}) := \{\vec{x} : \textit{für ein } q \in F \textit{ und ein } \vec{z} \in D^* : \langle i_0, \# \rangle \xrightarrow{\vec{x}} \langle q, \vec{z} \rangle\}$$

Wir nennen dies die Sprache, welche von dem Kellerautomaten **über den Zustand** akzeptiert wird. Wir nennen einen Kellerautomaten **einfach**, wenn aus $\langle q, \vec{z} \rangle \in \delta(p, Z, a)$ folgt: $|\vec{z}| \leq 1$, oder auch $\vec{z} \in D_\varepsilon$. Den folgenden Satz zu beweisen ist eine Übung.

Proposition 2.3.2 *Zu jedem Kellerautomaten $\mathfrak{K}$ gibt es einen einfachen Kellerautomaten $\mathfrak{L}$ mit $L(\mathfrak{L}) = L(\mathfrak{K})$.*

Deswegen werden wir oft auch stillschweigend annehmen, daß der Automat nicht eine Zeichenkette sondern höchstens ein Symbol in den Keller schreibt. Neben $L(\mathfrak{K})$ gibt es auch die Sprache, die von dem Automaten **über den Stapel** akzeptiert wird. Diese ist definiert durch

$$L^s(\mathfrak{K}) := \{\vec{x} : \textit{für ein } q \in Q : \langle i_0, \# \rangle \xrightarrow{\vec{x}} \langle q, \varepsilon \rangle\}$$

Natürlich sind $L(\mathfrak{K})$ und $L^s(\mathfrak{K})$ nicht unbedingt gleich für ein gegebenes $\mathfrak{K}$. Allerdings ist die Menge aller Sprachen der Form $L(\mathfrak{K})$ für einen Kellerautomaten gleich der Menge der Sprachen mit $L^s(\mathfrak{K})$ für einen Kellerautomaten. Dies ist der Inhalt des folgenden Satzes.

Proposition 2.3.3 *Zu jedem Kellerautomaten $\mathfrak{K}$ existiert ein $\mathfrak{L}$ mit $L(\mathfrak{K}) = L^s(\mathfrak{L})$ sowie ein Kellerautomat $\mathfrak{M}$ mit $L^s(\mathfrak{K}) = L(\mathfrak{M})$.*

Beweis. Sei $\mathfrak{K} = \langle Q, i_0, F, D, \#, \delta \rangle$ gegeben. Wir nehmen zu Q noch zwei Zustände q_i und q_f hinzu. Es soll q_i der neue Startzustand sein und $F^{\mathfrak{L}} := \{q_f\}$. Ferner nehmen wir noch ein neues Symbol $\flat$, welches jetzt der Stapelbeginn in $\mathfrak{L}$ wird. Wir definieren $\delta^{\mathfrak{L}}(q_i, \flat, \varepsilon) := \{\langle i_0, \# \cdot \flat \rangle\}$. Ansonsten soll es keine weiteren $\delta^{\mathfrak{L}}$–Übergänge aus q_i geben. Für $q \neq q_i, q_f$ und $Z \neq \flat$ soll $\delta^{\mathfrak{L}}$ wie $\delta^{\mathfrak{K}}$ definiert sein. Wir fügen zu $\delta^{\mathfrak{K}}(q, d, \varepsilon)$ noch $\{\langle q_f, \varepsilon \rangle\}$ hinzu, falls $q \in F$ und $d \neq \flat$, ansonsten nichts. Schließlich sei $\delta^{\mathfrak{L}}(q_f, Z, x) := \varnothing$ für $x \in A$ und $\delta^{\mathfrak{L}}(q_f, Z, \varepsilon) := \{\langle q_f, \varepsilon \rangle\}$ für $Z \in D \cup \{\flat\}$. Es sei nun $\vec{x} \in L(\mathfrak{K})$. Dann existiert eine $\mathfrak{K}$–Berechnung $\langle i_0, \# \rangle \xrightarrow{\vec{x}} \langle q, \vec{d} \rangle$ mit $q \in F$, und so auch eine $\mathfrak{L}$–Berechnung $\langle q_i, \flat \rangle \xrightarrow{\vec{x}} \langle q_f, \vec{d} \rangle$. Da $\langle q_f, \vec{d} \rangle \xrightarrow{\varepsilon} \langle q_f, \varepsilon \rangle$, so ist $\vec{x} \in L^s(\mathfrak{L})$. Also ist $L(\mathfrak{K}) \subseteq L^s(\mathfrak{L})$. Nun sei umgekehrt $\vec{x} \in L^s(\mathfrak{L})$, also $\langle q_i, \flat \rangle \xrightarrow{\vec{x}} \langle p, \varepsilon \rangle$ für ein gewisses p. Dann gilt, da $\flat$ stets als letztes gelöscht wird und dies nur in q_f geschehen kann, $p = q_f$. Ferner gilt dann $\langle q_i, \flat \rangle \xrightarrow{\vec{x}} \langle q, \vec{d} \cdot \flat \rangle$ für einen Zustand $q \in F$. Dies bedeutet, daß es eine $\mathfrak{L}$–Berechnung $\langle i_0, \# \cdot \flat \rangle \xrightarrow{\vec{x}} \langle q, \vec{d} \cdot \flat \rangle$ gibt. Dieser ist aber auch eine $\mathfrak{K}$–Berechnung. Dies zeigt $L^s(\mathfrak{L}) \subseteq L(\mathfrak{K})$ und damit die erste Behauptung. Nun zur Konstruktion von $\mathfrak{M}$. Wir nehmen zwei neue Zustände hinzu, q_f und q_i, ein neues Stapelsymbol $\flat$, welches der Stapelbeginn von $\mathfrak{M}$ sein soll, und setzen $F^{\mathfrak{M}} := \{q_f\}$. Wiederum ist $\delta^{\mathfrak{M}}(q_i, \flat, x) := \varnothing$ für $x \in A$ und $\delta^{\mathfrak{M}}(q_i, \flat, x) := \{\langle i_0, \# \cdot \flat \rangle\}$ für $x = \varepsilon$. Ferner wollen wir setzen $\delta^{\mathfrak{M}}(q, Z, x) := \delta^{\mathfrak{K}}(q, Z, x)$ für $Z \neq \flat$ und $\delta^{\mathfrak{M}}(q, \flat, \varepsilon) := \{\langle q_f, \varepsilon \rangle\}$, sowie $\delta^{\mathfrak{M}}(q, \flat, x) := \varnothing$ für $x \in A$. Ferner sei $\delta^{\mathfrak{M}}(q_f, Z, x) := \varnothing$. Dies definiert $\delta^{\mathfrak{M}}$. Nun betrachte man ein $\vec{x} \in L^s(\mathfrak{K})$. Es existiert eine $\mathfrak{K}$–Berechnung $\langle i_0, \# \rangle \xrightarrow{\vec{x}} \langle p, \varepsilon \rangle$ für ein gewisses p. Dann existiert eine $\mathfrak{L}$–Berechnung

$$\langle q_i, \flat \rangle \xrightarrow{\vec{x}} \langle p, \flat \rangle \xrightarrow{\varepsilon} \langle q_f, \varepsilon \rangle$$

Also $\vec{x} \in L(\mathfrak{M})$. Sei umgekehrt $\vec{x} \in L(\mathfrak{M})$. Es existiert also eine $\mathfrak{L}$–Berechnung $\langle q_i, \flat \rangle \xrightarrow{\vec{x}} \langle q_f, \vec{d} \rangle$ für gewisses $\vec{d}$. Man kann sich nun leicht überzeugen, daß $\vec{d} = \varepsilon$ sein

muß. Ferner faktorisiert diese Berechnung in

$$\langle q_i, \flat\rangle \xrightarrow{\varepsilon} \langle i_0, \# \cdot \flat\rangle \xrightarrow{\vec{x}} \langle p, \flat\rangle \xrightarrow{\varepsilon} \langle q_f, \varepsilon\rangle$$

wobei $p \in Q$ ist, also $p \neq q_f, q_i$. Aber jeder $\mathfrak{M}$–Übergang von i_0 nach p ist auch ein $\mathfrak{K}$-Übergang. Also gibt es eine $\mathfrak{K}$–Berechnung $\langle i_0, \#\rangle \xrightarrow{\vec{x}} \langle p, \varepsilon\rangle$. Daraus folgt $\vec{x} \in L^s(\mathfrak{K})$, und so $L^s(\mathfrak{K}) = L(\mathfrak{M})$. $\dashv$

Lemma 2.3.4 *Es sei S eine kontextfreie Sprache über A. Dann existiert ein Kellerautomat $\mathfrak{K}$ mit $S = L^s(\mathfrak{K})$.*

Beweis. Wir nehmen eine kontextfreie Grammatik $G = \langle S, N, A, R\rangle$ in Greibach–Form mit $S = L(G)$. Wir nehmen an, $\varepsilon \notin G$. (Falls $\varepsilon \in L(G)$, so läßt sich leicht eine Modifikation des Automaten angeben.) Der Automat besitzt nur einen Zustand, i_0, und benutzt N als Stapelalphabet. Der Stapelbeginn ist S.

$$\delta(i_0, X, x) := \{\langle i_0, \vec{Y}\rangle : X \to x \cdot \vec{Y} \in R\}$$

Dies definiert $\mathfrak{K} := \langle\{i_0\}, i_0, A, \{i_0\}, N, \mathtt{S}, \delta\rangle$. Wir zeigen $S = L^s(\mathfrak{K})$. Dazu überlegen wir, daß es zu jedem $\vec{x}$ eine linksseitige Ableitung gibt. Jede solche Ableitung erzeugt bei einer Grammatik in Greibach–Form Ketten der Form $\vec{y} \cdot \vec{Y}$. Nun zeige man induktiv, daß $G \vdash \vec{y} \cdot \vec{Y}$ genau dann, wenn $\langle i_0, \vec{Y}\rangle \in \delta(i_0, S, \vec{y})$. $\dashv$

Lemma 2.3.5 *Es sei $\mathfrak{K}$ ein Kellerautomat. Dann ist $L^s(\mathfrak{K})$ kontextfrei.*

Beweis. Es sei $\mathfrak{K} = \langle Q, i_0, A, F, D, \#, \delta\rangle$ ein Kellerautomat. Es sei $N = Q \times D \times (Q \cup \{\mathtt{S}\})$, wo $\mathtt{S}$ ein neues Symbol ist. $\mathtt{S}$ soll auch gleichzeitig das Startsymbol sein. Wir schreiben ein allgemeines Element von N in der Form $[q, A, p]$. Nun definieren wir $R := R^s \cup R^\delta \cup R^\varepsilon$, wobei

$$\begin{aligned}
R^s &:= \{\mathtt{S} \to [i_0, \#, q] : q \in Q\}\\
R^\delta &:= \{[p, Z, q] \to x[q_0, Y_0, q_1][q_1, Y_1, q_2]\ldots[q_{m-1}, Y_{m-1}, q] :\\
&\qquad \langle q_0, Y_0 Y_1 \ldots Y_{m-1}\rangle \in \delta(p, Z, x)\}\\
R^\varepsilon &:= \{[p, Z, q] \to [p, Y_0, q_1][q_1, Y_1, q_2]\ldots[q_{m-1}, Y_{m-1}, q] :\\
&\qquad \langle p, Y_0 Y_1 \ldots Y_{m-1}\rangle \in \delta(p, Z, \varepsilon)\}
\end{aligned}$$

Die so definierte Grammatik nennen wir $G(\mathfrak{A})$. Man beachte, daß bei der Definition von R^δ der Fall $m = 0$ zugelassen ist; in diesem Fall ergibt sich eine Regel der Form $[p, Z, q] \to x$, für alle p, q, x, Z mit $\langle q, \varepsilon\rangle \in \delta(p, Z, x)$. Wir zeigen nun, daß für alle $\vec{x} \in A^*$, alle $p, q \in Q$ sowie $Z \in D$ gilt

$$(\dagger) \qquad [p, Z, q] \vdash_G \vec{x} \quad \Leftrightarrow \quad \langle q, \varepsilon\rangle \in \delta(p, Z, \vec{x})$$

Dies genügt für den Beweis. Denn falls $\vec{x} \in L(G)$, so ist dann $[i_0, \#, q] \vdash_G \vec{x}$ und so wegen (†) $\langle q, \varepsilon\rangle \in \delta(i_0, \#, \vec{x})$, was nichts anderes ist als $\vec{x} \in L^s(\mathfrak{K})$. Und falls Letzteres gilt, so haben wir $[i_0, \#, q] \vdash_G \vec{x}$ und so $\mathtt{S} \vdash_G \vec{x}$, was nichts anderes ist als $\vec{x} \in L(G)$.

Nun also zum Beweis von (†). Es ist klar, daß (†) aus der Behauptung (‡) folgt.

$$(\ddagger) \qquad \begin{array}{c} [p, Z, q] \vdash_G^\ell \vec{y} \cdot [q_0, Y_0, q_1][q_1, Y_1, q_2] \ldots [q_{m-1}, Y_{m-1}, q] \\ \Leftrightarrow \quad \langle q_0, Y_0 Y_1 \ldots Y_{m-1}\rangle \in \delta(p, Z, \vec{y}) \end{array}$$

(‡) nun beweist man leicht durch Induktion. ⊣

Man überlege sich, daß es zu jedem Automaten $\mathfrak{K}$ auch einen Automaten $\mathfrak{L}$ gibt mit nur einem akzeptierenden Zustand, der die gleiche Sprache akzeptiert. Nimmt man $\mathfrak{L}$ anstelle von $\mathfrak{K}$, so erübrigt sich der Kunstgriff mit dem neuen Startsymbol. Anders gesagt kann man dann $[i_0, \#, q]$ als Startsymbol wählen, wo q der akzeptierende Zustand von $\mathfrak{L}$ ist.

Theorem 2.3.6 (Chomsky) *Die kontextfreien Sprachen sind genau die Sprachen, welche durch Kellerautomaten akzeptiert werden, entweder über den Zustand oder über den Stapel.*

Aus den Beweisen lassen sich noch etliche weitere Folgerungen ziehen. Die erste Folgerung ist, daß man zu jedem Kellerautomaten $\mathfrak{K}$ einen Kellerautomaten $\mathfrak{L}$ konstruieren kann, für den $L^s(\mathfrak{L}) = L^s(\mathfrak{K})$, welcher keine ε–Bewegungen enthält. Ferner existiert ein Kellerautomat $\mathfrak{M}$ mit $L^s(\mathfrak{M}) = L^s(\mathfrak{K})$, der nur einen Zustand besitzt, welcher gleichzeitig initialer und akzeptierender Zustand ist. Für einen solchen Automaten reduzieren sich die Definitionen erheblich. Ein solcher Automat besitzt als Gedächtnis lediglich eine Zeichenkette. Die Handlungsfunktion kann dann reduziert werden auf eine Funktion ζ von $A \times D^*$ in endliche Teilmengen von D^*. (Wir erlauben keine ε–Bewegungen.)

Der Kellerautomat läuft die Kette von links nach rechts durch. Er erkennt also in linearer Zeit, ob die Zeichenkette in der Sprache ist oder nicht. Allerdings ist der Automat nichtdeterministisch. Hierbei wird das Konzept eines deterministischen Automaten wie folgt definiert.

Definition 2.3.7 *Ein Kellerautomat $\mathfrak{K} = \langle Q, i_0, A, F, D, \#, \delta\rangle$ ist **deterministisch**, falls gilt: für jedes $q \in Q$ ist $|\delta(q, Z, x)| \leq 1$ und für alle $q \in Q$ und $Z \in D$ entweder (a) $\delta(q, Z, \varepsilon) = \varnothing$ oder (b) $\delta(q, Z, a) = \varnothing$ für alle $a \in A$. Eine Sprache S heißt **deterministisch**, falls $S = L(\mathfrak{A})$ für einen deterministischen Automaten $\mathfrak{A}$. Die Menge der deterministischen Sprachen wird mit Δ bezeichnet.*

Deterministische Sprachen sind also solche, welche von einem deterministischen Automaten über den Zustand akzeptiert werden. Kann man also analog zu den regulären Sprachen einen deterministischen Automaten bauen, der dieselbe Sprache erkennt? Die Antwort ist nein. Dazu betrachten wir die **Spiegelsprache** $\{\vec{x}\vec{x}^T : \vec{x} \in A^*\}$. Diese ist sicher kontextfrei. Es gibt aber keinen deterministischen Kellerautomaten, der diese Sprache akzeptiert. Man muß sich dazu vergegenwärtigen, daß der Automat die Zeichenkette $\vec{x} \cdot \vec{x}^T$ mindestens bis zu $\vec{x}$ in den Keller tun muß, um sie dann anschließend mit dem restlichen Wort, $\vec{x}^T$, abzugleichen. Die Maschine muß allerdings raten, wann der Zeitpunkt gekommen ist, den Keller zu leeren. Der Leser möge sich vergewissern, daß dies nicht möglich ist, ohne das ganze Wort zu kennen.

Theorem 2.3.8 *Deterministische Sprachen sind in DTIME(n).*

Der Beweis ist eine Übung. Wir haben gesehen, daß auch reguläre Sprachen in $DTIME(n)$ sind. Dennoch gibt es deterministische Sprachen, welche nicht regulär sind. Eine solche ist $S = \{\vec{x}\mathtt{c}\vec{x}^T : \vec{x} \in \{\mathtt{a}, \mathtt{b}\}^*\}$. Im Gegensatz zu der Spiegelsprache ist S deterministisch. Dazu muß ja die Maschine nur die Zeichenkette in den Keller tun, bis sie das Symbol `c` erreicht, um anschließend den Keller mit der restlichen Zeichenkette zu vergleichen.

Es entsteht nun die Frage, ob man deterministischen Sprachen mit einem deterministischen Automaten auch über den Stapel erkennen kann. Dies ist nicht der Fall. Sei nämlich S eine Sprache mit $S = L^s(\mathfrak{K})$, $\mathfrak{K}$ ein deterministischer Automat. Ist $\vec{x}\,\vec{y} \in S$ für $\vec{y} \neq \varepsilon$, dann ist $\vec{x} \notin S$. Man sagt, S sei **präfixfrei**. Denn ist $\vec{x} \in S$, so existiert eine $\mathfrak{K}$–Berechnung von $\langle q_0, \# \rangle$ nach $\langle q, \varepsilon \rangle$. Ferner gilt, da $\mathfrak{K}$ deterministisch ist: ist $\langle q_0, \# \rangle \xrightarrow{\vec{x}} \mathfrak{z}$, so ist $\mathfrak{z} = \langle q, \varepsilon \rangle$. Ist der Stapel jedoch geleert, kann der Automat nicht mehr weiterarbeiten. Also ist $\vec{x}\,\vec{y} \notin S$. Es gibt nun allerdings deterministische Sprachen, welche nicht präfixfrei sind. Wir stellen eine wichtige Familie von solchen Sprachen vor, die *Dyck*–Sprachen. Dazu sei A ein Alphabet mit r Zeichen. Zu jedem Zeichen x sei $\underline{x}$ ein neues Zeichen. Wir schreiben $\underline{A} := \{\underline{x} : x \in A\}$. Wir führen eine Kongruenz θ auf Zeichenketten ein, welche durch die Gleichung

$$x\underline{x}\ \theta\ \varepsilon$$

erzeugt wird. (Die analoge Gleichung $\underline{x}x\ \theta\ \varepsilon$ gilt jedoch nicht.) Eine Zeichenkette $\vec{x} \in (A \cup \underline{A})^*$ heißt **ausgeglichen**, falls $\vec{x}\ \theta\ \varepsilon$. Genau dann ist $\vec{x}$ ausgeglichen, wenn $\vec{x}$ durch schrittweises Streichen von Teilworten der Form $x\underline{x}$ zu ε umgeschrieben werden kann.

Definition 2.3.9 ***D_r*** *bezeichent die Menge der ausgeglichenen Zeichenketten für ein Alphabet mit r Zeichen. Eine Sprache heißt* ***Dyck–Sprache****, falls sie die Form D_r hat für ein r.*

Proposition 2.3.10 *Dyck–Sprachen sind deterministisch, aber nicht präfixfrei.*

In der Tat erzeugen die folgenden Grammatiken die Dyck–Sprachen:

$$\mathtt{S} \to \mathtt{SS} \mid x\mathtt{S}\underline{x} \mid \varepsilon$$

Dyck–Sprachen sind also kontextfrei. Nun ist leicht zu sehen, daß mit $\vec{x}, \vec{y} \in D_r$ auch $\vec{x}\vec{y} \in D_r$. Daher sind Dyck–Sprachen also keineswegs präfixfrei. Daß Dyck–Sprachen deterministisch sind, folgt aus allgemeinen Sätzen, welche wir später bereitstellen werden. Wir überlassen es daher dem Leser, einen deterministischen Automaten zu konstruieren, welcher eine gegebene Dyck–Sprache erkennt. Damit ist also gezeigt, daß die Sprachen, welche von einem deterministischen Automaten über den Stapel akzeptiert werden, eine echte Teilmenge der deterministischen Sprachen ist. Dies rechtfertigt die folgende Definition.

Definition 2.3.11 *Eine Sprache S heißt **strikt deterministisch**, falls es einen deterministischen Automaten $\mathfrak{K}$ gibt mit $S = L^s(\mathfrak{K})$. Die Menge der strikt deterministischen Sprachen wird mit Δ^s bezeichnet.*

Theorem 2.3.12 *Genau dann ist S strikt deterministisch, wenn S deterministisch und präfixfrei ist.*

Beweis. Wir haben gesehen, daß strikt deterministische Sprachen präfixfrei sind. Nun sei S deterministisch und präfixfrei. Dann existiert ein Automat $\mathfrak{K}$, welcher S über den Zustand akzeptiert. Da S präfixfrei ist, gilt für jedes Wort $\vec{x} \in S$ und jedes echte Präfix $\vec{y}$ von $\vec{x}$, daß wenn $\langle q_0, \#\rangle \stackrel{\vec{y}}{\to} \langle q, \vec{Y}\rangle$, so ist q kein akzeptierender Zustand. Wir nehmen also folgende Veränderung an $\mathfrak{K}$ vor. Es sei $\delta_1(q, Z, x) := \delta^{\mathfrak{K}}(q, Z, x)$, falls q kein akzeptierender Zustand ist. Es sei $\delta_1(q, Z, x) := \varnothing$, falls $q \in F$ und $x \in A$. Es sei $\delta_1(q, Z, \varepsilon) := \{\langle q, \varepsilon\rangle\}$, $Z \in D$. Sei $\mathfrak{L}$ der Automat, der aus $\mathfrak{K}$ durch Ersetzen der Handlungsfunktion $\delta^{\mathfrak{K}}$ durch δ_1 entsteht. $\mathfrak{L}$ ist deterministisch, wie man leicht nachprüft. Ferne ist eine $\mathfrak{L}$–Berechnung faktorisierbar in eine $\mathfrak{K}$–Berechnung gefolgt von einer Löschung des Stapels. Wir behaupten, daß $L(\mathfrak{K}) = L^s(\mathfrak{L})$. Daraus folgt dann die Behauptung. Denn sei $\vec{x} \in L(\mathfrak{K})$. Dann existiert eine $\mathfrak{K}$–Berechnung mit $\vec{x}$ von $\langle q_0, \#\rangle$ nach $\langle q, \vec{Y}\rangle$, wobei $q \in F$. Für kein echtes Präfix $\vec{y}$ von $\vec{x}$ gibt es eine Berechnung in einen akzeptierenden Zustand, da S präfixfrei ist. Also existiert eine $\mathfrak{L}$–Berechnung mit $\vec{x}$ von $\langle q_0, \#\rangle$ nach $\langle q, \vec{Y}\rangle$. Nun ist $\langle q, \vec{Y}\rangle \stackrel{\varepsilon}{\to} \langle q, \varepsilon\rangle$, also $\vec{x} \in L^s(\mathfrak{L})$. Sei umgekehrt $\vec{x} \in L^s(\mathfrak{L})$. Dann existiert eine Berechnung $\langle q_0, \#\rangle \stackrel{\vec{x}}{\to} \langle q, \varepsilon\rangle$. Sei $\vec{Y} \in D^*$ die längste Zeichenkette derart, daß $\langle q_0, \#\rangle \stackrel{\vec{x}}{\to} \langle q, \vec{Y}\rangle$. Dann ist der $\mathfrak{L}$–Schritt vor dem Erreichen von $\langle q, \vec{Y}\rangle$ bereits ein $\mathfrak{K}$–Schritt. Also gibt es eine $\mathfrak{K}$–Berechnung für $\vec{x}$ von $\langle q_0, \#\rangle$ nach $\langle q, \vec{Y}\rangle$, und so $\vec{x} \in L(\mathfrak{K})$. $\dashv$

Der Beweis zu diesem Satz ergibt auch die folgende Tatsache.

Theorem 2.3.13 *Es sei U eine deterministische kontextfreie Sprache. Sei S die Menge aller $\vec{x} \in U$, für welche kein echtes Präfix in U ist. Dann ist S strikt deterministisch.*

Für die folgende Definition treffen wir eine Vereinbahrung, von welcher wir des öfteren Gebrauch machen werden. Es bezeichnet ${}^{(k)}\vec{\alpha}$ das Präfix von $\vec{\alpha}$ der Länge k, falls $\vec{\alpha}$ mindestens die Länge k hat, und andernfalls ist ${}^{(k)}\vec{\alpha} := \vec{\alpha}$.

Definition 2.3.14 *Es sei $G = \langle \mathtt{S}, N, A, R\rangle$ eine Grammatik und $\Pi \subseteq \wp(N \cup A)$ eine Partition. Wir schreiben $\alpha \equiv \beta$, falls es ein $M \in \Pi$ gibt mit $\alpha, \beta \in M$. Π heißt **strikt für G**, falls gilt:*

1. *$A \in \Pi$*
2. *Für $C, C' \in N$ und $\vec{\alpha}, \vec{\gamma}_1, \vec{\gamma}_2 \in (N \cup A)^*$ gilt: ist $C \equiv C'$ und sowohl $C \to \vec{\alpha}\,\vec{\gamma}_1$ als auch $C' \to \vec{\alpha}\,\vec{\gamma}_2 \in R$, so gilt entweder*
   (a) $\vec{\gamma}_1, \vec{\gamma}_2 \neq \varepsilon$ und ${}^{(1)}\vec{\gamma}_1 \equiv {}^{(1)}\vec{\gamma}_2$ oder
   (b) $\vec{\gamma}_1 = \vec{\gamma}_2 = \varepsilon$ und $C = C'$.

Definition 2.3.15 *Eine kontextfreie Grammatik G heißt **strikt deterministisch**, falls es eine für G strikte Partition gibt.*

Als Beispiel betrachten wir die folgende Grammatik (aus [22]):

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{aA} \mid \mathtt{aB} \\ \mathtt{A} & \to & \mathtt{aAa} \mid \mathtt{bC} \\ \mathtt{B} & \to & \mathtt{aB} \mid \mathtt{bD} \\ \mathtt{C} & \to & \mathtt{bC} \mid \mathtt{a} \\ \mathtt{D} & \to & \mathtt{bDc} \mid \mathtt{c} \end{array}$$

$\Pi = \{\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, \{\mathtt{S}\}, \{\mathtt{A}, \mathtt{B}\}, \{\mathtt{C}, \mathtt{D}\}\}$ ist eine strikte Partition. Die erzeugte Sprache ist $\{\mathtt{a}^n\mathtt{b}^k\mathtt{a}^n, \mathtt{a}^n\mathtt{b}^k\mathtt{c}^k : k, n \geq 1\}$.

Wir werden nun zeigen, daß die von strikt deterministischen Grammatiken erzeugten Sprachen genau die strikt deterministischen Sprachen sind. Dies rechtfertigt im Nachhinein die Namensgebung. Zunächst werden wir ein paar einfache Folgerungen aus der Definition ziehen. Eine strikt deterministische Grammatik ist in einem gewissen Sinne invertierbar. Ist $G = \langle \mathtt{S}, N, A, R\rangle$ strikt deterministisch und $R' \subseteq R$, so ist auch $G' = \langle \mathtt{S}, N, A, R'\rangle$ strikt deterministisch. Deshalb kann man zu jeder strikt deterministischen Grammatik eine strikt deterministische schlanke Grammatik konstruieren. Es bezeichne $\vec{\alpha} \Rightarrow_L^n \vec{\gamma}$ die Tatsache, daß $\vec{\gamma}$ aus $\vec{\alpha}$ in n Schritten linksseitig herleitbar ist. Dann gilt:

Lemma 2.3.16 *Es sei G eine kontextfreie Grammatik mit strikter Partition Π. Dann gilt: Für $C, C' \in N$ und $\vec{\alpha}, \vec{\gamma}_1, \vec{\gamma}_2 \in (N \cup A)^*$ gilt: ist $C \equiv C'$ und $C \Rightarrow_L^n \vec{\alpha}\,\vec{\gamma}_1$ sowie $C' \Rightarrow_L^n \vec{\alpha}\,\vec{\gamma}_2$, so gilt entweder*

1. *$\vec{\gamma}_1, \vec{\gamma}_2 \neq \varepsilon$ und ${}^{(1)}\vec{\gamma}_1 \equiv {}^{(1)}\vec{\gamma}_2$ oder*
2. *$\vec{\gamma}_1 = \vec{\gamma}_2 = \varepsilon$ und $C = C'$.*

Der Beweis ist eine einfache Induktion über die Länge der Ableitung.

Lemma 2.3.17 *Es sei G eine schlanke strikt deterministische Grammatik. Ist dann $C \Rightarrow_L^+ D\,\vec{\alpha}$, so gilt $C \not\equiv D$.*

Beweis. Es sei etwa $C \Rightarrow_L^n D\,\vec{\alpha}$. Dann gilt wegen Lemma 2.3.16 für alle $k \geq 1$: $C \Rightarrow_L^{kn} D\,\vec{\gamma}$ für ein $\vec{\gamma}$. Daraus folgt aber, daß es keine terminierende linksseitige Ableitung aus C gibt. Dies ist ein Widerspruch dazu, daß G schlank ist. $\dashv$

Es folgt daraus, daß eine strikt deterministische Grammatik nicht linksrekursiv ist, das heißt, daß $A \Rightarrow_L^+ A\,\vec{\alpha}$ nicht gelten kann. Wir können daraus wie folgt eine Greibach–Form für G erstellen. Es sei $\rho = C \to \alpha\,\vec{\gamma}$ eine Regel. Ist nun $\alpha \notin A$, so überspringen wir ρ, indem wir ρ durch die Menge der Regeln $C \to \vec{\eta}\,\vec{\gamma}$ ersetzen für $\alpha \to \vec{\eta} \in R$. Das Lemma 2.3.16 sichert uns, daß Π auch für die neue Grammatik eine strikte Partition ist. Diesen Prozeß wiederholen wir so oft wie nötig. Da G nicht linksrekursiv ist, kommen wir zu einem Ende.

Theorem 2.3.18 *Zu jeder strikt deterministischen Grammatik G existiert eine strikt deterministische Grammatik H in Greibach–Form, sodaß $L(G) = L(H)$.*

Nun jedoch zu der versprochenen Korrespondenz zwischen strikt deterministischen Sprachen und strikt deterministischen Grammatiken.

Lemma 2.3.19 *Es sei S strikt deterministisch. Dann existiert ein deterministischer Automat mit einem einzelnen Endzustand, der S über den Stapel akzeptiert.*

Beweis. Es sei $\mathfrak{A}$ gegeben. Wir fügen noch einen neuen Zustand q hinzu, in welchen der Automat wechselt, sobald der Stapel geleert ist. $\dashv$

Lemma 2.3.20 *Es $\mathfrak{A}$ ein deterministischer Automat mit einem einzelnen Endzustand. Dann ist $G(\mathfrak{A})$ strikt deterministisch.*

Beweis. Es sei $\mathfrak{A} = \langle Q, i_0, A, F, D, \#, \delta\rangle$. Wir können nach dem vorangegangenen Lemma annehmen, daß $F = \{q_f\}$. Nun sei $G(\mathfrak{A})$ wie in Proposition 2.3.5 definiert. Setze

$$\alpha \equiv \beta \quad \Leftrightarrow \quad \begin{cases} & \alpha, \beta \in A \\ \textit{oder} & \alpha = [q, Z, q'], \beta = [q, Z, q''] \\ & \textit{für gewisse } q, q', q'' \in Q, Z \in D \end{cases}$$

Wir zeigen, daß $\equiv$ eine strikte Partition ist. Seien dazu $[q, Z, q'] \to \vec{\alpha}\vec{\gamma}_1$ und $[q, Z, q''] \to \vec{\alpha}\vec{\gamma}_2$ zwei Regeln. Sei zunächst einmal $\vec{\gamma}_1, \vec{\gamma}_2 \neq \varepsilon$. Fall 1. $\vec{\alpha} = \varepsilon$. Betrachte $\zeta_i := {}^{(1)}\vec{\gamma}_i$. Ist $\zeta_1 \in A$, so ist auch $\zeta_2 \in A$, da $\mathfrak{A}$ deterministisch ist. Ist aber $\zeta_1 \notin A$, so gilt $\zeta_1 = [q, Y_0, q_1]$ und $\zeta_2 = [q, Y_0, q_1']$ und so $\zeta_1 \equiv \zeta_2$. Fall 2. $\vec{\alpha} \neq \varepsilon$. Sei jetzt $\eta := {}^{(1)}\vec{\alpha}$. Ist $\eta \in A$, so gilt jetzt $\zeta_1 = [q_i, Y_i, q_{i+1}]$ und $\zeta_2 = [q_i, Y_i, q_{i+1}']$ für gewisse $q_i, q_{i+1}, q_{i+1}' \in Q$.

Sei jetzt $\vec{\gamma}_1 = \varepsilon$. Dann ist $\vec{\alpha}\,\vec{\gamma}_1$ ein Präfix von $\vec{\alpha}\,\vec{\gamma}_2$. Fall 1. $\vec{\alpha} = \varepsilon$. Dann ist $\vec{\alpha}\,\vec{\gamma}_2 = \varepsilon$, also $\vec{\gamma}_2 = \varepsilon$. Fall 2. $\vec{\alpha} \neq \varepsilon$. Dann ist leicht zu sehen, daß $\vec{\gamma}_2 = \varepsilon$. Also ist in beiden Fällen $\vec{\gamma}_2 = \varepsilon$, und so $q' = q''$. Dies zeigt die Behauptung. $\dashv$

Theorem 2.3.21 *Es sei S eine strikt deterministische Sprache. Dann existiert eine strikt deterministische Grammatik G mit $L(G) = S$.*

Die Strategie, ein Wort in den Keller zu tun und anschließend herauszuholen, hat zu folgender Begriffsbildung geführt. Ein **Kellerschritt** ist ein Schritt, bei dem die Maschine ein Symbol in den Keller schreibt oder ein Symbol aus dem Keller entnimmt. Der Automat macht einen **Turn**, falls er im letzten Kellerschritt ein Symbol in den Keller getan hat und nun ein Symbol aus dem Keller holt, oder umgekehrt.

Definition 2.3.22 *Eine Sprache S heißt n–**Turn–Sprache**, falls es einen Kellerautomaten gibt, der jede Zeichenkette S in höchstens n Kellerwechseln erkennt. Eine Sprache heißt **ultralinear**, falls sie eine n–Turn–Sprache ist für ein $n \in \omega$.*

Man überlege sich im Übrigen, daß eine kontextfreie Sprache S genau dann n–Turn ist, wenn es einen Kellerautomaten gibt, der S akzeptiert und bei dem *jede* Berechnung aus höchstens n Kellerwechseln besteht. Denn hat man einen Automaten $\mathfrak{K}$, der S erkennt und bei dem für $\vec{x} \in S$ eine Berechnung mit höchstens n Kellerwechseln existiert, so bilde man einen Automaten $\mathfrak{L}$, der $\mathfrak{K}$ simuliert, aber eine Berechnung abbricht, falls sie den $n+1$ten Kellerwechsel macht. Dazu muß man $\mathfrak{K}$ einfach mit einem Gedächtnis ausstatten, welches die Anzahl der Kellerwechsel zählt. Es ist nicht schwer, einen solchen Automaten zu konstruieren.

Wir wollen das Gebiet der ultralinearen Sprachen nicht vertiefen. Ein Spezialfall jedoch ist wichtig, nämlich die 1–Turn–Sprachen. Eine kontextfreie Grammatik ist

linear, falls in jeder Regel $X \to \vec{\alpha}$, $\vec{\alpha}$ höchstens ein nichtterminales Symbol enthält. Eine Sprache heißt **linear**, falls sie durch eine lineare Grammatik erzeugt werden kann.

Theorem 2.3.23 *Eine kontextfreie Sprache S ist genau dann linear, wenn sie 1–Turn ist.*

Beweis. Sei G eine lineare Grammatik. Ohne Beschränkung der Allgemeinheit ist eine Regel von der Form $X \to aY$ oder $X \to Ya$. Ferner gebe es noch Regeln der Form $X \to \varepsilon$. Wir konstruieren folgenden Automaten. $D := \{\#\} \cup A$, $\#$ der Stapelbeginn, $Q := \{X^+ : X \in N\} \cup \{X^- : X \in N\}$, $i_0 := \mathtt{S}^+$, $F := \{\mathtt{S}^-\}$. Ferner sei für $x \in A_\varepsilon$ $\delta(\langle X^+, Z, x\rangle) := \{\langle Y^+, Z\rangle)\}$, falls $X \to xY \in R$; es sei $\delta(\langle Y^-, x, x\rangle) := \{\langle X^-, \varepsilon\rangle\}$, falls $X \to Yx \in R$. Und schließlich ist $\delta(\langle X^+, x, x\rangle) := \{\langle X^-, \varepsilon\rangle\}$, falls $X \to x \in R$. Dies definiert den Automaten $\mathfrak{K}(G)$. Es ist nicht schwer zu sehen, daß $\mathfrak{K}(G)$ nur Berechnungen mit einem Kellerwechsel zuläßt. Denn ist der Automat in Zuständen der Form X^+, so darf der Stapel nicht abnehmen, es sei denn, der Automat wechselt in den Zustand X^-. Ist er in einem Zustand X^-, so darf der Stapel nicht zunehmen, und er kann auch nur in einen Zustand Y^- kommen. Wir überlassen dem Leser den Nachweis, daß $L(\mathfrak{K}(G)) = L(G)$. Daher ist also $L(G)$ eine 1–Turn–Sprache. Sei umgekehrt $\mathfrak{K}$ ein Automat, der nur Berechnung mit einem Kellerwechsel gestattet. Man überlegt sich dann, daß, falls das oberste Symbol eines Stapels einmal gelöscht wird, der Automat nicht mehr auf den Stapel legen darf. Denn der Automat kann als erstes den Stapel aufbauen, um ihn dann wieder zu leeren. Betrachten wir also $G(\mathfrak{K})$, wie oben definiert. Dann sind alle Regeln von der Form $X \to x\vec{Y}$ mit $x \in A_\varepsilon$. Sei $\vec{Y} = Y_0Y_1 \dots Y_{n-1}$. Wir behaupten, daß jede Y_i–Produktion für $i > 0$ von der Form $Y_i \to a$ oder $Y_i \to X$ ist. Falls nicht, so gäbe es eine Berechnung, in der der Automat 2 Kellerwechsel macht, wie wir oben überlegt hatten. (Diese Argument benutzt stillschweigend, daß der Automat auch wirklich eine Berechnung besitzt, in der er den Übergang zu $Y_i = [p, X, q]$ auch wirklich macht, das heiß, daß er von p nach q kommt, wo X das oberste Stapelsymbol ist. Ist dies jedoch nicht der Fall, dann lassen sich die entsprechenden Übergänge aus dem Automaten gefahrlos tilgen.) Es ist nun leicht, die Regeln der Form $Y_i \to X$ zu elimieren, indem man sie überspingt. Anschließendes Überspringen der Regeln $Y_i \to a$ liefert schließlich eine lineare Grammatik. $\dashv$

Die automatentheoretischen Analysen legen den Schluß nahe, daß das Erkennungsproblem für kontextfreie Sprachen sehr kompliziert sein muß. Dies ist jedoch nicht der Fall. Es zeigt sich, daß Erkennung und Analyse in $O(n^3)$ Zeitschritten lösbar sind. Dazu sei G gegeben. Wir nehmen ohne Beschränkung der Allgemeinheit an, G sei in Chomsky–Normalform. Es sei $\vec{x}$ eine Zeichenkette der Länge n. Wir versuchen nun als ersten Schritt, sämtliche Teilworte aufzulisten, welche Konstituenten sind, zusammen mit ihrem Typ. Falls dann $\vec{x}$ eine Konstituente ist vom Typ S, so

ist $\vec{x} \in L(G)$, falls nicht, so $\vec{x} \notin L(G)$. Um die Teilworte aufzuzählen, bedienen wir uns einer $(n+1) \times (n+1)$–Matrix, deren Einträge Teilmengen von N sind. Eine solche Matrix heißt ein **Chart**. Jedes Teilwort ist durch ein Paar $\langle i, j\rangle$ von Zahlen bestimmt, wo $0 \leq i < j \leq n+1$. In das Feld $\langle i, j\rangle$ tragen wir alle $X \in N$ ein, für die das Teilwort $x_i x_{i+1} \dots x_{j-1}$ eine Konstituente vom Typ X ist. Zu Beginn der Berechnungen ist die Matrix leer. Nun beginnen wir, die Matrix aufzufüllen, und zwar induktiv über die Länge des Teilwortes, d. h. die Differenz $d := j - i$ und induktiv über i. Es ist $0 < d < n+1$. Wir beginnen also mit $d = 1$, $i = 0$. Danach kommt $i = 1$, …, $i = n$, und dann $d = 2$ und $i = 0$, $i = 1$ etc. Wir betrachten ein Paar $\langle d, i\rangle$. Das Teilwort $x_i \dots x_{i+d}$ ist eine Konstituente vom Typ X genau dann, wenn es in Teilworte $\vec{y} = x_i \dots x_{i+e}$ und $\vec{z} = x_{i+e+1} \dots x_{i+d}$ dergestalt zerfällt, daß es eine Regel $X \to YZ$ gibt, wo $\vec{y}$ eine Konstituente vom Typ Y und $\vec{z}$ eine Konstituente vom Typ Z ist. Das bedeutet, die Menge der $X \in N$, welche wir bei $\langle i, i+d\rangle$ eintragen, berechnet sich aus allen echten Zerlegungen des entsprechenden Teilwortes. Es gibt $d - 1 \leq n$ solcher Zerlegungen. Für jede Zerlegung ist der Rechenaufwand beschränkt und hängt nur noch von einer Konstante c_G ab, deren Wert von der Grammatik bestimmt ist. Für jedes Paar gibt es also $c_G \cdot (n+1)$ Rechenschritte. Nun existieren $\binom{n}{2}$ echte Teilworte. Also ist der Aufwand sicher beschränkt durch $c_G \cdot n^3$.

In Figur 2.1 ist die Berechnung eines Charts durchgeführt anhand der folgenden Grammatik und des Wortes `abaabb`. Da die Grammatik invertierbar ist, hat ein Teilwort höchstens einen Typ. Dies muß im Allgemeinen natürlich nicht so sein. (Man kann sich jedoch wegen Satz 2.2.13 immer auf eine invertierbare Grammatik zurückziehen.)

$$\begin{array}{lcl} \texttt{S} & \to & \texttt{SS} \mid \texttt{AB} \mid \texttt{BA} \\ \texttt{A} & \to & \texttt{AS} \mid \texttt{SA} \mid \texttt{a} \\ \texttt{B} & \to & \texttt{BS} \mid \texttt{SB} \mid \texttt{b} \end{array}$$

Wir können die Berechnung eines Charts wie folgt zusammenfassen. Es sei $C_{\vec{x}}(i, j)$ die Menge aller Nichtterminalsymbole A derart, daß $A \vdash_G x_i x_{i+1} \dots x_{j-1}$. Ferner sei für zwei Nichtterminalsymbole X und Y $X \odot Y := \{Z : Z \to XY \in R\}$, und für Mengen $\mathbb{U}, \mathbb{V} \subseteq N$ sei

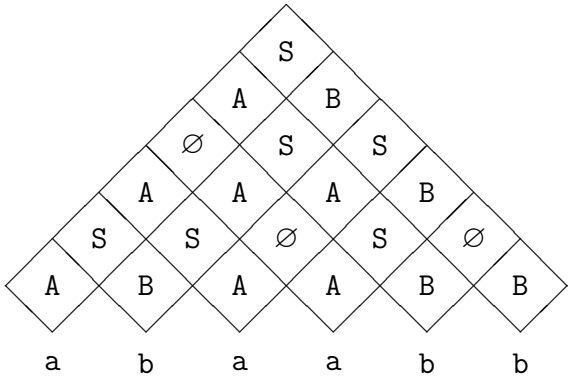
$$\mathbb{U} \odot \mathbb{V} := \bigcup \langle X \odot Y : X \in \mathbb{U}, Y \in \mathbb{V}\rangle$$

Nun können wir $C_{\vec{x}}(i, j)$ induktiv ausrechnen. Der Induktionsparameter ist $j - i$. Ist $j - i = 1$, so ist $C_{\vec{x}}(i, j) = \{X : X \to x \in R\}$. Ist $j - i > 1$, so gilt folgende Gleichung:

$$C_{\vec{x}}(i, j) = \bigcup_{i<k<j} C_{\vec{x}}(i, k) \odot C_{\vec{x}}(k, j)$$

Dabei ist stets $j - k, k - i < j - i$. Nun sei entschieden, daß $\vec{x} \in L(G)$. Wie kann man eine Derivation für $\vec{x}$ finden? Dazu bedienen wir uns des fertigen Charts. Wir beginnen mit $\vec{x}$ und zerlegen es auf irgendeine Weise; da $\vec{x}$ den Typ `S` hat, muß es

Abbildung 2.1: Ein Chart für `abaabb`



eine Regel $\mathtt{S} \to XY$ geben und eine Zerlegung in $\vec{x}$ vom Typ X und $\vec{y}$ vom Typ Y. Oder aber $\vec{x} = a \in A$ und $\mathtt{S} \to a$ ist eine Regel. Ist die Zerlegung gefunden, so fährt man mit den Teilworten $\vec{x}$ und $\vec{y}$ auf die gleiche Weise fort. Jede Zerlegung beansprucht eine Zeit, welche nur von G abhängt. Ein Teilwort der Länge i hat $i \leq n$ Zerlegungen. Auf unserem Weg werden wir insgesamt höchstens $2n$ Teilworte analysieren. Dies folgt daraus, daß es in einem echt verzweigenden Baum mit n Blättern höchstens $2n$ Knoten geben kann. Insgesamt haben wir einen Bedarf an Zeit von $d_G \cdot n^2$ für eine gewisse, nur von G abhängige Konstante d_G.

Daraus folgt nun, daß man im allgemeinen Fall, wenn die Grammatik nicht in 2–Standardform ist, zur Erkennung und zur Analyse höchstens $O(n^3)$ Zeitschritte braucht bei gleichzeitigem Platzbedarf von $O(n^2)$. Denn sei G gegeben. Man transformiere G in 2–Standardform in die Grammatik G^2. Da $L(G^2) = L(G)$, ist das Erkennungsproblem für G in der gleichen Zeit lösbar wie für G^2. Man benötigt $O(n^2)$ Zeitschritte, um von einem Chart für $\vec{x}$ einen Baum zu konstruieren. Man benötigt noch einmal $O(n^2)$ Zeitschritte, um daraus einen Baum für G zu machen, und $O(n)$ Schritte, um daraus eine Derivation zu machen.

Dies ist allerdings noch kein Beweis, daß das Problem die Zeitkomplexität $O(n^3)$ und Platzbedarf $O(n^2)$ hat, denn wir müssen nun eine Turingmaschine finden, welche das Problem in der gegebenen Zeit bzw. mit dem gegebenem Platz löst. Dies ist möglich; das wurde unabhängig von Cocke, Kasami und Younger gezeigt.

Theorem 2.3.24 (Cocke, Kasami, Younger) *Kontextfreie Sprachen haben die folgende Komplexität.*

1. CFS $\subseteq$ *DTIME*(n^3).
2. CFS $\subseteq$ *DSPACE*(n^2).

Beweis. Wir konstruieren eine deterministische 3–Band Turingmaschine, welche nur $O(n^2)$ Platz und $O(n^3)$ Zeit benötigt. Der wesentliche Trick besteht in der Beschriftung der Bänder. Wir benötigen außer dem Alphabet A noch zwei Hilfssymbole `B` und `Q`, sowie ein Symbol $[U]$ für jedes $U \subseteq N$ und ein Symbol $[U]^{\surd}$. Wir haben auf Band 1 das Eingabewort, $\vec{x}$. Es sei $C(i,j) := C_{\vec{x}}(i,j)$. Es habe $\vec{x}$ die Länge n. Auf Band 1 erzeugen wir nun eine Sequenz der folgenden Form:

$$\mathtt{QB}^n\mathtt{QB}^{n-1}\mathtt{Q}\ldots\mathtt{QBBQBQ}$$

Dies ist das Skelett des Charts. Wir nennen eine Sequenz von `B`s zwischen zwei `Q`s einen *Block*. Der erste Block wird nun wie folgt gefüllt. Das Wort $\vec{x}$ wird schrittweise gelöscht, und die Sequenz $\mathtt{B}^n$ wird dabei ersetzt durch die Sequenz der $C(i,i+1)$. Dies Prozedur erfordert $O(n^2)$ Schritte. Für jedes d von 1 bis $n-1$ werden wir nun den $d+1$ten Block auffüllen. Sei also d gegeben. Wir schreiben auf Band 2 die Sequenz

$$\begin{array}{l}\mathtt{Q}[C(0,1)][C(0,2)]\ldots[C(0,d)]\\ \quad\mathtt{Q}[C(1,2)][C(1,3)]\ldots[C(1,d+1)]\\ \quad\quad\ldots\\ \quad\mathtt{Q}[C(n-d,n-d+1)][C(n-d,n-d+2)]\ldots[C(n-d,n)]\mathtt{Q}\end{array}$$

Auf das Band 3 schreiben wir die Sequenz

$$\begin{array}{l}\mathtt{Q}[C(0,d)][C(1,d)]\ldots[C(d-1,d)]\\ \quad\mathtt{Q}[C(1,d+1)][C(2,d+1)]\ldots[C(d,d+1)]\\ \quad\quad\ldots\\ \quad\mathtt{Q}[C(n-d,n)][C(n-d+1,n)]\ldots[C(n-1,n)]\mathtt{Q}\end{array}$$

Aus diesen Sequenzen läßt sich der $d+1$te Block schnell berechnen. Der Automat muß den ersten Block auf Band 2 und den zweiten Block auf Band 3 gleichmäßig durchlaufen und sich jeweils das Ergebnis von $C(0,j)\cdot C(j,d+1)$ merken. Ist er am Ende angekommen, hat er $C(0,d+1)$ berechnet und kann das Ergebnis auf Band 1 sofort eintragen. Nun geht er auf dem zweiten Band und dem dritten Band die folgenden Blöcke gleichmäßig durch und berechnet so $C(1,d+2)$. Und so fort. Es ist klar, daß die Berechnung linear von der Länge des Bandes 2 (und des Bandes 3) abhängt, also $O(n^2)$ Zeit benötigt. Am Ende dieses Vorgangs wird Band 2 und Band 3 gelöscht. Auch dies benötigt nur quadratisch viel Zeit. Also Letztes muß man sich noch überlegen, daß das Beschriften der Bänder 2 und 3 auch nur $O(n^2)$ viel Zeit benötigt. Dann wäre für jedes d der Zeitaufwand $O(n^2)$, und damit insgesamt der Aufwand mit $O(n^3)$ gegeben.

Dazu schreiben wir zuerst Q und positionieren wir den Kopf des Bandes 1 auf das Element $[C(0,1)]$. Wir schreiben $[C(0,1)]$ auf Band 2 und $[C(0,1)]^{\surd}$ auf Band 1. (Wir nennen das 'abticken' des Symbols. Dies dient als Merkhilfe.) Nun rücken wir vor zu $[C(1,2)]$, kopieren dies auf Band 2 und ersetzen es durch $[C(1,2)]^{\surd}$. Und so weiter. Dies benötigt nur linear viel Zeit, denn die Symbole $[C(i,i+1)]$ erkennen wir daran, daß sie rechts von Q stehen. Sind wir fertig, schreiben wir auf Band 2 noch Q und gehen auf Band 1 an den Anfang und dann zum ersten Symbol rechts von einem 'getickten' Symbol. Dies ist $[C(1,2)]$. Wir kopieren es auf Band 2 und ticken es ab. Nun weiter zum nächsten Symbol rechts von einem getickten Symbol, kopieren und ticken. Auf diese Weise entsteht Band 2 in quadratischer Zeit. Als letzten Schritt werden die getickten Symbole auf Band 1 wieder 'enttickt'. Auch dies benötigt $O(n^2)$ viel Zeit. Analog entsteht die Beschriftung auf Band 3. $\dashv$

Übung 41. Beweisen Sie Proposition 2.3.2.

Übung 42. Beweisen Sie Satz 2.3.8. *Hinweis.* Zeigen Sie, daß die Anzahl der ε–Bewegungen eines Automaten $\mathfrak{A}$ beim Lesen einer Zeichenkette $\vec{x}$ durch $k_{\mathfrak{A}} \cdot |\vec{x}|$ beschränkt ist, wobei $k_{\mathfrak{A}}$ eine von $\mathfrak{A}$ abhängige Zahl ist. Kodieren Sie anschließend die Arbeitsweise eines beliebigen Kellerautomaten mit Hilfe einer 2–Band–Turingmaschine und zeigen Sie, daß jedem Schritt des Kellerautomaten eine beschränkte Anzahl von Schritten der Turingmaschine entspricht.

Übung 43. Zeigen Sie, daß eine kontextfreie Sprache genau dann 0–Turn ist, wenn sie regulär ist.

Übung 44. Man gebe ein Verfahren an, wie man einen Chart auf das Band einer Turingmaschine kodieren kann.

Übung 45. Man skizziere die Arbeitsweise einer deterministischen Turingmaschine, welche eine gegebene kontextfreie Sprache in $O(n^2)$ Platz erkennt.

Übung 46. Man zeige: $\{\vec{w}\,\vec{w}^T : \vec{w} \in A^*\}$ ist kontextfrei aber nicht deterministisch.

Übung 47. Man konstruiere einen deterministischen Automaten, welcher eine gegebene Dyck–Sprache erkennt.

Übung 48. Beweisen Sie Satz 2.3.13.

2.4 Ambiguität, Transparenz und Parsingstrategien

In diesem Abschnitt werden wir das Verhältnis zwischen Zeichenketten einer Grammatik und Bäumen beleuchten. Wie in Abschnitt 1.6 erläutert, besteht eine eineindeutige Beziehung zwischen Ableitungen in G und Ableitungen in der zugehörigen Baumgrammatik. Also definiert eine Ableitung $\Delta = \langle \vec{\alpha}_i : i < p \rangle$ in G einen Baum $\mathfrak{B}$

mit Marken in $N \cup A$, dessen assoziierte Zeichenkette genau $\vec{\alpha}_{p-1}$ ist. Ist die letzte Zeichenkette der Ableitung nicht terminal, dann sind die Marken der Blätter nicht unbedingt terminal. Solche Bäume heißen *partiell*.

Definition 2.4.1 *Es sei G eine kontextfreie Grammatik. $\vec{\alpha}$ heißt eine* ***G–Konstituente vom Typ*** *A, falls $A \vdash_G \vec{\alpha}$. Ist $\mathfrak{B}$ ein Baum mit assoziierter Zeichenkette $\vec{x}$. Sei $\vec{y}$ ein Teilwort von $\vec{x}$, $\vec{x}$ eine G–Konstituente vom Typ A und $D(\vec{y})$. Das Vorkommen von $\vec{y}$ heißt* ***akzidentielle Konstituente vom Typ*** *A in $\mathfrak{B}$, falls das Vorkommen von $\vec{y}$ in D keine Konstituente vom Typ A in $\mathfrak{B}$ ist.*

Ein Beispiel soll diese Begriffsbildung illustrieren. Sei G die folgende Grammatik.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{SS} \mid \mathtt{AB} \mid \mathtt{BA} \\ \mathtt{A} & \to & \mathtt{AS} \mid \mathtt{SA} \mid \mathtt{a} \\ \mathtt{B} & \to & \mathtt{BS} \mid \mathtt{SB} \mid \mathtt{b} \end{array}$$

Die Zeichenkette $\vec{x} = \mathtt{abaabb}$ hat mehrere Ableitungen, welche unter anderem den folgenden Klammerungen entsprechen:

$$(\mathtt{a}(\mathtt{b}(\mathtt{a}((\mathtt{ab})\mathtt{b})))), \quad ((\mathtt{ab})(((\mathtt{a}(\mathtt{ab}))\mathtt{b})))$$

Wir geben eine vollständige Liste von G–Konstituenten an, welche in $\vec{x}$ vorkommen:

$$\begin{array}{l} \mathtt{A} : \mathtt{a}, \mathtt{aab}, \mathtt{aba}, \mathtt{baa}, \mathtt{abaab} \\ \mathtt{B} : \mathtt{b}, \mathtt{abb} \\ \mathtt{S} : \mathtt{ab}, \mathtt{aabb}, \mathtt{abaabb} \end{array}$$

Manche der Konstituenten kommen mehrfach vor, zum Beispiel $\mathtt{ab}$ in $\langle \varepsilon, \mathtt{aabb} \rangle$ und $\langle \mathtt{aba}, \mathtt{b} \rangle$. Nehmen wir die erste Klammerung, $(\mathtt{a}(\mathtt{b}(\mathtt{a}((\mathtt{ab})\mathtt{b}))))$. Die Konstituenten sind demnach $\mathtt{a}$ (Kontexte: $\langle \varepsilon, \mathtt{baabb} \rangle$, $\langle \mathtt{ab}, \mathtt{abb} \rangle$, $\langle \mathtt{aba}, \mathtt{bb} \rangle$), $\mathtt{b}$, $\mathtt{ab}$ im Kontext $\langle \mathtt{aba}, \mathtt{b} \rangle$, $\mathtt{abb}$ im Kontext $\langle \mathtt{aba}, \varepsilon \rangle$, $\mathtt{aabb}$, $\mathtt{baabb}$ und $\mathtt{abaabb}$. Dies sind also die Konstituenten des Baumes. Das Vorkommen $\langle \mathtt{aba}, \mathtt{b} \rangle$ von $\mathtt{ab}$ in $\mathtt{ababb}$ ist demnach ein akzidentielles Vorkommen einer G–Konstituente vom Typ $\mathtt{S}$. Man beachte, daß es auch vorkommen kann, daß $\vec{y}$ eine Konstituente des Baumes $\mathfrak{B}$ ist, aber daß sie als G–Konstituente vom Typ C akzidentiell auftritt, weil sie in $\mathfrak{B}$ den Typ $D \neq C$ hat.

Definition 2.4.2 *Eine Grammatik G heißt* ***transparent****, falls keine G–Konstituente in einem Wort akzidentiell auftritt. Eine Grammatik, welche nicht transparent ist, heißt* ***opak****. Eine Sprache heißt* ***inhärent opak****, falls es keine transparent Grammatik gibt, die sie erzeugt.*

Ein Beispiel mag dies illustrieren. Die Polnische Notation läßt sich durch eine transparente Grammatik erzeugen. Für jedes n–stellige f seien die Regeln

$$\mathtt{S} \to \mathtt{F}_n\mathtt{S}\ldots\mathtt{S} \quad (n\text{–mal}), \qquad \mathtt{F}_n \to f$$

aufgenommen, sowie $\mathtt{S} \to x$ für jedes $x \in X$. Dies definiert die Grammatik PN_Ω. Sei $\vec{x} = x_0 x_1 \dots x_{n-1}$ eine Zeichenkette. Dann sei $\gamma(\vec{x}) := \sum_{i<n} \gamma(x_i)$, wobei $\gamma(x) := -1$, $\gamma(f) := \Omega(f) - 1$.

Lemma 2.4.3 *Genau dann ist $\vec{x}$ ein Term, wenn $\gamma(\vec{x}) = -1$ und wenn für alle echten Präfixe $\vec{y}$ von $\vec{x}$ gilt $\gamma(\vec{y}) \geq 0$.*

Aus diesem Satz folgt, daß ein echtes Präfix eines Termes kein Term ist. Habe also $\vec{x}$ ein akzidentielles Vorkommen eines Terms $\vec{y}$. Dann überschneidet sich $\vec{y}$ echt mit einer Konstituente $\vec{z}$. Ohne Beschränkung der Allgemeinheit sei $\vec{y} = \vec{u} \cdot \vec{v}$ und $\vec{z} = \vec{v} \cdot \vec{w}$. Es folgt nun $\gamma(\vec{v}) = \gamma(\vec{y}) - \gamma(\vec{u}) < 0$, da ja $\gamma(\vec{u}) \geq 0$. Es existiert also ein echtes Präfix $\vec{u}_1$ von $\vec{u}$ mit $\vec{u}_1 = -1$. (Dazu muß man sich überlegen, daß die Menge aller $\gamma(\vec{p})$, für $\vec{p}$ Präfix eines gegebenen Termes, konvex ist.)

Theorem 2.4.4 *Die Grammatik PN_Ω für die Polnische Notation ist transparent.*

Man betrachte nun die Sprachen $\mathtt{a}^+\mathtt{b}$ und $\mathtt{a}^+$. Beide sind regulär. Eine reguläre Grammatik, welche $\mathtt{a}^+\mathtt{b}$ erzeugt, ist notwendig transparent. Denn Konstituenten sind: $\mathtt{a}$ und $\mathtt{a}^+\mathtt{b}$. Keine Konstituente tritt akzidentiell auf. Betrachte nun auf der anderen Seite die Sprache $\mathtt{a}^+$. Die rechtslineare Grammatik, welche diese Sprache erzeugt, ist opak. Denn sie erzeugt alle $\mathtt{a}^+$, welche also Kontituenten sind. $\mathtt{aa}$ tritt aber in $\mathtt{aaa}$ akzidentiell auf. Wir können zeigen, daß $\mathtt{a}^+$ nicht nur keine transparente reguläre Grammatik besitzt, sondern sogar inhärent opak ist.

Proposition 2.4.5 $\mathtt{a}^+$ *ist inhärent opak.*

Beweis. Angenommen, es existiert eine kontextfreie Grammatik G, welche a^* erzeugt. Es sei $K := \{\mathtt{a}^k : \mathtt{a}^k \text{ ist Konstituente}\}$. Es gibt mindestens zwei Elemente in K, etwa $\mathtt{a}^p$ und $\mathtt{a}^q$, $q > p$. Nun existieren zwei Vorkommen von $\mathtt{a}^p$ in $\mathtt{a}^q$, welche überlappen. Eines dieser Vorkommen muß demnach akzidentiell sein. $\dashv$

Man überlegt sich, daß wenn S transparent ist und $\varepsilon \in S$, so muß bereits $S = \{\varepsilon\}$ sein. Es ist leicht zu sehen, daß eine Sprache über einem Buchstaben nur dann transparent sein kann, wenn sie nicht mehr als ein Wort umfaßt. Viele Eigenschaften kontextfreier Grammatiken sind unentscheidbar. Für Transparenz ist dies jedoch nicht der Fall.

Theorem 2.4.6 (Fine) *Es sei G eine kontextfreie Grammatik. Dann ist entscheidbar, ob G transparent ist oder nicht.*

Beweis. Der Beweis folgt im Wesentlichen aus dem nachstehenden Sätzen. Denn sei n das Maximum aller Längen von rechten Seiten einer Produktion. Sei k_G die Konstante des Pumplemmas. Diese läßt sich effektiv bestimmen. Nach Lemma 2.4.7 existiert ein akzidentielles Vorkommen einer Konstituente genau dann, wenn es ein akzidentielles Vorkommen einer rechten Seite einer Produktion gibt. Diese haben die Länge $p + 1$, wo p die maximale Produktivität con Regeln aus G ist. Ferner müssen wir wegen Lemma 2.4.9 nur solche Konstituenten auf akzidentielle Vorkommen überprüfen, deren Länge $p^2 + p$ nicht übersteigt. Dies kann in endlich vielen Schritten geschehen. ⊣

Lemma 2.4.7 *Genau dann ist G opak, falls es eine Produktion $\rho = A \to \vec{\alpha}$ gibt, sodaß $\vec{\alpha}$ akzidentiell auftritt.*

Beweis. Es sei $\vec{\varphi}$ ein Zeichenkette minimaler Länge, welche akzidentiell auftritt. Sei etwa C ein akzidentielles Vorkommen von $\vec{\varphi}$. Es sei ferner $\vec{\varphi} = \vec{\gamma}_1\vec{\alpha}\vec{\gamma}_2$, und $A \to \vec{\alpha}$ eine Regel. Dann existieren zwei Fälle. (A) Das Vorkommen von $\vec{\alpha}$ ist akzidentiell. Dann haben wir einen Widerspruch zur Minimalität von $\vec{\varphi}$. (B) Das Vorkommen von $\vec{\alpha}$ ist nicht akzidentiell. Dann tritt $\vec{\gamma}_1 A\vec{\gamma}_2$ ebenfalls akzidentiell in C auf! (Wir können dann nämlich die Ersetzung $A \to \vec{\alpha}$ in der Zeichenkette, die $\vec{\varphi}$ enthält, rückwärts anwenden, da $\vec{\alpha}$ eine Konstituente ist.) Auch dies widerspricht der Minimalität von $\vec{\varphi}$. Also ist $\vec{\varphi}$ eine rechte Seite einer Produktion. ⊣

Lemma 2.4.8 *Es sei G eine kontextfreie Grammatik ohne Regeln der Produktivität -1 und $\vec{\alpha}$, $\vec{\gamma}$ Zeichenketten. Es sei $\vec{\gamma}$ eine Konstituente vom Typ A in welchem $\vec{\alpha}$ akzidentiell auftritt. Ferner sei $\vec{\gamma}$ minimal in dem folgenden Sinne: es existiert kein $\vec{\delta}$ vom Typ A mit (1) $|\vec{\delta}| < |\vec{\gamma}|$ und (2) $\vec{\delta} \vdash_G \vec{\gamma}$ und (3) $\vec{\alpha}$ tritt akzidentiell in $\vec{\delta}$ auf. Dann schneidet jede Konstituente der Länge > 1 von $\vec{\gamma}$ das akzidentielle Vorkommen von $\vec{\alpha}$.*

Beweis. Es sei $\vec{\gamma} = \vec{\sigma}_1\,\vec{\eta}\,\vec{\sigma}_2$, $|\vec{\eta}| > 1$, und dieses Vorkommen von $\vec{\eta}$ eine Konstituente vom Typ A, welche $\vec{\alpha}$ nicht schneidet. Dann ist $\vec{\alpha}$ akzidentiell in $\vec{\delta} := \vec{\sigma}_1\,A\,\vec{\sigma}_2$. Ferner ist $|\vec{\delta}| < |\vec{\gamma}|$, im Widerspruch zur Minimalität von $\vec{\gamma}$. ⊣

Lemma 2.4.9 *Es sei G eine kontextfreie Grammatik mit Regeln der Produktivität mindestens 0 und höchstens p und $\vec{\alpha}$ eine Zeichenkette der Länge n, welche akzidentiell auftritt. Dann existiert eine Konstituente $\vec{\gamma}$ der Länge $\leq np$, in welcher $\vec{\alpha}$ akzidentiell auftritt.*

Beweis. Sei $A \vdash_G \vec{\gamma}$ minimal im Sinne des vorigen Lemmas. Dann gilt, daß jede Konstituente aus $\vec{\gamma}$ der Länge > 1 $\vec{\alpha}$ echt schneidet. Also wurde $\vec{\gamma}$ mit Hilfe von

höchstens n Anwendungen von Regeln der Produktivität > 0 erzeugt. Daher gilt $|\vec{\gamma}| \leq np$. $\dashv$

Die Eigenschaft der Transparenz ist stärker als die der eindeutigen Lesbarkeit, welche wie folgt definiert wird.

Definition 2.4.10 *Eine Grammatik heißt* ***eindeutig****, falls es zu jeder Zeichenkette $\vec{x}$ höchstens einen G–Baum mit assoziierter Zeichenkette $\vec{x}$ gibt. Ist G nicht eindeutig, so heißt G* ***mehrdeutig*** *oder* ***ambig****. Eine kontextfreie Sprache S heißt* ***inhärent ambig****, falls jede S erzeugende Grammatik mehrdeutig ist.*

Proposition 2.4.11 *Eine transparente Grammatik ist eindeutig.*

Es existieren inhärent mehrdeutige Sprachen. Der folgende Satz gibt ein ein Beispiel.

Theorem 2.4.12 (Parikh) *Die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m : n, m \in \omega\} \cup \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n : n, m \in \omega\}$ ist inhärent ambig.*

Beweis. Die Sprache ist kontextfrei, und so existiert eine kontextfreie Grammatik G mit $L(G) = S$. Wir werden zeigen, daß G ambig ist. Es gibt eine Konstante k_G, die das Pumplemma erfüllt. Es sei $n := k_G!$. Dann existiert eine Zerlegung von $\mathtt{a}^{2n}\mathtt{b}^{2n}\mathtt{c}^{3n}$ in

$$\vec{u}_1\,\vec{x}_1\,\vec{v}_1\,\vec{y}_1\,\vec{z}_1$$

derart, daß das Vorkommen von $\vec{x}_1\,\vec{v}_1\,\vec{y}_1$ nicht mehr als k_G vom linken Rand der Zeichenkette entfernt ist. Ferner können wir es so einrichten, daß $\vec{v}_1$ die Länge höchstens k_G hat. Wie wie schon früher gesehen haben, kann $\vec{x}_1\,\vec{y}_1$ nicht $\mathtt{a}$, $\mathtt{b}$ und $\mathtt{c}$ zugleich enthalten. Ferner besteht die Zeichenkette nicht nur aus $\mathtt{c}$. Es ist dann $\vec{x}_1 = \mathtt{a}^p$ und $\vec{y} = \mathtt{b}^p$ für ein gewisses p. Wir betrachten nun eine maximale Konstituente der Form $\mathtt{a}^q\,\mathtt{b}^{q'}$. Eine solche muß offensichtlich existieren. Darin ist eine Konstituente der Form $\mathtt{a}^{q-i}\,\mathtt{b}^{q'-i}$ enthalten für $i < k_G$. Dies folgt wieder nach dem Pumplemma. Also können wir $\mathtt{a}^i$ und $\mathtt{b}^i$ gleichzeitig hochpumpen und bekommen so Zeichenketten der Form

$$\mathtt{a}^{2p+ki}\,\mathtt{b}^{2p+ki}\,\mathtt{c}^{3q}$$

wobei eine Konstituente der Form $\mathtt{a}^{2p+ki-r}\,\mathtt{b}^{2p+ki-s}$ für gewisse $r, s \leq k_G$ existiert. Insbesondere erhalten wir daraus für $k = p/i$

$$\mathtt{a}^{3p}\,\mathtt{b}^{3p}\,\mathtt{c}^{3q}$$

Nun bilden wir eine Zerlegung von $\mathtt{a}^{3n}\mathtt{b}^{2n}\mathtt{c}^{2n}$ in

$$\vec{u}_2\,\vec{x}_2\,\vec{v}_2\,\vec{y}_2\,\vec{z}_2$$

derart, daß das Vorkommen von $\vec{x}_2\,\vec{v}_2\,\vec{y}_2$ nicht mehr als k_G vom linken Rand der Zeichenkette entfernt ist und $\vec{v}_2$ die Länge $\leq k_G$ hat. Analog bekommen wir dann eine Konstituente der Form $\mathtt{b}^{2p-s'}\,\mathtt{c}^{2p-r'}$ für gewisse $r', s' \leq k_G$. Falls nun G nicht ambig ist, sind beide gleichzeitig Konstituenten in einem Baum für diese Zeichenkette. Aber diese Konstituenten überlappen sich. Denn es enthält die linke Konstituente $3p - s$ viele $\mathtt{b}$, und die rechte enthält $3p - s'$ viele $\mathtt{b}$. Da $3p - s + 3p - s' = 6p - (s + s') > 3p$, müssen sich diese Konstituenten überlappen. Sie sind aber offensichtlich nicht gleich. Das ist jedoch nicht möglich. Also ist G ambig. Da G beliebig war, ist S inhärent ambig. $\dashv$

Wir kommen nun zu einer Eigenschaft, die eigentlich das genaue Gegenteil der Eindeutigkeit ist. Sie besagt, daß, falls in $\vec{x}$ ein Vorkommen eines Teilworts $\vec{y}$ eine G–Konstituente vom Typ B ist, so existiert auch ein Baum, in welchem dieses Vorkommen eine Konstituente vom Typ B ist. Dies können wir auch so formulieren.

Definition 2.4.13 *Eine kontextfreie Grammatik hat die* ***NTS–Eigenschaft****, falls aus $C \vdash_G \vec{\alpha}$, $\vec{\alpha} = \vec{\alpha}_1 \cdot \vec{\beta} \cdot \vec{\alpha}_2$ und $B \to \vec{\beta} \in R$ folgt: $C \vdash_G \vec{\alpha}_1 \cdot B \cdot \vec{\alpha}_2$. Eine Sprache heißt* ***NTS–Sprache****, falls sie durch eine NTS–Grammatik erzeugt werden kann.*

Die folgende Grammatik ist zum Beispiel nicht NTS.

$$\mathtt{X} \to \mathtt{aX}, \quad \mathtt{X} \to \mathtt{a}$$

Denn zwar ist $\mathtt{X} \vdash \mathtt{aa}$, aber es gilt nicht $\mathtt{X} \vdash \mathtt{Xa}$. Im allgemeinen sind reguläre Grammatiken nicht NTS. Es gilt jedoch

Theorem 2.4.14 *Alle regulären Sprachen sind NTS–Sprachen.*

Beweis. Es sei $\mathtt{S}$ regulär. Dann existiert ein endlicher Automat $\mathfrak{A} = \langle Q, q_0, F, \delta\rangle$ mit $S = L(\mathfrak{A})$. Nun sei $N := \{S^\star\} \cup \{L(p,q) : p, q \in Q\}$, das Startsymbol sei $\mathtt{S}^\star$. Die Regeln sind

$$\begin{array}{lcll} \mathtt{S}^\star & \to & \mathtt{L}(q_0, q) & q \in F \\ \mathtt{L}(p,q) & \to & \mathtt{L}(p,r)\mathtt{L}(r,q) & \\ \mathtt{L}(p,q) & \to & a & q \in \delta(p,a) \end{array}$$

Dann gilt $\mathtt{L}(p,q) \vdash_G \vec{x}$ genau dann, wenn $q \in \delta(p, \vec{x})$, wie man durch Induktion bestätigt. Hieraus folgt $\mathtt{S}^\star \vdash_G \vec{x}$ genau dann, wenn $\vec{x} \in L(\mathfrak{A})$. Also ist $L(G) = S$. Es bleibt also zu zeigen, daß G die NTS–Eigenschaft hat. Dazu sei $\mathtt{L}(p,q) \vdash_G \vec{\alpha}_1 \cdot \vec{\beta} \cdot \vec{\alpha}_2$ und $\mathtt{L}(r,s) \vdash_G \vec{\beta}$. Zu zeigen ist $\mathtt{L}(p,q) \vdash_G \vec{\alpha}_1 \cdot \mathtt{L}(r,s) \cdot \vec{\alpha}_2$. Dazu erweitern wir den Automaten $\mathfrak{A}$ zu einem Automaten, der Zeichenketten aus $N \cup A$ liest. Dabei ist $q \in \delta(p, C)$ genau dann, wenn für jede Zeichenkette $\vec{y}$ mit $C \vdash_G \vec{y}$ gilt $q \in \delta(p, \vec{y})$.

Dann ist klar, daß $q \in \delta(p, \mathtt{L}(p,q))$. Dann gilt weiterhin $\mathtt{L}(p,q) \vdash_G \vec{\alpha}$ genau dann, wenn $q \in \delta(p, \vec{\alpha})$. Also ist $r \in \delta(p, \vec{\alpha}_1)$ und $q \in \delta(s, \vec{\alpha}_2)$. Hieraus folgt $\mathtt{L}(p,q) \vdash_G \mathtt{L}(p,r)\mathtt{L}(r,s)\mathtt{L}(s,q)$ und schließlich $\mathtt{L}(p,q) \vdash_G \vec{\alpha}_1\mathtt{L}(r,s)\vec{\alpha}_2$. $\dashv$

Hat eine Grammatik die NTS–Eigenschaft, so lassen sich die Zeichenketten sehr schnell daraufhin überprüfen, ob sie von der Grammatik erzeugt werden können. Wir skizzieren die Arbeitsweise eines Kellerautomaten, der $L(G)$ erkennt. Von links nach rechts kommend, werden die Symbole in den Keller getan. Der Automat merkt sich mit Hilfe seiner Zustände die Beschaffenheit des Kellers bis zu der Tiefe κ, wo κ die Länge der längsten rechten Seite einer Regel ist. Ist eine rechte Seite einer Produktion gefunden, so wird sie aus dem Keller gelöscht und das Nichtterminalsymbol ersetzt. In diesem Moment muß der Automat allerdings den Keller κ viele Symbole tief hervorholen und sich so wieder vergewissern, welche die letzten κ Symbole sind. Wiederum reicht dazu ein endliches Gedächtnis aus. Anschließend wird die Zeichenkette wieder in den Keller getan, und der Automat arbeitet wie beschrieben weiter. Wichtig ist, daß das Reduzieren des Kellers immer dann vorgenommen wird, wenn es möglich ist.

Theorem 2.4.15 *Es sei G eine NTS–Grammatik. Dann ist G deterministisch. Insbesondere ist das Erkennungs– und Analyseproblem in DTIME(n).*

Wir wollen dieses Resultat vertiefen. Dazu abstrahieren wir ein wenig von dem Kellerautomaten und führen einen Kalkül ein, der Paare $\vec{\alpha} \vdash \vec{x}$ manipuliert. Hierbei möge man bei $\vec{\alpha}$ an den Keller des Automaten denken und bei $\vec{x}$ an die Zeichenkette rechts vom Lesekopf. Es ist im Übrigen nicht zwingend erforderlich, auf der rechten Seite nur terminale Zeichenketten zu betrachten, aber die Verallgemeinerung auf beliebige Zeichenketten ist einfach. Die folgende Operation, **Schieben** genannt, simuliert das Lesen des ersten Zeichens in den Keller.

$$\text{Schieben:} \qquad \frac{\vec{\eta} \ \vdash x\vec{y}}{\vec{\eta}x \vdash \vec{y}}$$

Eine andere Operation ist das **Reduzieren**.

$$\text{Reduzieren von } \rho\text{:} \qquad \frac{\vec{\eta}\vec{\alpha} \vdash \vec{x}}{\vec{\eta}X \vdash \vec{x}}$$

Hierbei muß $\rho = X \to \vec{\alpha}$ eine G–Regel sein. Diesen Kalkül nennen wir den Kalkül des **Schiebens und Reduzierens für G**. Es gilt nun folgender Satz, der leicht durch Induktion über die Länge der Ableitung zu erbringen ist.

Theorem 2.4.16 *Es sei G eine Grammatik. Genau dann ist $\vec{\alpha} \vdash_G \vec{x}$, wenn es eine Ableitung von $\vec{\alpha} \vdash \varepsilon$ aus $\varepsilon \vdash \vec{x}$ im Kalkül des Schiebens und Reduzierens für G gibt.*

Tabelle 2.1: Eine Ableitung durch Schieben und Reduzieren

ε	$\vdash$ aacbb
a	$\vdash$ acbb
A	$\vdash$ acbb
Aa	$\vdash$ cbb
AA	$\vdash$ cbb
AAc	$\vdash$ bb
AAS	$\vdash$ bb
AASb	$\vdash$ b
AASB	$\vdash$ b
AS	$\vdash$ b
ASb	$\vdash \varepsilon$
ASB	$\vdash \varepsilon$
S	$\vdash \varepsilon$

Diese Strategie kann auf jede Sprache angewendet werden. Wir nehmen folgende Grammatik

$$\begin{array}{lcl} \texttt{S} & \to & \texttt{ASB} \mid \texttt{c} \\ \texttt{A} & \to & \texttt{a} \\ \texttt{B} & \to & \texttt{b} \end{array}$$

Dann ist $S \vdash_G \texttt{aacbb}$. In der Tat bekommen wir die in Tabelle 2.1 gezeigte Ableitung. Natürlich ist dieser Kalkül nicht eindeutig; wir müssen an vielen Stellen raten, ob wir Schieben sollen oder Reduzieren und welche Regel wir reduzieren (auch hier kann es zu Konkurrenzfällen kommen, nämlich wenn die rechte Seite einer Produktion ein Suffix der rechten Seite einer anderen Produktion ist). Wir nennen nun eine k–**Strategie** eine Funktion f, welche zu einem Paar $\vec{\alpha} \vdash \vec{x}$ festlegt, ob man reduzieren soll oder schieben. Dabei soll f nur abhängen von (1) den möglichen Reduktionsregeln, welche auf $\vec{\alpha}$ angewendet werden können, und (2) von den ersten k Zeichen von $\vec{x}$. Wir nehmen an, daß der im Konkurrenzfall nur eine Regel zum Zuge kommen kann. Eine k–Strategie ist eine Abbildung von $R \times \bigcup_{i<k} A^i$ nach $\{s, r\}$. Ist $\vec{\alpha} \vdash \vec{x}$ gegeben, so bestimmt man die nächste Regelanwendung wie folgt. Es sei $\vec{\beta}$ ein Suffix von $\vec{\alpha}$, welches reduzierbar ist. Ist $f(\vec{\beta}, {}^{(k)}\vec{x}) = s$, so wird geschoben; ist $f(\vec{\beta}, {}^{(k)}\vec{x}) = r$, so wird $\vec{\beta}$ reduziert. Dies ist im allgemeinen noch nicht eindeutig; denn es kann ja eine rechte Seite einer Regel ein Präfix einer rechten Seite einer anderen Regel sein. Wir betrachten daher die Eigenschaft

(∗) Ist $\rho_1 = X_1 \to \vec{\beta}_1 \in R$ und $\rho_2 = X_2 \to \vec{\beta}_2 \in R$, $\rho_1 \neq \rho_2$, und hat $\vec{y}$ die Länge $\leq k$, so ist $f(\vec{\beta}_1, \vec{y})$ oder $f(\vec{\beta}_2, \vec{y})$ undefiniert.

Definition 2.4.17 *Eine Grammatik heißt* ***LR***(k)*–**Grammatik**, falls nicht* $\mathtt{S} \Rightarrow^+ \mathtt{S}$ *und falls für ein* $k \in \omega$ *eine* k*–Strategie für das Ableitungsverfahren existiert. Eine Sprache heißt* $LR(k)$*–**Sprache**, falls es eine* $LR(k)$*–Grammatik gibt, welche sie erzeugt.*

Theorem 2.4.18 *Eine Grammatik ist eine* $LR(k)$*–Grammatik, falls gilt: sind* $\vec{\eta}_1\vec{\alpha}_1\vec{x}_1$ *und* $\vec{\eta}_2\vec{\alpha}_2\vec{x}_2$ *rechtsseitig ableitbar, und sei* $p := |\vec{\eta}_1\vec{\alpha}_1| + k$*, und ist*

$$^{(p)}\vec{\eta}_1\vec{\alpha}_1\vec{x}_1 = {}^{(p)}\vec{\eta}_2\vec{\alpha}_2\vec{x}_2$$

so ist $\vec{\eta}_1 = \vec{\eta}_2$*,* $\vec{\alpha}_1 = \vec{\alpha}_2$ *und* $^{(k)}\vec{x}_1 = {}^{(k)}\vec{x}_2$*.*

Dieser Satz ist nicht schwer zu zeigen. Er besagt, daß die Strategieentscheidung tatsächlich nur von dem Präfix der Länge k der zu lesenden Zeichenkette abhängt. Dies ist im Wesentlichen also die Eigenschaft $(*)$. Man muß sich dabei noch klarmachen, daß eine Ableitung im Kalkül des Schiebens und Reduzierens einer rechtsseitigen Ableitung in G entspricht, vorausgesetzt, man reduziert so früh wie möglich. Wir werden im Folgenden zwei Sätze beweisen, welche genau die $LR(k)$–Sprachen charakterisieren. Es stellt sich nämlich heraus, daß jede deterministische Sprache schon eine $LR(1)$–Sprache ist, während die $LR(0)$–Sprachen eine echte Teilklasse bilden. Da eine $LR(k)$–Sprache auch schon eine $LR(k+1)$–Sprache ist, und jede $LR(k)$–Sprache deterministisch ist, bekommt man also nur zwei Klassen von Sprachen: $LR(0)$– und $LR(1)$–Sprachen.

Theorem 2.4.19 *$LR(k)$–Sprachen sind deterministisch.*

Diesen Satz überlassen wir dem Leser. Die Aufgabe ist nun lediglich, aus einer Strategie einen deterministischen Automaten zu extrahieren, der die Sprache erkennt.

Lemma 2.4.20 *Jede $LR(k)$–Sprache ist eine $LR(k+1)$–Sprache.*

Der Beweis dieses Satzes ist einfach. Wir haben daher die folgende Hierarchie:

$$LR(0) \subseteq LR(2) \subseteq LR(2) \subseteq LR(3)\ldots$$

Diese Hierarchie ist jedoch schon ab $k = 1$ stationär.

Lemma 2.4.21 *Es sei $k > 0$. Ist S eine $LR(k+1)$–Sprache, so ist S auch eine $LR(k)$–Sprache.*

Beweis. Zum Beweis konstruieren wir aus einer $LR(k+1)$–Grammatik G eine $LR(k)$–Grammatik $G^{>}$. Der Einfachheit halber nehmen wir an, daß G in Chomsky–Normalform ist. Der allgemeine Fall ist leicht aus dem Spezialfall zu bekommen. Die Idee zu hinter der Konstruktion ist wie folgt. Eine Konstituente von $G^{>}$ entspricht einer um ein Zeichen nach rechts verschobenen Konstituente von G. Wir führen dazu neue Symbole ein, nämlich $[a, X, b]$, $a, b \in A$, und $[a, X, \varepsilon]$, $a \in A$. Das Startsymbol von $G^{>}$ ist das Startsymbol von G. Die Regeln sind wie folgt:

$$\begin{array}{lcll}
\mathtt{S} & \to & \varepsilon & \text{falls } \mathtt{S} \to \varepsilon \in R \\
\mathtt{S} & \to & a\ [a, \mathtt{S}, \varepsilon] & a \in A \\
[a, X, b] & \to & [a, Y, c]\ [c, Z, b] & a, b, c \in A, X \to YZ \in R \\
[a, X, \varepsilon] & \to & [a, Y, c]\ [c, Z, \varepsilon] & a, b \in A, X \to YZ \in R \\
[a, X, b] & \to & b & a, b \in A, X \to a \in R \\
[a, X, \varepsilon] & \to & \varepsilon & a \in A, X \to a \in R
\end{array}$$

Durch Induktion über die Länge der Ableitung zeigt man nun:

$$[a, X, b] \vdash_{G^{>}} \vec{\alpha}\, b \qquad \Leftrightarrow \qquad X \vdash_G a\, \vec{\alpha}$$

$$[a, X, \varepsilon] \vdash_{G^{>}} \vec{\alpha} \qquad \Leftrightarrow \qquad X \vdash_G a\, \vec{\alpha}$$

Daraus läßt sich nun leicht ableiten, daß $G^{>}$ eine $LR(k)$–Grammatik ist. Seien dazu $\vec{\eta}_1\vec{\alpha}_1\vec{x}_1$ und $\vec{\eta}_2\vec{\alpha}_2\vec{x}_2$ rechtsseitig in $G^{>}$ ableitbar, und sei $p := |\vec{\eta}_1\vec{\alpha}_1| + k$, sowie

$${}^{(p)}\vec{\eta}_1\vec{\alpha}_1\vec{x}_1 = {}^{(p)}\vec{\eta}_2\vec{\alpha}_2\vec{x}_2$$

Dann ist $\vec{\eta}_1\vec{\alpha}_1\vec{x}_1 = \vec{\eta}'_1\vec{\alpha}'_1 b\vec{x}_1$ für ein $b \in A$ und gewisse $\vec{\eta}'_1$, $\vec{\alpha}'_1$ mit $a\vec{\eta}_1 = \vec{\eta}'_1 c$ für $a, c \in A$ und $c\vec{\alpha}_1 = \vec{\alpha}'_1 b$. Ferner ist und $\vec{\eta}_2\vec{\alpha}_2\vec{x}_2 = \vec{\eta}'_2\vec{\alpha}'_2 b\vec{x}_2$, $a\vec{\eta}_2 = \vec{\eta}'_2 c$ und $c\vec{\alpha}_2 = \vec{\alpha}'_2$ für gewisse $\vec{\eta}'_2$ und $\vec{\alpha}'_2$. Also haben wir

$${}^{(p)}\vec{\eta}'_1\vec{\alpha}'_1 b\vec{x}_1 = {}^{(p)}\vec{\eta}'_2\vec{\alpha}'_2 b\vec{x}_2$$

und $p = |\vec{\eta}'_1\vec{\alpha}'_1| + k + 1$. Ferner sind die linke und rechte Zeichenkette rechtsseitig ableitbar in G. Daraus folgt jetzt, da G eine $LR(k)$–Grammatik ist, daß $\vec{\eta}'_1 = \vec{\eta}'_2$ und $\vec{\alpha}'_1 = \vec{\alpha}'_2$, sowie ${}^{(k+1)}b\vec{x}_1 = {}^{(k+1)}b\vec{x}_2$. Wir bekommen daraus $\vec{\eta}_1 = \vec{\eta}_2$, $\vec{\alpha}_1 = \vec{\alpha}_2$ und ${}^{(k)}\vec{x}_1 = {}^{(k)}\vec{x}_2$, wie verlangt. $\dashv$

Wir werden nun folgenden wichtigen Satz beweisen.

Theorem 2.4.22 *Jede deterministische Sprache ist eine $LR(1)$–Sprache.*

Der Beweis ist relativ langwierig. Zu allererst werden wir uns ein paar Hilfssätze besorgen, die zeigen, daß strikt deterministische Sprachen genau die durch strikt deterministische Grammatiken erzeugten Sprachen sind, und daß sie eindeutig sind und sogar $LR(0)$. Dies wird dann den Schlüssel für den allgemeinen Satz geben.

Zunächst also zurück zu den strikt deterministischen Sprachen. Wir sind noch den Nachweis schuldig geblieben, daß strikt deterministische Grammatiken nur strikt deterministische Sprachen erzeugen können. Dies ist im Wesentlichen die Folge einer Eigenschaft, die wir die *Linkstransparenz* nennen wollen. Wir sagen, $\vec{\alpha}$ trete in $\vec{\eta}_1\,\vec{\alpha}\,\vec{\eta}_2$ mit **linkem Kontext** $\vec{\eta}_1$ auf.

Definition 2.4.23 *Es sei G eine Grammatik. G heißt **linkstransparent**, falls eine Konstituente in einer Zeichenkette niemals mit gleichem linken Kontext akzidentiell auftreten kann. Dies bedeutet: ist $\vec{x}$ eine Konstituente vom Typ C in $\vec{y}_1\,\vec{x}\,\vec{y}_2$, und ist $\vec{z} := \vec{y}_1\,\vec{x}\,\vec{y}_3$ eine G–Zeichenkette, so ist $\vec{x}$ auch in $\vec{z}$ eine Konstituente vom Typ C.*

Für den folgenden Satz benötigen wir ein paar Definitionen. Es sei $\mathfrak{B}$ ein Baum und $n \in \omega$ eine natürliche Zahl. Dann bezeichnet ${}^{(n)}\mathfrak{B}$ denjenigen Baum, der aus allen Knoten oberhalb der ersten n Blätter besteht. Sei dazu P die Menge der Blätter von $\mathfrak{B}$, etwa $P = \{p_i : i < q\}$ und sei $p_i \sqsubset p_j$ genau dann, wenn $i < j$. Dann setze $N_n := \{p_i : i < n\}$, und $O_n := \uparrow N_n$. ${}^{(n)}\mathfrak{B} := \langle O_n, r, <, \sqsubset\rangle$, wobei $<$ und $\sqsubset$ die auf O_n relativierten Relationen sind. Ist nun ℓ eine Markierung und $\mathfrak{T} = \langle\mathfrak{B}, \ell\rangle$ ein markierter Baum, so sei ${}^{(n)}\mathfrak{T} := \langle {}^{(n)}\mathfrak{B}, \ell \restriction O_n\rangle$. Wiederum bezeichnen wir $\ell \restriction O_n$ schlicht mit ℓ. Wir bemerken, daß die Menge $R_n := {}^{(n)}\mathfrak{B} - {}^{(n-1)}\mathfrak{B}$ durch $<$ linear geordnet ist. Genauer existiert für jedes $y \in R_n$, welches kein Blatt ist, ein z mit $z \prec y$. Betrachten wir nun das größte Element u von R_n. Dann entstehen zwei Fälle. (a) z hat keine rechte Schwester. (b) z hat eine rechte Schwester. Im fall (a) wird beim Übergang von ${}^{(n-1)}\mathfrak{B}$ zu ${}^{(n)}\mathfrak{B}$ die Konstituente der Mutter von u abgeschlossen. Es sei y am **rechten Rand** von $\mathfrak{T}$, falls es kein z gibt mit $y \sqsubset z$. Es besteht $\uparrow R_n$ aus genau denjenigen Elementen, die am rechten Rand von ${}^{(n)}\mathfrak{B}$, und R_n besteht aus denjenigen Elementen, welche in ${}^{(n)}\mathfrak{B}$ am rechten Rand sind nicht aber in ${}^{(n-1)}\mathfrak{B}$. Nun gilt:

Proposition 2.4.24 *Sei G eine strikt deterministische Grammatik. Dann ist G linkstransparent. Ferner gilt: Es seien $\mathfrak{T}_1 = \langle\mathfrak{B}_1, \ell_1\rangle$ und $\mathfrak{T}_2 = \langle\mathfrak{B}_2, \ell_2\rangle$ partielle G–Bäume, so daß folgendes gilt:*

1. *Ist C_i die Marke der Wurzel von $\mathfrak{T}_i$, dann ist $C_1 \equiv C_2$.*
2. *${}^{(n)}k(\mathfrak{T}_1) = {}^{(n)}k(\mathfrak{T}_2)$.*

Dann gibt es einen Isomorphismus $f : {}^{(n+1)}\mathfrak{B}_1 \twoheadrightarrow {}^{(n+1)}\mathfrak{B}_2$ mit $\ell_2(f(x)) = \ell_1(x)$, falls x nicht am rechten Rand von ${}^{(n+1)}\mathfrak{B}_1$ ist und $\ell_2(f(x)) \equiv \ell_1(x)$ sonst.

Beweis. Wir beweisen den Satz durch Induktion über n. Wir nehmen an, er gelte für alle $k < n$. Ist $n = 0$, so gilt diese Annahme ohnehin. Nun zeigen wir die

Behauptung für n. Es existiert also nach Voraussetzung ein Isomorphismus $f_n : {}^{(n)}\mathfrak{B}_1 \to {}^{(n)}\mathfrak{B}_2$ mit den oben angegebenen Eigenschaften. Setze wieder $R_{n+1} := {}^{(n+1)}\mathfrak{B}_1 - {}^{(n)}\mathfrak{B}_1$. Wir werden nun als erstes zeigen, daß $\ell_2(f_n(x)) = \ell_1(x)$ für alle $x \notin \uparrow R_{n+1}$. Daraus folgt unmittelbar, daß $\ell_2(f_n(x)) \equiv \ell_1(x)$ für alle $x \in \uparrow R_{n+1} - R_{n+1}$, da G strikt deterministisch ist. Die Behauptung zeigen wir durch Induktion über die Höhe von x. Ist $h(x) = 0$, so ist x Blatt, und die Behauptung gilt aufgrund der Annahme, daß $\mathfrak{T}_1$ und $\mathfrak{T}_2$ dieselben assoziierten Zeichenketten haben. Ist $h(x) > 0$, so sind alle Töchter von x nicht in $\uparrow R_{n+1}$. Nach Induktionsvoraussetzung ist demnach $\ell_2(f_n(y)) = \ell_1(y)$ für alle $y \prec x$. Da G strikt deterministisch ist, ist die Marke von x eindeutig bestimmt, denn $\ell_2(f_n(x)) \equiv \ell_1(x)$, nach Induktionsannahme. Daher gilt jetzt: $\ell_2(f_n(x)) = \ell_1(x)$, Dies zeigt die erste Behauptung. Nun werden wir noch f_n zu einem Isomorphismus f_{n+1} von ${}^{(n+1)}\mathfrak{B}_1$ auf ${}^{(n+1)}\mathfrak{B}_2$ fortsetzen und gleichzeitig beweisen, daß $\ell_2(f_n(x)) \equiv \ell_1(x)$ für alle $x \in \uparrow R_{n+1}$. Dies gilt schon nach Induktionsvoraussetzung für alle $x \notin R_{n+1}$. Also müssen wir dies nur für $x \in R_{n+1}$ zeigen. Dies machen wir wie folgt. Es sei u_0 der größte Knoten in R_{n+1}. Gewiß ist u_0 nicht die Wurzel. Es sei daher v die Mutter von u_0. f_n ist auf v definiert, und wir haben $\ell_2(f_n(v)) \equiv \ell_1(v)$. Nach Voraussetzung gilt für alle $x \sqsubset u$: $\ell_2(f_n(x)) = \ell_1(x)$. Also gilt jetzt erst einmal, daß es eine Tochter x_0 von $f_n(v)$ gibt, welche nicht im Bild von f_n ist. Wir wählen x_0' minimal mit dieser Eigenschaft. Dann setzen wir $f_{n+1}(u_0) := x_0$. Es gilt jetzt $\ell_2(f_{n+1}(u_0)) \equiv \ell_1(u_0)$. Wir fahren nun mit u_0 anstelle von v fort. Auf diese Weise erhalten wir eine Abbildung f_{n+1} von ${}^{(n)}\mathfrak{B}_1 \cup R_{n+1} = {}^{(n+1)}\mathfrak{B}_1$ in ${}^{(n+1)}\mathfrak{B}_2$ mit $\ell_2(f_{n+1}(x)) \equiv \ell_1(x)$, falls $x \in R_{n+1}$, und $\ell_2(f_{n+1}(x)) = \ell_1(x)$ sonst. Daß f_{n+1} auch surjektiv ist, sieht man so. Angenommen, u_k sei das Blatt von $\mathfrak{B}_1$ in R_{n+1}. Dann ist jetzt $x_k = f_{n+1}(u_k)$ kein Blatt in $\mathfrak{B}_2$, und dann existiert ein x_{k+1}, welches in ${}^{(n+1)}\mathfrak{B}_2 - {}^{(n)}\mathfrak{B}_2$ ist. Es gilt $\ell_2(f_{n+1}(x_k)) \equiv \ell_1(u_k)$. Sei x_p das Blatt in L. Aufgrund von Lemma 2.3.17 ist dann $\ell_2(x_p) \not\equiv \ell_2(x_k)$ und deshalb auch $\ell_2(x_p) \not\equiv \ell_1(u_k)$. Aber nach Voraussetzung is ja x_p das n+1te Blatt in $\mathfrak{B}_2$, und ebenso ist u_k das $n+1$te Blatt in $\mathfrak{B}_1$, woraus dann $\ell_1(u_k) = \ell_2(x_p)$ folgt, im Widerspruch zu dem eben Gezeigten. $\dashv$

Theorem 2.4.25 *Es sei G eine strikt deterministische Grammatik. Dann ist $L(G)$ eindeutig. Ferner ist G eine $LR(0)$–Grammatik und strikt deterministisch.*

Beweis. Die Strategie des Schiebens und Reduzierens können wir auf folgende Weise anwenden: immer wenn, von links kommend, eine rechte Seite einer Regel $X \to \vec{\mu}$ identifiziert ist, so ist schon eine Konstituente vom Typ X gefunden, und wir können reduzieren. Dies zeit, daß wir eine 0–Strategie haben. Daher ist die Grammatik eine $LR(0)$–Grammatik. $L(G)$ ist sicher eindeutig. Ferner ist $L(G)$ deterministisch, nach Satz 2.4.19. Zu guter Letzt müssen wir noch noch zeigen, daß $L(G)$ präfixfrei ist, denn dann folgt mit Satz 2.3.12 schon, daß $L(G)$ strikt deterministisch ist. Nun sei $\vec{x}\,\vec{y} \in L(G)$. Ist dann auch $\vec{x} \in L(G)$, so gilt nach Proposition 2.4.24 schon $\vec{y} = \varepsilon$. $\dashv$

Es erscheint auf den ersten Blick, daß das Lemma 2.4.21 auch für $k = 0$ gilt. Die Konstruktion läßt sich ja ohne weiteres auf diesen Fall ausdehnen. In der Tat bekommen wir dann so etwas wie eine $LR(0)$–Grammatik; allerdings überlege man sich, daß eine Strategie für $G^{>}$ nun nicht mehr von dem nächsten Symbol abhängt, wohl aber von der Tatsache, ob das zu noch lesende Wort leer ist oder nicht. Die Strategie ist also nicht ganz unabhängig von dem rechten Kontext, obwohl die Abhängigkeit schon stark reduziert ist. Daß $LR(0)$–Sprachen in der Tat spezieller sind als $LR(1)$–Sprachen, das ist eine Folge aus dem nächsten Satz.

Theorem 2.4.26 (Geller & Harrison) *Sei S eine deterministische kontextfreie Sprache. Dann sind äquivalent:*

1. *S ist $LR(0)$–Sprache.*
2. *Falls $\vec{x} \in S$, $\vec{x}\,\vec{v} \in S$ und $\vec{y} \in S$, so ist auch $\vec{y}\,\vec{v} \in S$.*
3. *Es gibt strikt deterministische Sprachen U und V, sodaß $S = U \cdot V^*$.*

Beweis. Es sei S eine $LR(0)$–Sprache. Dann gibt es eine $LR(0)$–Grammatik G für S. Also gilt: ist $X \to \vec{\alpha}$ eine Regel, und ist $\vec{\eta}\,\vec{\alpha}\,\vec{y}$ G–ableitbar, so ist auch $\vec{\eta}\,X\,\vec{y}$ in G ableitbar. Durch Induktion läßt sich dieser Sachverhalt auch für alle Paare X, $\vec{\alpha}$ zeigen, für die $X \vdash_G \vec{\alpha}$. Nun sei $\vec{x} \in S$ und $\vec{x}\,\vec{v} \in S$. Dann gilt $\mathtt{S} \vdash_G \vec{x}$, und so haben wir nach dem Vorangegangenen $\vdash_G \mathtt{S}\,\vec{v}$. Nun gilt auch $\mathtt{S} \vdash_G \vec{y}$, und so $\vdash_G \vec{y}\,\vec{v}$. Also gilt (2.). Nun gelte die Eigenschaft (2.) für S. Es sei U die Menge aller $\vec{x} \in S$ derart, daß $\vec{y} \notin S$ für jedes echte Präfix $\vec{y}$ von $\vec{x}$. Es sei V die Menge aller $\vec{v}$ derart, daß $\vec{x}\,\vec{v} \in S$ für ein $\vec{x} \in U$, aber $\vec{x}\,\vec{w} \notin S$ für jedes $\vec{x} \in U$ und jedes echte Präfix $\vec{w}$ von $\vec{v}$. V ist nun die Menge aller $\vec{y} \in V^* - \{\varepsilon\}$, für die kein echtes Präfix in $V^* - \{\varepsilon\}$ ist. Es gilt $U \cdot V^* = S$. Dazu sei erst einmal gezeigt, daß $S \subseteq U \cdot V^*$. Sei $\vec{u} \in S$. Wir unterscheiden zwei Fälle: (a) Kein echtes Präfix von $\vec{u}$ ist in S. Dann ist $\vec{u} \in U$, nach Definition von U. (b) Es gibt ein echtes Präfix $\vec{x}$ von $\vec{u}$, welches in S ist. Wir wählen $\vec{x}$ minimal. Dann ist $\vec{x} \in U$. Sei $\vec{u} = \vec{x}\,\vec{v}$. Nun entstehen wieder zwei Fälle: (A) Für kein echtes Präfix $\vec{w}_0$ von $\vec{v}$ ist $\vec{x}\,\vec{w}_0 \in S$. Dann ist $\vec{v} \in V$, und wir sind fertig. (B) Es gibt ein echtes Präfix $\vec{w}_0$ von $\vec{v}$ mit $\vec{x}\,\vec{w}_0 \in S$. Sei $\vec{v} = \vec{w}_0\,\vec{v}_1$. Dann gilt aufgrund der Eigenschaft (2) $\vec{x}\,\vec{v}_1 \in S$. (Setze in (2.) für $\vec{x}$ jetzt $\vec{x}\,\vec{w}_0$, für $\vec{y}$ setze $\vec{x}$ und für $\vec{w}$ setze $\vec{v}_1$.) $\vec{x}\,\vec{v}_1$ hat kleinere Länge als $\vec{x}\,\vec{v}$. Fahre nun mit $\vec{x}\,\vec{v}_1$ in der gleichen Weise fort. Am Ende erhält man eine Zerlegung von $\vec{v} = \vec{w}_0\,\vec{w}_1 \ldots \vec{w}_{n-1}$ mit $\vec{w}_i \in V$ für alle $i < n$. Also ist $S \subseteq U \cdot V^*$. Wir zeigen nun $U \cdot V^* \subseteq S$. Sei nämlich $\vec{u} = \vec{x} \cdot \prod_{i<n} \vec{w}_i$. Ist $n = 0$, so ist $\vec{u} = \vec{x}$ und nach Definition von U ist $\vec{u} \in S$. Nun sei $n > 0$. Mit der Eigenschaft (2) läßt sich zeigen, daß $\vec{x} \cdot \prod_{i<n-1} \vec{w}_i \in S$ ist. Dies zeigt, daß $\vec{u} \in S$. Schließlich müssen wir zeigen, daß U und V deterministisch sind. Dies folgt für U bereits aus Satz 2.3.13. Nun seien $\vec{x}, \vec{y} \in U$. Dann gilt wegen (2.) $P := \{\vec{v} : \vec{x}\,\vec{v} \in S\} = \{\vec{v} : \vec{y}\,\vec{v} \in S\}$. Der Leser möge sich überzeugen, daß P

deterministisch ist. Nun sei V die Menge aller $\vec{v}$, für die kein Präfix in $P - \{\varepsilon\}$ ist. Dann ist $P = V^*$ und wegen Satz 2.3.13 ist V strikt deterministisch. Dies zeigt die Behauptung (3.). Nun gelte schließlich (3.). Wir haben zu zeigen, daß S eine $LR(0)$–Sprache ist. Dazu sei $G_1 = \langle \mathtt{S}, N_1, A, R_1 \rangle$ eine strikt deterministische Grammatik, welche U erzeugt und $G_2 = \langle \mathtt{S}_2, N_2, A, R_2 \rangle$ eine strikt deterministische Grammatik, welche V erzeugt. Dann sei $G_3 := \langle \mathtt{S}_3, N_1 \cup N_2 \cup \{\mathtt{S}_3, \mathtt{S}_4\}, A, R_3 \rangle$ wie folgt definiert.

$$\begin{array}{rll} R_3 := & & \{\mathtt{S}_3 \to \mathtt{S}_1, \mathtt{S}_3 \to \mathtt{S}_1\,\mathtt{S}_4, \mathtt{S}_4 \to \mathtt{S}_2, \mathtt{S}_4 \to \mathtt{S}_2\,\mathtt{S}_4\} \\ & \cup & R_1 \cup R_2 \end{array}$$

Es ist nicht schwer zu sehen, daß G_3 eine $LR(0)$–Grammatik ist und daß $L(G_3) = S$. $\dashv$

Die Zerlegung in (3.) ist eindeutig, wenn man ausschließt, daß $V = \varnothing$ und daß $U = \{\varepsilon\}$ nur dann, wenn $V = \{\varepsilon\}$. Damit versorgt man die Fälle $S = \varnothing$ und $S = U$. Im Übrigen kann $U = V$ durchaus auftreten,. Dann ist $S = U^+$. Ein solcher Fall sind die Semi–Dyck–Sprachen.

Nun also zum Beweis von Satz 2.4.22. Sei S deterministisch. Dann setze $T := S \cdot \{\$\}$, wo \$ ein neues Symbol ist. T ist sicherlich deterministisch und auch präfixfrei, also auch strikt deterministisch. Es folgt, daß T eine $LR(0)$–Sprache ist. Es gibt also eine strikt deterministische Grammatik G, welche T erzeugt. Aus dem nachstehenden Lemma folgt dann, daß S eine $LR(1)$–Sprache ist.

Lemma 2.4.27 *Es sei G eine $LR(0)$–Grammatik der Form $G = \langle \mathtt{S}, N \cup \{\$\}, A, R \rangle$ mit $R \subseteq N \times ((N \cup A)^* \cup (N \cup A)^* \cdot \$)$ und $L(G) \subseteq A^*\$$, und es gebe keine Ableitung $\mathtt{S} \Rightarrow_R \mathtt{S}\$$ in G. Es sei H definiert durch $H := \langle \mathtt{S}, N, A, R_1 \cup R_2 \rangle$, wobei*

$$\begin{array}{rcl} R_1 & := & \{A \to \vec{\alpha} : A \to \vec{\alpha} \in R, \vec{\alpha} \in (N \cup A)^*\} \\ R_2 & := & \{A \to \vec{\alpha} : A \to \vec{\alpha}\,\$ \in R\} \end{array}$$

Dann ist H eine $LR(1)$–Grammatik und $L(H)\,\$ = L(G)$.

Zum Beweis überlege man sich das Folgende. Zunächst gilt nicht $\mathtt{S} \Rightarrow_L^n \mathtt{S}$ in H. Ferner: ist $\mathtt{S} \Rightarrow_L^+ \vec{\alpha}$ in H, so existiert ein D mit $\mathtt{S} \Rightarrow_L^+ \vec{\alpha}\,D$ in G, und ist $\mathtt{S} \Rightarrow_L^+ \vec{\beta}$ in G, so ist $\vec{\beta} = \vec{\alpha}\,D$ und $\mathtt{S} \Rightarrow_L^+ \vec{\alpha}$ in H. Daraus läß sich unmittelbar ableiten, daß H eine $LR(1)$–Grammatik ist.

Kommen wir zum Schluß noch auf den Kalkül des Schiebens und Reduzierens zurück. Wir verallgemeinern diese Strategie nun wie folgt. Zu jedem Symbol α unserer Grammatik nehmen wir noch ein Symbol $\underline{\alpha}$ hinzu. Dieses Symbol sei ein formales Inverses zu α; es signalisiert, daß an seiner Stelle ein α gesucht wird aber nicht gefunden wurde. Dies bedeutet, daß wir den folgenden Übergang erlauben

$$\frac{\vec{\eta}\underline{\alpha}\alpha \vdash \vec{x}}{\vec{\eta} \quad \vdash \vec{x}}$$

Wir nennen diese Regel **Kürzen**. Wir schreiben bei Zeichenketten $\vec{\alpha}$ auch $\underline{\vec{\alpha}}$. Dabei ist zu beachten, daß sich die Reihenfolge umkehrt. So ist $\underline{\mathtt{AB}} = \underline{\mathtt{B}} \cdot \underline{\mathtt{A}}$. Diese neuen Zeichen erlauben es, auf der linken Seite Reduktionen vorzunehmen, auch wenn nur ein Teil der rechten Seite der Produktion identifiziert worden ist. Die allgemeinste Regel ist die folgende:

$$\frac{\vec{\eta}\underline{X}\vec{\alpha} \vdash \vec{x}}{\vec{\eta}\underline{\vec{\delta}} \quad \vdash \vec{x}}$$

Diese Regel heiße die **LC**–Regel. Hierbei ist $X \to \vec{\alpha}\vec{\delta}$ eine G–Regel. Dies bedeutet anschaulich, daß wenn bereits $\vec{\alpha}$ gefunden wurde, die Zeichenkette ein X ist, sofern noch $\vec{\delta}$ folgt. Da $\vec{\beta}$ aber noch nicht vorhanden ist, muß man $\underline{\vec{\delta}}$ hinschreiben. Der **LC–Kalkül** besteht aus den Regeln Schieben, Kürzen und LC. Es gilt nun folgender Sachverhalt.

Theorem 2.4.28 *Es sei G eine Grammatik. Genau dann ist $\vec{\alpha} \vdash_G \vec{x}$, wenn es eine Ableitung von $\varepsilon \vdash \varepsilon$ aus $\underline{\vec{\alpha}} \vdash \vec{x}$ im **LC**–Kalkül gibt.*

Ein Spezialfall ist $\vec{\alpha} = \varepsilon$. Hier ist überhaupt kein Stück der rechten Seite erkannt worden, und man rät gewissermaßen eine Regel. Falls man anstelle der üblichen Reduktionsregeln nur diese Regeln nimmt, bekommt man die **Top–Down**–Strategie. Hier darf man also Schieben, Kürzen, und eine Regel raten. Eine Grammatik heißt $LL(k)$–Grammatik, falls es ein deterministisches Ableitungsverfahren mit Top–Down–Strategie gibt, in welchem der jeweils nächste Schritt von den k ersten Zeichen von $\vec{x}$ abhängt. Der Fall $k = 0$ ist hier allerdings von wenig Nutzen (siehe die Übungen).

Dieses Verfahren ist allerdings zu flexibel, um nützlich sein zu können. Man kann jedoch folgenden Weg gehen. Die rechte Seite einer Produktion wird hier in zwei Teile aufgeteilt, welche durch einen Punkt voneinander getrennt werden.

$$\begin{array}{lcl} \mathtt{S} & \to & \mathtt{A.SB} \mid \mathtt{c.} \\ \mathtt{A} & \to & \mathtt{a.} \\ \mathtt{B} & \to & \mathtt{b.} \end{array}$$

Dieser Punkt legt fest, welches Anfangsstück gelesen werden muß, damit die LC–Regel angewendet werden darf; und dann muß sie auch angewendet werden. Eine Strategie dieser Form heißt **Verallgemeinerte Left–Corner–Strategie**. Ist der Punkt stets ganz rechts, bekommen wir die Bottom–Up–Strategie, ist der Punkt stets ganz links, bekommen wir die Top–Down–Strategie.

Übung 49. Zeigen Sie, daß eine Grammatik G mit $L(G) \neq \{\varepsilon\}$ nur dann transparent ist, wenn sie keine Regeln $A \to \varepsilon$ besitzt.

Übung 50. Beweisen Sie Satz 2.4.19.

Tabelle 2.2: Verallgemeinerte LC–Strategie

$\underline{\texttt{S}}$	$\vdash \texttt{aacbb}$
$\underline{\texttt{S}}\,\texttt{a}$	$\vdash \texttt{acbb}$
$\underline{\texttt{S}}\,\texttt{A}$	$\vdash \texttt{acbb}$
$\underline{\texttt{B}}\,\underline{\texttt{S}}$	$\vdash \texttt{acbb}$
$\underline{\texttt{B}}\,\underline{\texttt{S}}\texttt{a}$	$\vdash \texttt{cbb}$
$\underline{\texttt{B}}\,\underline{\texttt{S}}\,\texttt{A}$	$\vdash \texttt{cbb}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}\,\underline{\texttt{S}}$	$\vdash \texttt{cbb}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}\,\underline{\texttt{S}}\texttt{c}$	$\vdash \texttt{bb}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}\,\underline{\texttt{S}}\texttt{S}$	$\vdash \texttt{bb}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}$	$\vdash \texttt{bb}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}\,\texttt{b}$	$\vdash \texttt{b}$
$\underline{\texttt{B}}\,\underline{\texttt{B}}\,\texttt{B}$	$\vdash \texttt{b}$
$\underline{\texttt{B}}$	$\vdash \texttt{b}$
$\underline{\texttt{B}}\,\texttt{b}$	$\vdash \varepsilon$
$\underline{\texttt{B}}\,\texttt{B}$	$\vdash \varepsilon$
ε	$\vdash \varepsilon$

Übung 51. Beweisen Sie Lemma 2.4.3. Zeigen Sie ferner: Ist $\vec{x}$ ein Term, so ist die folgende Menge P konvex.

$$P := \{\gamma(\vec{y}) : \vec{y} \text{ Präfix von } \vec{x}\}$$

Übung 52. Zeigen Sie: Ist S deterministisch, so ist auch $S/\{\vec{x}\}$ sowie $\{\vec{x}\}\backslash S$ deterministisch.

Übung 53. Man zeige, eine Grammatik ist eine $LL(0)$–Grammatik, falls sie genau einen Baum erzeugt.

Übung 54. Geben Sie ein Beispiel einer NTS–Sprache, welche keine $LR(0)$–Sprache ist.

2.5 Der Satz von Parikh

In diesem Abschnitt wollen wir den sogenannten Satz von Parikh ([37]) zeigen. Der Beweis wird hier geführt, indem wir zunächst zeigen, daß die Baummengen, welche von kontextfreien Grammatiken erzeugt werden, auch von unregulierten Baumadjunktionsgrammatiken erzeugt werden können. Bevor wir jedoch auch diesen Beweis antreten, wollen wir den Satz von Parikh inhaltlich vorstellen.

Der Leser möge sich an unsere Darstellung von Monoiden und Zeichenketten erinnern. Wir wollen wieder von einem endlichen Alphabet A ausgehen. Nun definieren wir eine zweistellige Operation $+$, welche folgenden Gesetzen unterliegt.

$$\begin{array}{lcl} x+0 & = & x \\ x+(y+z) & = & (x+y)+z \\ x+y & = & y+x \end{array}$$

Wir definieren die Notation nx wie folgt. Es ist $0 \cdot x := 0$ und $(k+1) \cdot x := k \cdot x + x$. Analog zu den Zeichenketten betrachten wir Terme, welche aus Elementen aus A aufgebaut sind mit Hilfe des Symbols 0 und dem Symbol $+$. Dabei gelten zwei Terme als gleich, wenn sie unter Benutzung der obigen Gleichungen ineinander übergeführt werden können. Wir bezeichnen mit $M(A)$ die Menge aller solcher Terme unter dieser Identifikation.

Definition 2.5.1 *Eine **kommutative Halbgruppe mit Eins** ist ein Tripel $\langle H, 0, + \rangle$, welches ein Monoid ist und in dem für alle $x, y \in H$ gilt $x + y = y + x$.*

Nach Konstruktion ist $\mathfrak{M}(A) := \langle M(A), 0, + \rangle$ eine kommutative Halbgruppe mit Eins. Mehr noch, $\mathfrak{M}(A)$ ist die von A frei erzeugte kommutative Halbgruppe. Dies nachzuweisen, ist dem Leser überlassen. Wir betrachten nun die Menge ω^n aller n–langen Folgen von natürlichen Zahlen, versehen mit der Operation $+$ definiert durch

$$\langle x_i : i < n \rangle + \langle y_i : i < n \rangle := \langle x_i + y_i : i < n \rangle \ .$$

Diese bildet zusammen mit dem Element $\vec{0}$, bestehend aus lauter Nullen, eine kommutative Halbgruppe, die wir mit Ω^n bezeichnen. Für den folgenden Satz benötigen wir noch das sogenannte **Kroneckersymbol**. Dies ist wie folgt definiert.

$$\delta^i_j := \begin{cases} 1 & \text{wenn } i = j, \\ 0 & \text{sonst.} \end{cases}$$

Theorem 2.5.2 *Es sei $A = \{\mathtt{a}_i : i < n\}$. Sei h diejenige Abbildung, welcher dem Element $\mathtt{a}_i$ die Folge $\vec{e}_i = \langle \delta^i_j : j < n \rangle$ zuordnet. Dann ist der Homomorphismus, welcher h fortsetzt, schon ein Isomorphismus von $\mathfrak{M}(A)$ auf Ω^n.*

Beweis. Es sei $\approx$ die kleinste Kongruenzrelation auf $T(A)$, welche folgende Gleichungen erfüllt.

$$\begin{array}{ll} 0+x & \theta\ x \\ x+(y+z) & \theta\ (x+y)+z \\ x+y & \theta\ y+x \end{array}$$

Wir zeigen, daß zu jedem $x \in T(A)$ ein $y\ \theta\ x$ existiert von der Form

$$\sum_{i<n} k_i \cdot \mathsf{a}_i \ .$$

Dies geschieht induktiv über den Aufbau von x. Hat y diese Form, so ist $h(x) = h(y) = \langle k_i : i < n \rangle$. Diese Abbildung ist sicher surjektiv. Sie ist ein auch Homomorphismus. Denn sei $h(x) = \langle k_i : i < n \rangle$ und $h(y) = \langle \ell_i : i < n \rangle$, so gilt

$$x + y\ \theta\ \sum_{i<n} k_i \cdot \mathsf{a}_i + \sum_{i<n} \ell_i \cdot \mathsf{a}_i\ \theta\ \sum_{i<n} (k_i + \ell_i) \cdot \mathsf{a}_i \ .$$

Also $h(x + y) = h(x) + h(y)$, wie zu beweisen war. $\dashv$

Der Satz bestätigt uns also, daß freie kommutative Halbgruppen im Wesentlichen als Halbgruppen von Zahlenvektoren aufgefaßt werden können. Ein allgemeines Element von $M(A)$ kann also notiert werden als

$$\sum_{i<n} k_i \cdot \mathsf{a}_i \ ,$$

wobei $k_i \in \omega$.

Nun definieren wir die Abbildung $\mu : A^* \to M(A)$ durch

$$\begin{array}{lcl} \mu(1) & := & 0, \\ \mu(\mathsf{a}_i) & := & \mathsf{a}_i, \\ \mu(\vec{x} \cdot \vec{y}) & := & \mu(\vec{x}) + \mu(\vec{y}) \ . \end{array}$$

Diese Abbildung ist ein Homomorphismus von Monoiden und sogar surjektiv. Er ist aber nicht injektiv, außer in dem Fall, wo A aus nur einem Element besteht. Die Abbildung μ heißt **Parikh–Abbildung**. Es gilt dann also

$$\mu\left(\prod_{i<k} \vec{x}_i\right) = \sum_{i<k} \mu(\vec{x}_i) \ .$$

Definition 2.5.3 *Zwei Sprachen $S, T \subseteq A^*$ heißen **buchstabenäquivalent**, falls $\mu[S] = \mu[T]$.*

Definition 2.5.4 *Eine Menge $U \subseteq M(A)$ heißt **linear**, falls*

$$U = \{u + \sum_{i<\alpha} k_i \cdot v_i : k_i \in \omega\}$$

für ein gewisses $\alpha \in \omega$ und gewisse $u, v_i \in M(A)$. Die v_i heißen auch ***zyklische Vektoren*** *von U. Das kleinste α, für das U eine solche Darstellung besitzt, heißt auch die* ***Dimension*** *von U. U heißt* ***semilinear****, falls U endliche Vereinigung von linearen Mengen ist. Eine Sprache $S \subseteq A^*$ heißt* ***semilinear*** *oder* ***Parikh–Sprache****, falls $\mu[S]$ semilinear ist.*

Wir können lineare Mengen relativ kompakt wie folgt aufschreiben. Sind U und V Teilmengen von $M(A)$, so bezeichne $U + V := \{x + y : x \in U, y \in V\}$. Ferner sei $x + U := \{x + y : y \in U\}$. Einzelne Elemente werden also den Einermengen gleichgestellt. Zudem schreiben wir $nU := \{nx : n \in \omega\}$. Schließlich bezeichnet ωU die Vereinigung aller nU, $n \in \omega$. Damit schreiben wir die Menge U aus der Definition als

$$U = u + \omega v_0 + \omega v_1 + \ldots + \omega v_{\alpha-1} \; .$$

Die wiederum kürzen wir wie folgt ab.

$$U = u + \sum_{i<\alpha} \omega v_i$$

Wir ziehen eine nützliche Folgerung aus den Definitionen.

Theorem 2.5.5 *Es sei $v : A \to B^*$ eine Abbildung und sei $S \subseteq A^*$ semilinear. Dann ist $\overline{v}[S]$ auch semilinear.*

Beweis. v induziert eine Abbildung $\kappa_v : \mathfrak{M}(A) \to \mathfrak{M}(B)$. Das κ_v–Bild einer semilinearen Menge ist semilinear. Denn es ist für eine Zeichenkette $\vec{x} \in A^*$ $\mu(\overline{v}(\vec{x})) = \kappa_v(\mu(\vec{x}))$, wie man leicht durch Induktion über die Länge von $\vec{x}$ bestätigt. Sei M linear, etwa $M = \{u + \sum_{i<k} k_i \cdot v_i : k_i \in \omega\}$. Dann ist

$$\kappa_v[M] = \kappa_v(u) + \sum_{i<k} \omega \kappa_v(v_i)$$

Daraus folgt die Behauptung. Es gilt also $\mu[\overline{v}[S]] = \kappa_v[\mu[S]]$. Die rechte Seite ist semilinear, wie wir gesehen haben. $\dashv$

Daraus folgt zum Beispiel, daß wenn wir in einer semilinearen Sprache gewisse Buchstaben tilgen, das Resultat wieder semilinear ist. Dies führt uns also zunächst auf die semilinearen Mengen mit nur einem Buchstaben, das heißt, auf die Frage, welche Teilmengen von ω semilinear sind. Eine Menge $M \subseteq \omega$ ist linear, wenn es ein n_0 und ein c gibt derart, daß für alle $k > n_0$ gilt: $k \in M$ genau dann wenn $k + c \in M$ genau dann wenn $k - c \in M$. Das bedeutet: bis auf endlich viele Ausnahmen ist M die Menge aller Zahlen der Form $a_0 + mc$ für gewisse a_0 und c. Dies ist eine Konsequenz aus dem Chinesischen Restsatz (siehe zum Beispiel [1]).

Denn sei $M = \{p_0 + \sum_{i<k} \lambda_i q_i : \lambda_i \in \omega\}$. Setze $c := ggT\{q_i : i < k\}$. c läßt sich nach dem Chinesischen Restsatz darstellen als

$$c = \sum_{i<k} \mu_i q_i,$$

mit ganzen Zahlen μ_i. Da die μ_i nicht unbedingt positiv sind, behelfen wir uns wie folgt. Sei $\mu_0 < 0$. Sei d das kleinste gemeinsame Vielfache der q_i und sei $d = q_0 r_0$ für ein gewisses r_0. Dann ist

$$c + d = (\mu_0 + r_0) + \sum_{0<i<k} \mu_i q_i \ .$$

Dies wiederholen wir so oft bis der Koeffizient von q_0 positiv ist und dies tun wir für jedes $i < k$. Damit erhalten wir eine Darstellung

$$c + id = \sum_{i<k} \nu_i q_i \ .$$

Hier sind alle $\nu_i \geq 0$. Ebenso können wir $2c + i'd$ als positive Summe darstellen für gewisses i'. Wählen wir i genügend groß, besitzen alle Zahlen der Form $\gamma c + id$ zwischen id und $(i+1)d$ eine positive Darstellung. Da d eine positive Darstellung besitzt, ist jede Zahl der Form $p_0 + uc$ in M, sofern $uc \geq id$ ist. Da nun eine Zahl r, die nicht durch c teilbar ist, sich nicht als ganzzahlige Summe der q_i darstellen läßt, ist $r \notin M$, falls $r - p_0$ nicht durch c teilbar ist. Wir nennen nun eine Menge **arithmetisch**, falls sie die Form $p + \omega c = \{p + kc : k \in \omega\}$ hat. Diese Menge ist $= \{p\}$, falls $c = 0$.

Theorem 2.5.6 *Eine Teilmenge von ω ist semilinear genau dann, wenn sie Vereinigung von endlich vielen arithmetischen Mengen ist.*

Wir können dies noch verschärfen. Wir nennen U **fast periodisch**, wenn es ein n_0 gibt und ein p derart, daß für alle $n \in U$ mit $n \geq n_0$ gilt $n + p \in U$. Man überlegt sich leicht, daß eine Vereinigung von fast periodischen Mengen wieder fast periodisch ist. Also gilt

Theorem 2.5.7 *Eine Teilmenge von ω is semilinear genau dann, wenn sie fast periodisch ist.*

Hieraus folgt, daß das Komplement eine semilinearen Menge $\subseteq \omega$ wieder semilinear ist. Wir wollen hier ohne Beweis feststellen, daß dies im allgemeinen auch gilt:

Theorem 2.5.8 *Das Komplement einer semilinearen Teilmenge von Ω^n ist wieder semilinear. Also ist auch der Schnitt von endlich vielen semilinearen Teilmengen von Ω^n wieder semilinear.*

Theorem 2.5.9 *Eine Sprache ist eine Parikh–Sprache genau dann, wenn sie buchstabenäquivalent ist zu einer regulären Sprache.*

Beweis. ($\Rightarrow$). Sei S semilinear, also $S = \bigcup T_i$ für gewisse Sprachen T_i, für die $\mu[T_i]$ linear ist. Es genügt, die Behauptung für die T_i zu zeigen. Sei dazu $T := T_i$ und

$$\mu[T] = u + \sum_{j<\alpha} \omega \cdot v_j$$

Dann gibt es zu u ein $\vec{x}$ mit $\mu(\vec{x}) = u$ und zu v_j ein $\vec{y}_j$ mit $\mu(\vec{y}_j) = v_j$, $j < \alpha$. Sowohl $\vec{x}$ als auch die $\vec{y}_j$ sind Zeichenketten, also reguläre Terme. Jetzt setze

$$R := \vec{x} \cdot (\vec{y}_0)^* \cdot (\vec{y}_1)^* \cdot \ldots \cdot (\vec{y}_{\alpha-1})^*$$

Dann ist $L(R)$ regulär und $\mu[L(R)] = \mu[T]$. ($\Leftarrow$). Induktiv über den Aufbau des regulären Terms R beweisen wir, daß $\mu[L(R)]$ semilinear ist. Dies ist klar für $R = \mathsf{a}_i$ oder $R = \varepsilon$. Ebenso ist dies klar für $R = S_1 \cup S_2$. Nun sei $R = S_1 \cdot S_2$. Falls S_1, S_2 linear sind, etwa

$$\mu[L(S_1)] = u + \sum_{i<\alpha} \omega v_i, \qquad \mu[L(S_2)] = u' + \sum_{i<\alpha'} \omega v'_i$$

dann ist

$$\mu[L(R)] = (u + u') + \sum_{i<\alpha} \omega v_i + \sum_{j<\alpha'} \omega v'_j$$

Sind S_1, S_2 lediglich semilinear, so beachte man, daß wegen $(S \cup T) \cdot U = S \cdot U \cup T \cdot U$ und $U \cdot (S \cup T) = U \cdot S \cup U \cdot T$, R eine Vereinigung von Produkten von linearen Mengen ist, also semilinear. Nun sei zum Schluß $R = S^*$. Falls $S = T \cup U$, so ist $R = (T^* \cdot U^*)^*$, so daß wir uns wiederum auf den Fall beschränken können, daß S linear ist. Nun sei $R = S^*$ und S semilinear, etwa

$$\mu[L(S)] = u + \sum_{i<\alpha} \omega v_i$$

Dann ist

$$\mu[L(R)] = \omega u + \sum_{i<\alpha} \omega v_i$$

Also ist auch R linear. Dies beendet den Beweis. $\dashv$

Nun werden wir wie angekündigt die Einbettung von kontextfreien Baummengen in die Baummengen von unregulierten Baumadjunktionsgrammatiken beweisen. Sei $G = \langle S, N, A, R\rangle$ eine kontextfreie Grammatik. Wir wollen eine Baumadjunktionsgrammatik $\mathsf{Ad}_G = \langle \mathsf{C}_G, \mathsf{A}_G\rangle$ definieren dergestalt, daß $L_B(G) = L_B(\mathsf{Ad}_G)$. Wir definieren C_G als die Menge derjenigen Bäume $\mathfrak{B}$, die man $L_B(G)$ ableiten kann,

welche Adjunktionsbäume sind, bei denen auf keinem Pfad ein Nichtterminalsymbol doppelt auftritt. Da es nur endlich viele Symbole gibt und Terminalsymbole auf einem Pfad ohnehin nicht doppelt auftreten und der Verzweigungsgrad der Bäume beschränkt ist, gibt es nur endlich viele Bäume in C_G. Nun zu den A_G. Es enthalte A_G alle Adjunktionsbäume $\mathfrak{B}_X$, $X \in N$, dergestalt, daß (1.) $\mathfrak{B}_X$ sich aus X in γG herleiten läßt, (2.) kein Symbol entlang eines Pfades, der nicht die Wurzel enthält, doppelt autritt. A_G ist endlich. Es ist nun nicht schwer zu zeigen, daß $L_B(\mathsf{Ad}_G) \subseteq L_B(G)$. Die umgekehrte Inklusion beweisen wir durch Induktion über die Anzahl der Knoten im Baum. Es sei $\mathfrak{B}$ aus $L_B(G)$. Entweder tritt entlang eines Pfades ein Symbol doppelt auf oder nicht. Im zweiten Fall ist $\mathfrak{B} \in \mathsf{A}_G$ und so $\mathfrak{B} \in L_B(\mathsf{Ad}_G)$. Im ersten Fall wählen wir $x \in B$ von minimaler Höhe, sodaß ein $y < x$ existiert mit gleicher Marke; nennen wir sie X. Betrachte den Teilbaum $\mathfrak{U}$ basiert auf der Menge $\downarrow x - \downarrow y \cup \{y\}$. Wir behaupten, daß $\mathfrak{U} \in \mathsf{A}_G$. Dazu ist zu zeigen. (a) $\mathfrak{U}$ ist ein Adjunktionsbaum, (b) $\mathfrak{U}$ läßt sich aus X herleiten, (c) kein Symbol tritt doppelt auf entlang eines Pfades, der x nicht enthält. Zu (a). Ein Blatt von $\mathfrak{U}$ ist entweder ein Blatt von $\mathfrak{B}$ oder aber $= y$. Im ersten Fall ist die Marke ein Terminalsymbol, im zweiten Fall gleich der Marke von x, nach Wahl von x und y. Zu (b). Falls $\mathfrak{B}$ ein Baum von G ist, so ist $\mathfrak{U}$ aus X herleitbar. Zu (c). Sei π ein Pfad, der x nicht enthält, und seien $u, v \in \pi$ Knoten mit gleicher Marke und $u < v$. Dann ist $v < x$, und dies widerspricht der Minimalität von x. Also sind alle drei Bedingungen erfüllt, und wir können $\mathfrak{U}$ gewissermaßen 'aushängen'. Das bedeutet, es gibt ein $\mathfrak{B}'$ derart, daß $\mathfrak{B}$ durch Adjunktion von $\mathfrak{U}$ aus $\mathfrak{B}'$ hervorgeht. Es ist $\mathfrak{B}' \in L_B(G)$, und nach Induktionsvoraussetzung daher $\mathfrak{B}' \in L_B(\mathsf{Ad}_G)$. Also $\mathfrak{B} \in L_B(\mathsf{Ad}_G)$, was zu beweisen war.

Theorem 2.5.10 (Joshi & Levy & Takahashi) *Jede von einer kontextfreien Grammatik erzeugbare Baummenge ist erzeugbar von einer unregulierten Baumadjunktionsgrammatik.* $\dashv$

Wir werden nun Parikh's Satz für unregulierte Baumadjunktionsgrammatiken beweisen. Dazu definieren wir für einen markierten Baum $\mathfrak{B}$ und ein Symbol α die Zahl $\sigma_\alpha(\mathfrak{B})$. Dies sei die Anzahl aller Knoten, welche das Symbol α tragen. Bei einem Adjunktionsbaum wollen wir jedoch das Symbol an der Wurzel *nicht* mitzählen. Sei nun $\langle \mathsf{C}, \mathsf{A} \rangle$ eine Adjunktionsgrammatik und $\mathsf{C} = \{\mathfrak{C}_i : i < \alpha\}$, $\mathsf{A} = \{\mathfrak{A}_j : j < \beta\}$.

Lemma 2.5.11 *Es entstehe $\mathfrak{B}'$ aus $\mathfrak{B}$ durch Adjunktion eines Baumes $\mathfrak{A}$. Dann gilt $\sigma_\alpha(\mathfrak{B}') = \sigma_\alpha(\mathfrak{B}) + \sigma_\alpha(\mathfrak{A})$.*

Der Beweis dieses Lemmas ist einfach. Es folgt aus diesem Lemma, daß wir in einer Ableitung lediglich zählen müssen, wie oft ein entsprechender Baum adjungiert worden ist, aber nicht, zu welchem Zeitpunkt. Sei nun $\mathfrak{B}$ ein Baum, der durch p_j–

fache Adjunktion von $\mathfrak{A}_j$ für $j < \beta$ aus $\mathfrak{C}_i$ enstanden ist. Dann ist

$$\sigma_\alpha(\mathfrak{B}) = \sigma_\alpha(\mathfrak{C}_i) + \sum_{i<\beta} p_j \cdot \sigma_\alpha(\mathfrak{A}_j) \ .$$

Sei nun $\mu(\mathfrak{B}) := \sum_{a\in A} \sigma_a(\mathfrak{B}) \cdot a$. Dann gilt

$$\mu(\mathfrak{B}) = \mu(\mathfrak{C}_i) + \sum_{i<\beta} p_j \cdot \mu(\mathfrak{B}_j)$$

Wir definieren nun die Mengen

$$\Sigma_i := \{\mu(\mathfrak{C}_i) + \sum_{j<\beta} p_j \cdot \mu(\mathfrak{A}_j) : p_j \in \omega\}$$

Dann gilt, daß $\mu[L_B(\langle \mathsf{C}, \mathsf{A}\rangle)] \subseteq \bigcup_{i<n} \Sigma_i$. Allerdings gilt nicht immer die Gleichheit. Folgendes muß man beachten. Ein Baum $\mathfrak{A}_j$ ist an einen Baum $\mathfrak{B}$ nur dann adjungierbar, wenn die Marke seiner Wurzel Marke irgendeines Knotens in $\mathfrak{B}$ ist. Daher sind nicht alle Elemente von $\bigcup \Sigma_i$ die Werte unter μ eines ableitbaren Baumes. Allerdings ist es so, daß wenn $\mathfrak{A}_j$ an $\mathfrak{B}$ adjungierbar ist, so ist er adjungierbar an allen Bäumen, die aus $\mathfrak{B}$ durch Adjunktion hervorgehen. Daher müssen wir unsere Mengen wie folgt modifizieren. Wir betrachten die Menge D aller Paare $\langle k, W\rangle$ dergestalt, daß $k < \alpha$, $W \subseteq \beta$ und es eine Ableitung eines Baumes gibt, die mit $\mathfrak{C}_k$ beginnt und genau die Bäume aus W mindestens einmal bei Adjunktion verwendet. Für $\langle k, W\rangle \in D$ sei

$$L(k, W) = \{\mu(\mathfrak{C}_i) + \sum_{j\in W} k_j \cdot \mu(\mathfrak{A}_j) : k_j \in \omega\} \ .$$

Dann ist $L := \bigcup\langle L(k, W) : \langle k, W\rangle \in D\rangle$ eine semilineare Menge. Gleichzeitig ist sie die Menge aller $\mu(\mathfrak{B})$, wo $\mathfrak{B}$ aus $\langle \mathsf{A}, \mathsf{C}\rangle$ ableitbar ist. Wir haben nun den

Theorem 2.5.12 (Parikh) *Sei S die Sprache der Zeichenketten einer unregulierten Baumadjunktionsgrammatik. Dann ist S semilinear. Insbesondere ist jede kontextfreie Sprache semilinear.* $\dashv$

Der Satz von Parikh ist in mancher Hinsicht bemerkenswert. Er wird uns noch öfter begegnen. Semilineare Mengen sind nach Satz 2.5.8 abgeschlossen unter Komplement und daher auch unter Schnitt. Wir werden hier zeigen, daß dies für semilineare Sprachen nicht gelten muß.

Proposition 2.5.13 *Es gibt kontextfreie Sprachen L_1 und L_2 derart, daß $L_1 \cap L_2$ nicht semilinear ist.*

Beweis. Sei $M_1 := \{\mathtt{a}^n\mathtt{b}^n : n \in \omega\}$ und $M_2 := \{\mathtt{b}^n\mathtt{a}^{2n} : n \in \omega\}$. Nun setze

$$\begin{array}{lcl} L_1 & := & \mathtt{b}M_1^*\mathtt{a}^* \\ L_2 & := & M_2^+ \end{array}$$

Aufgrund von Satz 1.5.8 sind L_1 und L_2 kontextfrei. Nun betrachte $L_1 \cap L_2$. Es ist leicht zu sehen, daß der Schnitt aus den folgenden Zeichenketten besteht

$$\mathtt{ba}^2, \quad \mathtt{ba}^2\mathtt{b}^2\mathtt{a}^4, \quad \mathtt{ba}^2\mathtt{b}^2\mathtt{a}^4\mathtt{b}^4\mathtt{a}^8, \quad \mathtt{ba}^2\mathtt{b}^2\mathtt{a}^4\mathtt{b}^4\mathtt{a}^8\mathtt{b}^8\mathtt{a}^{16}, \ldots$$

Das Parikh–Bild ist $\{(2^{n+1} - 2)\mathtt{a} + (2^n - 1)\mathtt{b} : n \in \omega\}$. Diese Menge ist nicht semilinear, denn das Resultat der Tilgung von $\mathtt{b}$ (das heißt der Projektion auf $\mathtt{a}^*$) ist nicht fast periodisch. $\dashv$

Wir wissen im Übrigen, daß zu jeder semilinearen Menge $N \subseteq M(A)$ eine reguläre Grammatik G dergestalt existiert, daß $\mu[L(G)] = N$. Allerdings kann G relativ kompliziert sein. Es stellt sich nun die Frage, ob etwa das volle Urbild $\mu^{-1}[N]$ regulär oder wenigstens kontextfrei ist. Dies ist nicht der Fall. Jedoch gilt

Theorem 2.5.14 *Das volle Urbild einer semilinearen Menge über einem Buchstaben ist regulär.*

Dies ist das beste Resultat, denn schon ab zwei Buchstaben ist dies falsch.

Theorem 2.5.15 *Das volle Urbild von $\omega(\mathtt{a} + \mathtt{b})$ ist nicht regulär aber kontextfrei. Das volle Urbild von $\omega(\mathtt{a} + \mathtt{b} + \mathtt{c})$ ist nicht kontextfrei.*

Beweis. Wir zeigen die zweite Behauptung zuerst. Sei

$$W := \mu^{-1}[\omega(\mathtt{a} + \mathtt{b} + \mathtt{c})]$$

Angenommen, W sei kontextfrei. Dann ist der Schnitt mit der regulären Sprache $\mathtt{a}^*\mathtt{b}^*\mathtt{c}^*$ auch kontextfrei. Dies ist aber gerade die Menge $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \in \omega\}$. Widerspruch. Nun zu der ersten Behauptung. Es bezeichne $b(\vec{x})$ die Anzahl der in $\vec{x}$ vorkommenden $\mathtt{a}$ minus der Anzahl der in $\vec{x}$ vorkommenden $\mathtt{b}$. Dann ist $V := \{\vec{x} : b(\vec{x}) = 0\}$ das volle Urbild von $\omega(\mathtt{a} + \mathtt{b})$. V ist nicht regulär; andernfalls ist der Schnitt mit $\mathtt{a}^*\mathtt{b}^*$ auch regulär. Aber dieser ist $\{\mathtt{a}^n\mathtt{b}^n : n \in \omega\}$. Widerspruch. V ist aber kontextfrei. Wir werden zum Beweis eine kontextfreie Grammatik G angeben, welche V erzeugt. Wir nehmen drei Nichterminalsymbole, $\mathtt{S}$, $\mathtt{A}$, und $\mathtt{B}$. Die Regeln seien

$$\begin{array}{lcll} \mathtt{S} & \to & \mathtt{SS} \mid \mathtt{AB} \mid \mathtt{BA} \mid x & x \in A - \{\mathtt{a}, \mathtt{b}\} \\ \mathtt{A} & \to & \mathtt{AS} \mid \mathtt{SA} \mid \mathtt{a} & \\ \mathtt{B} & \to & \mathtt{BS} \mid \mathtt{SB} \mid \mathtt{b} & \end{array}$$

Das Startsymbol sei $\mathtt{S}$. Wir behaupten: $\mathtt{S} \vdash_G \vec{x}$ genau dann, wenn $b(\vec{x}) = 0$, $\mathtt{A} \vdash_G \vec{x}$ genau dann, wenn $b(\vec{x}) = 1$ und $\mathtt{B} \vdash_G \vec{x}$ genau dann, wenn $b(\vec{x}) = -1$. Die Richtungen von links nach rechts sind einfach zu verifizieren. Daraus folgt dann, daß $V \subseteq L(G)$. Die anderen Richtungen beweisen wir durch Induktion über die Länge von $\vec{x}$. Offensichtlich genügt der Nachweis der folgenden Behauptung.

> Ist $b(\vec{x}) \in \{1, 0, -1\}$, so existieren $\vec{y}$ und $\vec{z}$ mit $|\vec{y}|, |\vec{z}| < |\vec{x}|$, sodaß $\vec{x} = \vec{y}\vec{z}$ und $b(\vec{y}), b(\vec{z}) \in \{-1, 0, 1\}$.

Sei also $\vec{x} = x_0 x_1 \dots x_{n-1}$ gegeben. Definiere $k(\vec{x}, j) := b(x_0 x_1 \dots x_{j-1})$, und $K := \{k(\vec{x}, j) : j < n + 1\}$. Diese Menge ist, wie man leicht sieht, ein Interval $[m, m']$ mit $m \leq 0$. Ferner ist $k(\vec{x}, n) = b(\vec{x})$. (a) Sei $b(\vec{x}) = 0$. Dann setze $\vec{y} := x_0$ und $\vec{z} := \prod_{0<i<n} x_i$. Dies erfüllt die Bedingungen. (b) Sei $b(\vec{x}) = 1$. Fall 1: $x_0 = \mathtt{a}$. Dann setze wiederum $\vec{y} := x_0$ und $\vec{z} := \prod_{0<i<n} x_i$. Fall 2: $x_0 = \mathtt{b}$. Dann ist $k(\vec{x}, 1) = -1$ und es existiert ein j mit $k(\vec{x}, j) = 0$. Setze $\vec{y} := \prod_{i<j} x_i$, $\vec{z} := \prod_{j \leq i < n} x_i$. Da $0 < j < n$, gilt $|\vec{y}|, |\vec{z}| < |\vec{x}|$. Ferner ist $b(\vec{y}) = 0$ und $b(\vec{z}) = 1$. (c) $b(\vec{x}) = -1$. Analog zu (b). $\dashv$

Übung 55. Es sei $|A| = 1$. Zeigen Sie, daß $\mathfrak{Z}(A)$ isomorph ist zu $\mathfrak{M}(A)$.

Übung 56. Sei $|A| = 1$ und Ad eine unregulierte Baumadjunktionsgrammatik. Zeigen Sie, daß die von Ad erzeugte Sprache über A^* regulär ist.

Übung 57. Es sei $A = \{\mathtt{a}, \mathtt{b}\}$. Ferner sei $U := \mathtt{a}^* \cup \mathtt{b}^*$. Sei nun $N \subseteq M(A)$ eine Menge derart, daß $N - U$ unendlich ist. Zeigen Sie, daß es $2^{\aleph_0}$ viele Sprachen S gibt mit $\mu[S] = N$. *Hinweis.* Es ist die Mächtigkeit von A^* gerade $\aleph_0$, also kann es in jedem Fall nicht mehr als $2^{\aleph_0}$ solche Sprachen geben. Zeigen Sie also nur, daß es mindestens so viele gibt.

Übung 58. Zeigen Sie Satz 2.5.14. *Hinweis.* Beschränken Sie sich zunächst auf den Fall, daß $A = \{\mathtt{a}\}$ ist.

Übung 59. Es sei $N \subseteq M(A)$ semilinear. Man zeige, daß das volle Urbild, $\mu^{-1}[N]$, kontextsensitiv ist. *Hinweis.* Man kann sich darauf beschränken, dies für lineare Mengen zu zeigen.

2.6 Sind natürliche Sprachen kontextfrei?

Wir beenden unsere Ausführungen über kontextfreie Grammatiken, indem wir natürliche Beispiele von nichtkontextfreien Sprachen vorstellen. Die Komplexität der natürlichen Sprachen stand eigentlich am Anfang der Chomsky–Hierarchie. Chomsky's Absicht war es, den Strukturalismus zu diskreditieren, indem er eine Hierarchie von Erzeugungsprozessen vorschlug und gleichzeitig behauptete, der Strukturalismus

bedeutete, daß natürliche Sprachen kontextfrei sein müßten. Um sein eigenes Programm, die Transformationsgrammatik, zu verteidigen, trat er außerdem den Beweis an, daß natürliche Sprachen nicht kontextfrei sind. Damit glaubte er den Strukturalismus widerlegt und ebenso die Notwendigkeit alternativer Sprachbeschreibungen klargestellt zu haben. (Man lese dazu die erhellende Darstellung in Manaster–Ramer und Kac [36].) Leider hat sich im Verlaufe der Debatte gezeigt, daß die Beweise, die von Chomsky und später von Postal gegeben wurden, nicht stichhaltig waren. Gazdar, Pullum und andere haben immer wieder Fehler in den Argumenten gefunden. Dies hat sie zu der These veranlaßt, natürliche Sprachen seien eben doch kontextfrei (siehe [15]). Der erste, der einen stichhaltigen Beweis des Gegenteils geliefert hat, war Riny Huybregts, dem wenig später Stuart Shieber folgte. (Siehe [26] und [47].) Dabei muß man noch klarstellen, daß bereits klar war, daß natürliche Sprachen gewiß nicht stark kontextfrei sind; nur erschien es immer noch möglich, daß sie wenigsten schwach kontextfrei sind.

Wie führt man nun den Beweis, daß eine Sprache S nicht kontextfrei ist? Dazu nimmt man sich eine geeignete reguläre Sprache R und schneidet sie mit S. Falls S kontextfrei ist, so muß dies auch $S \cap R$ sein. Anschließend wählt man einen geeigneten Homomorphismus, um $R \cap S$ so auf eine bekannte, nicht kontextfreie Sprache abzubilden. Wir geben das Beispiel von Shieber aus [47]. Betrachtet man geschachtelte Infinitive im Schweizerdeutschen, so fällt auf, daß sie ganz anders aufgebaut sind, als die entsprechenden Konstruktionen im Hochdeutschen. Hier ein paar Beispiele.

(2.1) `Jan säit, das Hans es huus aastricht.`
Jan sagt, daß Hans das Haus anstreicht.

(2.2) `Jan säit, das mer em Hans es huus hälfed aastriche.`
Jan sagt, daß wir Hans das Haus anstreichen helfen.

(2.3) `Jan säit, das mer d'chind em Hans es huus lönd hälfe aastriche.`
Jan sagt, daß wir die Kinder Hans das Haus anstreichen helfen lassen.

Indem wir fragen, wer was tut (wir lassen, die Kinder helfen, Hans streicht an), sehen wir, daß die Konstituenten im Hochdeutschen jeweils kontinuierlich sind, und daß die zusammengehörenden Element jeweils ineinander geschachtelt sind. Im Schweizerdeutschen ist dies genau anders. Die Infinitive folgen einander in der entgegengesetzten Reihenfolge. Wir nehmen nun an, dies sei so (das ist allerdings nicht ohne weiteres klar, da es letztlich eine empirische Frage ist). Dabei sei auch betont, daß die Schachtelungstiefe der Infinitive nicht beschränkt ist, auch wenn ab der Tiefe 4 erhebliche Verständnisschwierigkeiten entstehen. Die entstehenden Sätze gelten dessenungeachtet als grammatisch, wenn sie nach dem angegebenen Muster gebaut sind.

Man verfährt nun so. Die Verben verlangen entweder den Akkusativ oder den Dativ. Diese sind nicht austauschbar. Daß dem so ist, zeigt die Tatsachen, daß die folgenden Sätze ungrammatisch sind:

(2.5) *Jan säit, das mer de Hans es huus hälfed aastriche.

(2.6) *Jan säit, das mer em chind em Hans es huus lönd hälfe aastriche.

Wir nehmen nun die folgende reguläre Sprache:

$$R := \texttt{Jan säit, das} \cdot (\texttt{em} \cup \texttt{d'} \cup \texttt{de} \cup \texttt{mer} \cup \texttt{Hans} \cup \texttt{es} \cup \texttt{huus})^* \cdot (\texttt{laa} \cup \texttt{lönd} \cup \texttt{hälfe}) \cdot \texttt{aastriche}$$

Anschließend definieren wir folgenden Homomorphismus v. v sendet `d'`, `de`, `laa` und `lönd` auf `a`, `em` und `hälfe` auf `d`, alles andere wird auf ε abgebildet. Die Behauptung ist nun, daß

$$h[S \cap R] = \{\vec{x}\,\vec{x} : \vec{x} \in \texttt{a} \cdot (\texttt{a} \cup \texttt{d})^*\}$$

Dazu überlegen wir, daß ein Verb auf `d` gesendet wird, wenn es ein Dativobjekt hat, auf `a`, wenn es ein Akkusativobjekt hat. Ein Akkusativobjekt ist von der Form `de` N oder `d'` N (N ein Nomen) und wird von $\overline{v}$ auf `a` abgebildet. Ein Dativobjekt hat die Form `em` N, N ein Nomen, und wird auf `d` abgebildet. Da die Nomina in derselben Folge stehen wie die assoziierten Infinitive, bekommen wir das gewünschte Ergebnis.

Das nächste Beispiel in unserer Reihe ist angelehnt an den Beweis der Nichtkontextfreiheit von `Algol`. Es betrifft eine Sprache, die sehr universell ist, nämlich die Prädikatenlogik. Üblicherweise ist Prädikatenlogik mit Hilfe von endlich vielen Relations– und Funktionssymbolen, sowie einer endlichen Zahl von logischen Junktoren und einer abzählbar unendlichen Menge $\{x_i : i \in \omega\}$ von Variablen definiert. Um die Prädikatenlogik als Sprache in unserem Sinne auffassen zu können, kodieren wir die Variablen als Sequenzen $\alpha\texttt{x}\vec{\alpha}$, wobei $\vec{\alpha} \in \{\texttt{0}, \texttt{1}\}^+$. Es ist $\texttt{x}\vec{\alpha} = \texttt{x}\vec{\beta}$ genau dann, wenn $\vec{\alpha} = \vec{\beta}$. (Führende Nullen werden also nicht ignoriert.) Wir beschränken uns nun auf die reine Gleichungslogik. Das Alphabet ist $\{\forall, \exists, \texttt{(}, \texttt{)}, \doteq, \texttt{x}, \texttt{0}, \texttt{1}, \wedge, \neg, \rightarrow\}$. Wir haben folgende Regeln:

$$\begin{array}{lcl} \texttt{F} & \rightarrow & \texttt{QF} \mid \neg\texttt{(F)} \mid \texttt{(F} \wedge \texttt{F)} \mid \texttt{(F} \rightarrow \texttt{F)} \mid \texttt{P} \\ \texttt{P} & \rightarrow & \texttt{V} \doteq \texttt{V} \\ \texttt{Q} & \rightarrow & \forall\texttt{VF} \mid \exists\texttt{VF} \\ \texttt{V} & \rightarrow & \texttt{xZ} \\ \texttt{Z} & \rightarrow & \texttt{0Z} \mid \texttt{1Z} \mid \texttt{0} \mid \texttt{1} \end{array}$$

Hier definert `F` die Menge der Formeln, `P` die Menge der Primformeln, `Q` die Menge der Quantorenpräfixe, `V` die Menge der Variablen, und `E` die Menge der echten Zeichenketten aus `0` und `1`. Es sei $\vec{x}$ eine Formel und C ein Vorkommen einer Variablen $\texttt{x}\vec{\alpha}$. Wir sagen nun, dieses Vorkommen der Variablen sei **gebunden** in $\vec{x}$, falls es ein Vorkommen D einer Formel $(Q\texttt{x}\vec{\alpha})\vec{y}$ in $\vec{x}$ gibt mit $Q \in \{\forall, \exists\}$, welches C enthält. Eine Formel heißt **Satz**, falls jede Variable überall gebunden vorkommt.

Theorem 2.6.1 *Die Menge der Sätze der Prädikatenlogik ist nicht kontextfrei.*

Beweis. Es sei S die Menge der Sätze der reinen Gleichungslogik. (Es genügt, diese zu betrachten, wie wir gleich sehen werden.) Angenommen, die Menge ist kontextfrei. Dann gibt es nach dem Pumplemma ein k, derart, daß jede Zeichenkette der Länge $\geq k$ eine Zerlegung $\vec{u}\vec{x}\vec{v}\vec{y}\vec{w}$ besitzt, sodaß $\vec{u}\vec{x}^i\vec{v}\vec{y}^i\vec{z} \in S$ für alle i und $|\vec{x}\vec{v}\vec{y}| \leq k$. Betrachte die Formeln $\vec{p}_\alpha$

$$\forall \mathtt{x}\vec{\alpha}\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha}$$

Alle diese Formeln sind Sätze. Ist $\vec{\alpha}$ hinreichend lang (zum Beispiel länger als k), so existiert eine Zerlegung wie oben angegeben. Da nun $\vec{x}\vec{v}\vec{y}$ die Länge $\leq k$ haben muß, überlegt man sich, daß $\vec{x}$ und $\vec{y}$ nicht gleichzeitig disjunkt zu allen Vorkommen von $\vec{\alpha}$ sein können. Auf der anderen Seite folgt dann daraus, daß $\vec{x}$ und $\vec{y}$ nur aus $\mathtt{0}$ und $\mathtt{1}$ bestehen, und so notwendig zu mindestens einem Vorkommen von $\vec{\alpha}$ disjunkt sind. Wenn man nun $\vec{x}$ und $\vec{y}$ hochpumpt, so wird notwendig mindestens ein Vorkommen einer Variablen nicht gebunden sein. $\dashv$

Diese Ergebnis kann man noch erheblich verschärfen.

Theorem 2.6.2 *Die Menge der Sätze der Prädikatenlogik ist nicht semilinear.*

Beweis. Sei P die Menge der Sätze der Prädikatenlogik. Angenommen, P ist semilinear. Es sei P_1 die Menge aller Sätze, welche nur einen Quantor enthalten, und zwar nur einmal $\exists$. $\mu[P_1]$ ist der Schnitt von $\mu[P]$ mit der Menge aller Vektoren, deren $\exists$–Komponente 1 ist, und deren $\forall$–Komponente 0 ist. Diese ist also auch semilinear. Nun betrachten wir das Bild von $\mu[P_1]$ unter Löschung aller Symbole, welche verschieden von $\mathtt{x}$, $\mathtt{0}$ und $\mathtt{1}$ sind. Dies bezeichnen wir mit Q_1. Q_1 ist semilinear. Aufgrund der Konstruktion von P_1 existiert ein $\vec{\alpha} \in \{\mathtt{0},\mathtt{1}\}^*$ derart, daß jedes Vorkommen eine Variablen von der Form $\mathtt{x}\vec{\alpha}$ ist. Tritt die Variable also k mal auf und enthält $\vec{\alpha}$ p mal die $\mathtt{0}$ und q mal die $\mathtt{1}$, so bekommen wir als Ergebnis den Vektor $k\mathtt{x}+kp\mathtt{0}+kq\mathtt{1}$. Es ist leicht zu sehen, daß k ungerade sein muß. Denn in die Variable tritt einmal im Quantor auf, und ansonsten nur in Formeln der Form $\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha}$. Nun existieren die folgenden Sätze:

$$\exists \mathtt{x}\vec{\alpha}\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha}$$
$$\exists \mathtt{x}\vec{\alpha}(\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha} \wedge \mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha})$$
$$\exists \mathtt{x}\vec{\alpha}(\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha} \wedge (\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha} \wedge (\mathtt{x}\vec{\alpha} \doteq \mathtt{x}\vec{\alpha}))$$

Da wir jedes $\vec{\alpha}$ wählen können, ist

$$Q_1 = \{(2k+3)(\mathtt{x}+p\mathtt{0}+q\mathtt{1}) : k,p,q \in \omega\}$$

Q_1 ist eine unendliche Vereinigung von Ebenen der Form $(2k+3)(\mathtt{x}+\omega\mathtt{0}+\omega\mathtt{1})$. Wir zeigen: keine endliche Vereinigung von linearen Mengen ist gleich Q_1. Daraus ergibt

sich sofort ein Widerspruch. Sei etwa Q_1 Vereinigung der U_i, $i < n$, U_i linear. Dann existiert ein U_i, welches unendlich viele Vektoren der Form $(2k+3)\mathtt{x}$ enthält. Daraus leitet man leicht ab, daß U_i einen zyklischen Vektor der Form $m\mathtt{x}$, $m > 0$, haben muß. (Dies ist eine Übung.) Aber es ist klar, daß wenn $v \in Q_1$, so ist $m\mathtt{x} + v \notin Q_1$, und so haben wir einen Widerspruch. $\dashv$

Wir wollen nun ein recht einfaches Beispiel einer 'natürlichen' Sprache vorstellen, welche nicht semilinear ist. Dies wurde in etwas anderer Form von Arnold Zwicky vorgeschlagen. Es geht um die Zahlwörter des Deutschen. Der Vorrat an primitiven Namen für Zahlen ist endlich. Er umfaßt die Ziffern (`null` bis `neun`) die Namen für Zehner (`zehn` bis `neunzig`) sowie einige Namen für Zehnerpotenzen (`hundert`, `tausend`, `Million`, `Milliarde`, und so weiter). Nehmen wir einmal an, die größte Zehnerpotenz mit eigenem Nahmen sei 10^6. Wie benennt man dann größere Zahlen? Das Prinzip ist, Multiplikation durch einfache Iteration zu ersetzen. Die Zahl 10^{6k} wird durch k–faches Wiederholen des Wortes `Million` benannt. Zum Beispiel ist `eine Million Million Million` die Zahl 10^{18}. (Diese heißt auch eine `Trillion`. Das tut dem Beispiel keinen Abbruch.) Wie ist nun das Schema für korrekt gebildete Zahlnamen? Eine Zahl wird von rechts nach links in Blöcke zu je 6 Ziffern geteilt. Die Zahl wird also zerlegt in

$$\alpha_0 + \alpha_1 \times 10^6 + \alpha_2 \times 10^{12} \ldots$$

Hierbei sind die $\alpha_i < 10^6$. Nun wird die Zahl wie folgt benannt: [1]

$$\alpha_2 \ \mathtt{Million\ Million}\ \alpha_1 \ \mathtt{Million}\ \alpha_0$$

Ist $\alpha_i = 0$, so wird der $i$te Block ausgelassen. Es sei Z die Menge der Zahlnamen. Wir definieren folgende Abbildung φ. $\varphi(\mathtt{Million}) = \mathtt{b}$; alle anderen Zahlnamen werden auf `a` abgebildet. Wir bezeichnen das Parikh–Bild von $\varphi[Z]$ mit W. Es gilt nun

$$W = \{k_0\mathtt{a} + k_1\mathtt{b} : k_1 \geq \binom{\ulcorner k_0/9 \urcorner}{2}\}$$

Hierbei ist $\ulcorner k \urcorner$ die größte ganze Zahl unterhalb von k. Dies ist dem Leser als Übung überlassen. Wir werden beweisen, daß W nicht semilinear ist. Dies zeigt dann, daß Z nicht semilinear ist. Angenommen, W ist semilinear, etwa $W = \bigcup_{i<n} N_i$, wo die N_i linear sind. Es sei

$$N_i = u_i + \sum_{j<p_i} \omega v_j^i$$

[1] Wir ignorieren hier die Flektion der Zahlwörter. Es heißt also in unserem Beispiel 10^6 `eins Million` und nicht `eine Million`. 2×10^6 heißt `zwei Million` und nicht `zwei Millionen`. Dies macht das Leben etwas einfacher. Ferner heißt 73 `drei siebzig` und nicht `dreiundsiebzig`. Der Leser möge sich davon überzeugen, daß man die Definition von φ nur geschickt ändern muß, um diese Komplikationen miteinzubeziehen.

für gewisse u_i und $v_j^i = \lambda_j^i \mathtt{a} + \mu_j^i \mathtt{b}$. Angenommen, für gewisse i und j sei $\lambda_j^i \neq 0$. Betrachte die Menge

$$P := u_i + \omega v_j^i = \{u_i + k\lambda_j^i \mathtt{a} + k\mu_j^i \mathtt{b} : k \in \omega\}$$

Gewiß ist $P \subseteq N_i \subseteq W$. Ferner ist sicher $\mu_j^i \neq 0$. Setze nun $\zeta := \lambda_j^i / \mu_j^i$. Dann ist

$$P = \{u_i + k\mu_j^i(\mathtt{a} + \zeta \mathtt{b}) : k \in \omega\}$$

Daher gilt

Lemma 2.6.3 *Für jedes $\varepsilon > 0$ sind fast alle Elemente von P von der Form $p\mathtt{a}+q\mathtt{b}$, wobei $q/p \leq \zeta + \varepsilon$.*

Beweis. Es sei $u_i = x\mathtt{a} + y\mathtt{b}$. Dann ist ein allgemeines Element von der Form $(x + k\lambda_j^i)\mathtt{a} + (y + k\mu_j^i)\mathtt{b}$. Wir haben zu zeigen, daß für fast alle k die Ungleichung

$$\frac{x + k\lambda_j^i}{y + k\mu_j^i} \leq \lambda_{\mathtt{j}}^{\mathtt{i}} \mu_j^i + \zeta$$

erfüllt ist. Sei $k \geq \frac{x}{\varepsilon \mu_i^j}$. Dann ist

$$\frac{x + k\lambda_j^i}{y + k\mu_i^j} \leq \frac{x + k\lambda_j^i}{x + k\mu_j^i} = \zeta + \frac{x}{\frac{x}{\varepsilon \mu_j^i} \cdot \mu_j^i} =\leq \zeta + \varepsilon$$

Dies war zu zeigen. ⊣

Lemma 2.6.4 *Fast alle Punkte von P sind außerhalb von W.*

Beweis. Es sei n_0 so gewählt, daß $\binom{\lceil n_0/9 \rceil}{2} > n_0(\zeta+1)$. Dann ist für alle $n \geq n_0$ auch $\binom{\lceil n/9 \rceil}{2} > n(\zeta + 1)$. Sei $p\mathtt{a} + q\mathtt{b} \in W$ mit $p \geq n_0$. Dann ist $\frac{q}{p} > \zeta + \varepsilon$, und deswegen $p\mathtt{a} + q\mathtt{b} \notin P$. Also sind P und W disjunkt auf der Menge $H := \{p\mathtt{a} + q\mathtt{b} : p \geq n_0\}$. Aber $P \cap -H$ ist sicherlich endlich. ⊣

Wir haben nun den gewünschten Widerspruch; denn einerseits ist kein Vektor ein Vielfaches von $\mathtt{a}$, andererseits darf es keinen Vektor $m\mathtt{a} + n\mathtt{b}$ mit $n \neq 0$ geben. Also ist W nicht semilinear.

Übung 60. Man zeige: es sei U eine lineare Menge, welche unendlich viele Vektoren der Form $k\mathtt{a}$ enthält. Dann existiert ein zyklischer Vektor der Form $m\mathtt{a}$, $m > 0$. *Hinweis.* Man beachte, daß das Alphabet aus mehr als einem Buchstaben bestehen kann.

Übung 61. Man zeige, daß W die angegebene Form hat.

Übung 62. Man zeige, daß die Menge V nicht semilinear ist.

$$V = \left\{ k_0\mathtt{a} + k_1\mathtt{b} : k_1 \leq \binom{k_0}{2} \right\}$$

Hinweis. Offensichtlich darf keine lineare Menge $\subseteq V$ einen Vektor $k\mathtt{b}$ entalten. Dann ist also

$$\gamma := max \left\{ \frac{\mu^i_j}{\lambda^i_j} : i < n, j < p_i \right\}$$

wohlbestimmt. Zeigen Sie nun, daß für jedes $\varepsilon > 0$ fast alle Elemente von W von der Form $x\mathtt{a} + y\mathtt{b}$ sind, wo $y \leq (\gamma + \varepsilon)x$. Setzen wir zum Beispiel $\varepsilon = 1$ so erhalten wir nun einen Widerspruch.

Übung 63. Man beweise die eindeutige Lesbarkeit der Prädikatenlogik. *Hinweis.* Da wir streng genommen keine Terme definiert haben, begnüge man sich mit dem Nachweis, daß die gegebene Grammatik nicht ambig (sogar nicht opak) ist.

Kapitel 3

Kategorialgrammatik und Formale Semantik

3.1 Sprachen als Zeichensysteme

Sprachen sind gewiß nicht Mengen von Zeichenketten, sondern vor allem Kommunikationssysteme. Dies bedeutet, daß jede Zeichenkette einer Sprache auch eine Bedeutung hat. Da eine Sprache im allgemeinen unendlich viele Zeichenketten hat, muß es ein System geben, wie man einer beliebiger Zeichenkette eine Bedeutung zuordnen kann. Dabei spielt das Prinzip der sogenannten *Kompositionalität* eine wesentliche Rolle. Es besagt schlicht, daß die Bedeutung einer Zeichenkette nur von ihrem Analysebaum abhängt, und zwar in der folgenden Weise. Ist $\rho = X \to \alpha_0\alpha_1 \dots \alpha_{n-1}$ eine Regel, und ist $\vec{u}_i$ eine Zeichenkette des Typs α_i, so ist $\vec{v} := \vec{u}_0\vec{u}_1 \dots \vec{u}_{n-1}$ eine Zeichenkette des Typs X, und die Bedeutung von $\vec{v}$ hängt nur ab von den $\vec{u}_i$ und ρ. In dieser Form ist das Kompositionalitätsprinzip allerdings reichlich vage, und wir werden es daher im Verlaufe dieses Kapitels weiter präzisieren. Zunächst bleiben wir jedoch bei der gegebenen Definition. Es fällt auf, daß wir nur kontextfreie Regeln zugelassen haben. Dies ist, wie wir wissen, eine Einschränkung; wir werden im folgenden Kapitel allerdings zeigen, wie man sich von ihr wieder befreien kann.

Wir nehmen nun zunächst an, Bedeutungen entstammen einer nicht näher spezifierten Menge M. Zeichenketten sind nach wie vor Elemente aus A^*, wo A ein endliches Alphabet ist.

Definition 3.1.1 *Eine **interpretierte Sprache** über dem **Alphabet** A und mit **Bedeutungen** in M ist eine Relation $\mathfrak{I} \subseteq A^* \times M$. Die zu $\mathfrak{I}$ gehörende **Zeichen-***

kettensprache *ist*

$$S(\mathfrak{I}) := \{\vec{x} : \text{ es existiert } m \in M \text{ mit } \langle \vec{x}, m\rangle \in \mathfrak{I}\}$$

Die durch $\mathfrak{I}$ *ausgedrückten Bedeutungen sind*

$$M(\mathfrak{I}) := \{m : \text{ es existiert } \vec{x} \in A^* \text{ mit } \langle \vec{x}, m\rangle \in \mathfrak{I}\}$$

Wir können diese Definition auch etwas anders aufziehen. Wir können eine Sprache als Funktion f von A^* nach $\wp(M)$ definieren. Dann ist $S(f) := \{\vec{x} : f(\vec{x}) \neq \varnothing\}$ die zu f gehörende Zeichenkettensprache und $M(f) := \bigcup_{\vec{x}\in A^*} f(\vec{x})$ die Menge der ausgedrückten Bedeutungen von f. Beide Definitionen sind äquivalent.

Wir wollen ein Beispiel bringen, an welchem die gesamte Problematik bereits deutlich wird. Wir betrachten die Menge der Zahlterme, wie sie aus dem normalen Leben bekannt sind, zum Beispiel $\mathtt{((3+5)\times 2)}$. Wir wollen eine Grammatik schreiben, mit der man einen Term ausrechnen kann, wenn man seine Analyse kennt. Das bedeutet also, daß wir als interpretierte Sprache die Menge aller Paare $\langle t, x\rangle$ nehmen, wo t ein arithmetischer Term ohne Variablen ist und x sein Wert. Natürlich gibt die Analyse den Wert des Terms nicht direkt her, sondern wir wollen zusätzlich zur Grammatik noch Regeln angeben, welche uns sagen, wie wir induktiv über die Tiefe eines Knotens jedem Knoten seinen Wert zuordnen. Da die Knoten den jeweiligen Teiltermen entsprechen werden, ist dies in der Tat eine harmlose Sache. Es sei T die folgende Grammatik.

$$\begin{array}{lcl}
\mathtt{T} & \to & \mathtt{(T+T)} \mid \mathtt{(T-T)} \mid \mathtt{(T\times T)} \mid \mathtt{(T\div T)} \\
\mathtt{T} & \to & \mathtt{Z} \mid \mathtt{(-Z)} \\
\mathtt{Z} & \to & \mathtt{0} \mid \mathtt{1} \mid \mathtt{2} \mid \ldots \mid \mathtt{9}
\end{array}$$

(Diese Grammatik erzeugt nur Terme mit Ziffern als Zahlen. Dies ist also nur als Beispiel zu verstehen.) Sei nun ein beliebiger Term gegeben. Dann entspricht diesem Term eine eindeutig bestimmte Zahl (sofern wir von Division durch 0 absehen). Diese kann durch Induktion über die Ableitung berechnet werden. Dazu definieren wir eine Interpretationsabbildung I, welche jeder Zeichenkette eine Zahl oder $\star$ zuordnet:

$$\begin{array}{lcl}
I(\mathtt{(}\vec{x}+\vec{y}\mathtt{)}) & := & I(\vec{x}) + I(\vec{y}) \\
I(\mathtt{(}\vec{x}-\vec{y}\mathtt{)}) & := & I(\vec{x}) - I(\vec{y}) \\
I(\mathtt{(}\vec{x}\times\vec{y}\mathtt{)}) & := & I(\vec{x}) \times I(\vec{y}) \\
I(\mathtt{(}\vec{x}\div\vec{y}\mathtt{)}) & := & I(\vec{x}) \div I(\vec{y}) \\
I(-\vec{x}) & := & -I(\vec{x}) \\
I(\mathtt{0}) & := & 0 \\
I(\mathtt{1}) & := & 1 \\
 & \ldots & \\
I(\mathtt{9}) & := & 9
\end{array}$$

Man beachte, daß $\star$ eine besondere Rolle zukommt. So ist eine Operation immer dann $\star$, falls eines der Argumente $\star$ ist; ferner ist $x \div 0 = \star$. Auf diese Weise kann man die Undefiniertheit handhaben. Wir bemerken nun Folgendes. Ist $\vec{x}$ ein Term, so ist $I(\vec{x})$ in eindeutiger Weise bestimmt. Denn entweder ist $\vec{x}$ eine Ziffer von `0` bis `9`, oder eine negative Ziffer, oder aber $\vec{x} = (\vec{y}_1 \odot \vec{y}_2)$, für gewisse, eindeutig bestimmte $\vec{y}_1$ und $\vec{y}_2$ und $\odot \in \{+, -, \times, \div\}$. Auf diese Weise kann man $I(\vec{x})$ ausrechnen, wenn man $I(\vec{y}_1)$ und $I(\vec{y}_2)$ kennt. Der Wert eines Terms bestimmt sich also, indem man zunächst eine Ableitung berechnet, und dann von unten nach oben den Wert eines jeden Teilterms bestimmt. Man beachte nämlich, daß die Grammatik transparent ist, sodaß jeweils nur eine syntaktische Analyse existieren kann.

Das eben vorgestellte Verfahren hat allerdings einen Nachteil: die Interpretation eines Terms ist im allgemeinen nicht eindeutig, zum Beispiel dann, wenn eine Zeichenkette ambig ist. (Wir könnten etwa in unserem Beispiel die Klammern weglassen.) Wir können dies beheben, indem wir anstatt einer Funktion von Zeichenketten in die Zahlen mit $\star$ nunmehr eine Funktion von Zeichenkette in Mengen von Zahlen (oder $\star$) definieren. Ist die Sprache eindeutig, so ist $I(\vec{x})$ eine Menge mit höchstens einem Element. Ferner ist $I(\vec{x}) \neq \varnothing$ genau dann, wenn $\vec{x}$ eine Konstituente ist. Es gibt aber einen wesentlich eleganteren Weg. Betrachten wir die Grammatik U, welche aus T durch Löschung der Klammersymbole entsteht.

$$\begin{array}{lcl} \mathtt{T} & \to & \mathtt{T}+\mathtt{T} \mid \mathtt{T}-\mathtt{T} \mid \mathtt{T}\times\mathtt{T} \mid \mathtt{T}\div\mathtt{T} \\ \mathtt{T} & \to & \mathtt{Z} \mid -\mathtt{Z} \\ \mathtt{Z} & \to & \mathtt{0} \mid \mathtt{1} \mid \mathtt{2} \mid \ldots \mid \mathtt{9} \end{array}$$

Wir können die Zeichenketten von U als Bilder einer kanonischen trasparenten Grammatik betrachten. Dies könnte zum Beispiel T sein, aber aus gewissen Gründen wählen wir eine andere Grammatik. Intuitiv denken wir uns die Zeichenketten als Bilder von Termen, welche den Derivationsbaum kodieren. Der Derivationsbaum unterscheidet sich vom Strukturbaum dadurch, daß die intermediären Symbole nicht die Nichtterminalsymbole sind, sondern die Regelsymbole. Diesen Derivationsbaum kodieren wir als Term in polnischer Notation. Wir nehmen zu jeder Regel ρ ein neues Symbol $\mathtt{R}_\rho$ hinzu. Anstelle einer Regel $A \to \vec{\alpha}$ nehmen wir die Regel $A \to \mathtt{R}_\rho\vec{\alpha}$. Diese Grammatik, nennen wir sie V, ist transparent. $\vec{x} \in L(V)$ heißt ein **Ableitungsterm**. Wir definieren nun zwei Abbildungen, ζ und ι. ζ liefert zu jedem Ableitungsterm eine Zeichenkette, und ι liefert zu jedem Ableitungsterm eine Interpretation. Beide Abbildungen werden als Homomorphismen der Termalgebra konstruiert, obwohl die konkrete Definition nur über die Zeichenketten rekurriert. ζ kann uniform definiert werden durch Löschung der Symbole $\mathtt{R}_\rho$. Man beachte jedoch, daß die nachstehenden Regeln nur dann ein Ergebnis liefern, wenn die gegebene Zeichenkette ein Ableitungsterm ist.

$$\begin{array}{lcl} \zeta(\mathtt{R}_\rho\vec{\alpha}_0 \ldots \vec{\alpha}_{n-1}) & := & \zeta(\alpha_0) \cdot \zeta(\alpha_1) \cdot \ldots \cdot \zeta(\alpha_{n-1}) \\ \zeta(\alpha) & := & \alpha \end{array}$$

Hierbei ist α verschieden von allen $\mathtt{R}_\rho$. Wir haben hierbei vorausgesetzt, daß die Grammatik keine Regeln der Form $A \to \varepsilon$ besitzt, obwohl eine leichte Änderung auch in diesem Fall zum Erfolg verhilft. Nun zur Definition von ι. In unserem konkreten Fall ist die Sache problemlos:

$$\begin{array}{lcl}
\iota(\mathtt{R}_+\vec{\alpha}_0 + \vec{\alpha}_1) & := & \iota(\vec{\alpha}_0) + \iota(\vec{\alpha}_1) \\
\iota(\mathtt{R}_{-2}\vec{\alpha}_0 - \vec{\alpha}_1) & := & \iota(\vec{\alpha}_0) - \iota(\vec{\alpha}_1) \\
\iota(\mathtt{R}_\times\vec{\alpha}_0 \times \vec{\alpha}_1) & := & \iota(\vec{\alpha}_0) \times \iota(\vec{\alpha}_1) \\
\iota(\mathtt{R}_\div\vec{\alpha}_0 \div \vec{\alpha}_1) & := & \iota(\vec{\alpha}_0) \div \iota(\vec{\alpha}_1) \\
\iota(\mathtt{R}_{-1} - \vec{\alpha}) & := & -\iota(\vec{\alpha})
\end{array}$$

Dabei haben wir die Ableitungsterme in Polnische Notation gesetzt. Wie wir wissen, sind sie eindeutig lesbar. Allerdings gilt dies nur unter der Bedingung, daß jedes einzelne Symbol eindeutig lesbar ist. Man beachte, daß eine Symbol mehrere Bedeutungen haben kann — wie in unserem Beispiel das Minuszeichen. Deswegen haben wir zusätzlich zur Annotation mit den $\mathtt{R}$ auch noch eine Unterscheidung dieser Symbole eingeführt. Wir haben oben zwischen dem binären und dem unären Minuszeichen unterschieden. Da dies in der eigentlichen Zeichensprache nicht geschieht (wir schreiben unterschiedslos $-$), wird dies in der Annotation der Regeln nachgeholt: wir haben $\mathtt{R}_{-1}$ für die einstellige Regel und $\mathtt{R}_{-2}$ für die zweistellige Regel.

Die Abbildung ι ist ein Homomorphismus von der Algebra der Ableitungsterme in die Algebra der Zahlen (mit $\star$). Hierbei interpretierten wir das Symbol $\mathtt{R}_+$ durch die zweistellige Funktion $+ : \mathbb{R}_\star \times \mathbb{R}_\star \to \mathbb{R}_\star$, wobei $\mathbb{R}_\star := \mathbb{R} \cup \{\star\}$, und $\star$ den obenstehenden Gesetzen genügt. In Prinzip läßt sich diese Algebra gegen jede andere austauschen, welche es erlaubt, ein- und zweistellige Funktionen zu definieren. Wir weisen darauf hin, daß es nicht nötig ist, daß die Zielalgebra die gewählten Funktionen als Basisfunktionen besitzt. Es genügt, wenn man sie aus den Basisfunktionen aufbauen kann. Zum Beispiel können wir ein einstelliges Fuktionssymbol $\mathtt{d}$ einführen, dessen Interpretation gerade die Verdopplung ist. Nun ist $2x = x + x$, und somit die Verdopplung eine Funktion unserer Algebra, aber keine Basisfunktion. Der Begriff des Aufbauens von Funktionen wird uns bald näher interessieren.

Diese Ausführungen motivieren eine Begriffsbildung, welche die Bedeutungen und die Zeichenketten jeweils als Bilder von Homomorphismen eines abstrakten Zeichens konstruieren. Diese Idee wollen wir jetzt in voller Allgemeinheit entwickeln. Grundlage dazu ist der Begriff einer Zeichenalgebra.

Definition 3.1.2 *Es sei $\langle F, \Omega\rangle$ eine Signatur. Eine **partielle** Ω–**Algebra** ist ein Paar $\mathfrak{A} = \langle A, \Pi\rangle$, wo A eine beliebige nichtleere Menge ist und für $f \in F$ $\Pi(f)$ eine partielle, $\Omega(f)$–stellige Funktion auf A. Wir schreiben $f^{\mathfrak{A}}$ anstelle von $\Pi(f)$. Ist $\mathfrak{B} = \langle B, \{f^{\mathfrak{B}} : f \in F\}\rangle$ eine partielle Ω–Algebra und $h : A \to B$, so heißt h **Homomorphismus** von $\mathfrak{A}$ nach $\mathfrak{B}$, falls gilt: ist $\vec{a} \in A^{\Omega(f)}$ und ist $f^{\mathfrak{A}}(\vec{a})$ definiert,*

so ist $f^{\mathfrak{B}}(h(\vec{a}))$ *ebenfalls definiert und*

$$f^{\mathfrak{B}}(h(\vec{a})) = h(f^{\mathfrak{A}}(\vec{a})) \ .$$

Wir sagen, h *sei* ***starker Homomorphismus****, falls* $f^{\mathfrak{A}}(\vec{a})$ *immer dann definiert ist, wenn* $f^{\mathfrak{B}}(h(\vec{a}))$ *definiert ist. Ist* $B \subseteq A$*, so ist* $\mathfrak{B}$ ***Unteralgebra*** *von* $\mathfrak{A}$*, falls die Einbettung* $i : B \to A : x \mapsto x$ *ein starker Homomorphismus ist.*

Eine Unteralgebra ist also schlicht durch ihre Menge B bestimmt. Die Funktionen sind dann die Einschränkungen der Funktionen auf A. Man beachte, daß es nicht erlaubt ist, zusätzlich Funktionen zu partialisieren. Zum Beispiel ist $\langle A, \Xi \rangle$ mit $\Xi(f) = \varnothing$ keine Unteralgebra von $\mathfrak{A}$, wenn $\Pi(f) \neq \varnothing$.

Ein **Zeichen** ist anschaulich ein Tripel $\sigma = \langle E, T, M \rangle$, wo E der Exponent von σ ist, üblicherweise eine Zeichenkette über einem Alphabet A, T der Typ von σ und M seine Bedeutung. Dies läßt sich abstrakt wie folgt fassen. Wir fixieren zunächst eine Signatur $\langle F, \Omega \rangle$. Die Funktionssymbole aus F nennen wir in diesem Zusammenhang auch **Modi**. Zu dieser Signatur definieren wir eine Zeichenalgebra sowie gewisse Algebren von Exponenten, Typen und Bedeutungen. Sei dazu $\langle F, \Omega \rangle$ eine Signatur. Eine Zeichenalgebra über dieser Signatur ist dann schlicht eine 0–erzeugte partielle Algebra $\mathfrak{A}$ über dieser Signatur zusammen mit gewissen Homomorphismen, welche weiter unten erklärt werden.

Definition 3.1.3 *Eine (partielle)* Ω*–Algebra* $\mathfrak{A} = \langle A, \Pi \rangle$ *heißt* n*–****erzeugt****, falls es eine* n*–elementige Teilmenge* $X \subseteq A$ *gibt, sodaß die kleinste,* X *enthaltende (partielle) Unteralgebra gerade* $\mathfrak{A}$ *ist.*

Definition 3.1.4 *Eine* ***Zeichengrammatik*** *über der Signatur* Ω *ist ein Quadrupel* $\langle \mathfrak{A}, \varepsilon, \tau, \mu \rangle$*, wobei* $\mathfrak{A}$ *eine 0–erzeugte partielle* Ω*–Algebra ist und* $\varepsilon : \mathfrak{A} \to \mathfrak{E}$*,* $\tau : \mathfrak{A} \to \mathfrak{T}$ *und* $\mu : \mathfrak{A} \to \mathfrak{M}$ *Homomorphismen nach gewissen partiellen* Ω*–Algebren derart, daß der Homomorphismus* $\langle \varepsilon, \tau, \mu \rangle$ *injektiv und stark ist.* $\mathfrak{A}$ *heißt* ***Algebra der Zeichen****,* $\mathfrak{E}$ *Algebra der* ***Exponenten****,* $\mathfrak{T}$ *Algebra der* ***Typen*** *und* $\mathfrak{M}$ *Algebra der* ***Bedeutungen****.*

Dies bedeutet im Einzelnen:

- Jedes Zeichen σ ist eindeutig durch drei Angaben bestimmt:
  - seinen sogenannten **Exponenten** $\varepsilon(\sigma)$ (üblicherwiese eine Zeichenkette)
  - seine syntaktische Kategorie $\tau(\sigma)$ (welche wir auch seinen **Typ** nennen)

– seine Bedeutung $\mu(\sigma)$.

* Jedem Funktionssymbol $f \in F$ entspricht eine $\Omega(f)$–stellige Funktion f^{ε} in $\mathfrak{E}$, eine $\Omega(f)$–stellige Funktion f^{τ} in $\mathfrak{T}$ und eine $\Omega(f)$–stellige Funktion f^{μ} in $\mathfrak{M}$.

* Zeichen können mit Hilfe der Funktion $f^{\mathfrak{A}}$ kombiniert werden, wann immer ihre Exponenten mit f^{ε}, ihre Kategorien mit f^{τ} und ihre Bedeutungen mit f^{μ} kombiniert werden können. (Dies ist die Starkheit.)

Ist σ ein Zeichen, so ist $\langle \varepsilon(\sigma), \tau(\sigma), \mu(\sigma) \rangle$ einerseits eindeutig durch σ bestimmt, andererseits bestimmt es auch σ eindeutig. Wir nennen dieses Tripel die **Realisierung** von σ. Andererseits können wir σ durch einen Term in der freien Ω–Algebra darstellen. Um die Wechselbeziehung dieser Sichtweisen wird es uns jetzt gehen.

Ist $\mathfrak{A}$ eine Algebra, so ist die 0–erzeugte Unteralgebra von $\mathfrak{A}$ eindeutig bestimmt, denn sie ist die kleinste Unteralgebra von $\mathfrak{A}$. Es sei $\mathfrak{F}$ die freie 0–erzeugte (nicht partielle) Ω–Algebra. Diese ist eindeutig bestimmt, da in ihr alle Operationen total sind und nicht partiell. Die Elemente von $\mathfrak{F}$ heißen **Strukturterme**. Strukturterme werden in polnischer Notation geschrieben. Ist zum Beispiel `N` ein nullstelliger Modus, und ist `S` ein 1–stelliger Modus, so bekommen wir folgende Strukturterme.

$$\mathtt{N}, \mathtt{SN}, \mathtt{SSN}, \mathtt{SSSN}, \ldots$$

Wie oben angemerkt, existiert zu jedem Zeichen mindestens ein Strukturterm. Die Umkehrung muß aber nicht gelten. Wir wollen daher sagen, ein Strukturterm σ sei **orthographisch definit**, falls $\varepsilon(\sigma)$ definiert ist. σ heißt **syntaktisch definit**, falls $\tau(\sigma)$ definiert ist und **semantisch definit**, falls $\mu(\sigma)$ definiert ist. σ heißt schließlich **definit**, falls er sowohl orthographisch, wie syntaktisch wie auch semantisch definit ist. $\mathfrak{A}$ ist also isomorph zur partiellen Algebra aller $\langle \varepsilon(\sigma), \tau(\sigma), \mu(\sigma) \rangle$, wo σ ein definiter Strukturterm ist. Dies können wir auch etwas anders angehen. Wir bezeichnen nun mit $\mathfrak{D}$ die Algebra der definiten Strukturterme. $\mathfrak{D}$ ist nicht notwendigerweise eine Unteralgebra von $\mathfrak{F}$, da ja in $\mathfrak{F}$ alle Funktionen total sind. Aber es existieren eindeutige Homomorphismen $\varepsilon^D : \mathfrak{D} \to \mathfrak{E}$, $\tau^D : \mathfrak{D} \to \mathfrak{T}$ und $\mu^D : \mathfrak{D} \to \mathfrak{M}$ derart, daß der Homomorphismus $\delta := \langle \varepsilon^D, \tau^D, \mu^D \rangle : \mathfrak{D} \to \mathfrak{E} \times \mathfrak{T} \times \mathfrak{M}$ stark ist. Betrachte jetzt folgende Relation $\Theta := \{\langle \sigma_0, \sigma_1 \rangle : \delta(\sigma_0) = \delta(\sigma_1)\}$. Dies ist eine Kongruenz auf $\mathfrak{D}$; die Relation ist eine Äquivalenzrelation und es ist, falls $\sigma_i \; \Theta \; \tau_i$ für alle $i < \Omega(f)$, so ist $f(\vec{\sigma})$ genau dann definiert, wenn $f(\vec{\tau})$ definiert, und $f(\vec{\sigma}) \; \Theta \; f(\vec{\tau})$. Wir können nunmehr setzen:

$$f^{\mathfrak{A}}(\langle [\sigma_i]_\Theta : i < \Omega(f) \rangle) := [f(\langle \sigma_i : i < \Omega(f) \rangle)]_\Theta$$

Dies ist wohldefiniert, und wir erhalten eine Algebra, die Algebra $\mathfrak{D}/\Theta$. Es ist nun Folgendes leicht zu sehen.

Proposition 3.1.5

$$\mathfrak{A} \cong \mathfrak{D}/\Theta$$

$\mathfrak{D}/\Theta$ ist also isomorph zu der Algebra der Zeichen. Man beachte also, daß zu jedem Zeichen sicher ein Strukturterm gehört, aber unter Umständen auch mehrere. Als instruktives Beispiel betrachten wir die Algebra der Reste modulo 60. Als Modi nehmen wir $\mathtt{N}$ (die Null; nullstellig) und $\mathtt{S}$ (die Nachfolgerfunktion; 1–stellig). Dann ist $\mathtt{S}$ sogar total definiert, aber es gilt

$$\mathtt{N} = \mathtt{S}^{60}\mathtt{N} \ .$$

Es hat jedes Zeichen sogar unendlich viele Strukturterme, ist also unendlich strukturell mehrdeutig. Dies ist zum Beispiel die Zeichenalgebra, die wir bekommen, wenn wir den Minutenzeiger einer Digitaluhr betrachten. $\mathtt{N}$ bezeichne die senkrechte Stellung und $\mathtt{S}$ die Funktion des Vorrückens um eine Minute. Als Bedeutung nehmen wir die Anzahl der Minuten, die ab der letzten vollen Stunde vergangen sind. In diesem Fall ist also jedes Zeichen strukturell mehrdeutig. Falls wir als Bedeutung übrigens die natürlichen Zahlen nehmen, und wenn wir $\mathtt{N}^\mu := 0$ sowie $\mathtt{S}^\mu := \lambda n.n + 1$ setzen, so repräsentiert jeder Strukturterm ein anderes Zeichen! Allerdings gibt es immer noch nur 60 Exponenten, aber jeder hat unendlich viele Bedeutungen.

Wir illustrieren die Konzeption einer Zeichengrammatik, indem wir unser Beispiel fortsetzen. Unser Alphabet ist $R := \{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}, \times, -, +, \div, \mathtt{(}, \mathtt{)}\}$. Die Algebra $\mathfrak{E}$ besteht aus R^* zusammen mit einigen Funktionen, welche wir im Einzelnen noch bestimmen werden. Wir beginnen mit der Bestimmung der Modi. Es gibt derer $\mathtt{R}_+$, $\mathtt{R}_{-2}$, $\mathtt{R}_\times$, $\mathtt{R}_\div$, welche jeweils 2–stellig sind, $\mathtt{R}_{-1}$, $\mathtt{T}$, welche einstellig sind und schließlich noch zehn nullstellige Modi $\mathtt{Z}_0$, $\mathtt{Z}_1$, …, $\mathtt{Z}_9$.

Wir beginnen mit den nullstelligen Modi. Diese sind — der Definition nach — bereits Zeichen. Zu ihrer Identifikation genügt also die Angabe der drei Komponenten. Zum Beispiel enstpricht dem Modus $\mathtt{Z}_0$ das Tripel $\langle \mathtt{0}, \mathtt{Z}, 0\rangle$. Dies bedeutet: der Exponent des Zeichens $\mathtt{Z}_0$ (also das, was wir letzlich sehen) ist die Ziffer $\mathtt{0}$; seine Kategorie ist $\mathtt{Z}$ und seine Bedeutung ist die Zahl 0. Ebenso mit den anderen nullstelligen Modi. Nun zu den einstelligen Modi. Diese sind Operationen von Zeichen nach Zeichen. Beginnen wir mit $\mathtt{R}_{-1}$. Auf der Ebene der Zeichenketten bekommen wir das Polynom $\mathtt{R}^\varepsilon_{-1}$, welches wie folgt definiert ist:

$$\mathtt{R}^\varepsilon_{-1}(\vec{x}) := \mathtt{(} - \vec{x}\mathtt{)}$$

Auf der Ebene der Kategorien bekommen wir die Funktion

$$\mathtt{R}^\tau_{-1}(K) := \begin{cases} \mathtt{T} & \text{falls } K = \mathtt{Z} \\ \star & \text{sonst} \end{cases}$$

Hierbei ist $\star$ ein Symbol dafür, daß die Funktion undefiniert ist. Schließlich haben wir noch $\mathtt{R}^{\mu}_{-1}$ zu definieren. Wir setzen

$$\mathtt{R}^{\mu}_{-1}(x) := -x$$

Man beachte, daß auch wenn wir die Funktion $x \mapsto -x$ iterieren können, der Modus $\mathtt{R}_{-1}$ nicht iterierbar ist, weil dies die Kategorienabbildung verbietet. Dies ist ein Artefakt unseres Beispiels, man kann die Dinge durchaus anders einrichten. Der Modus $\mathtt{T}$ schließlich wird durch folgende Funktionen definiert: $\mathtt{T}^{\varepsilon}(\vec{x}) := \vec{x}$, $\mathtt{T}^{\mu}(x) := x$ und $\mathtt{T}^{\tau}(K) := \mathtt{R}_{-1}(K)$. Als letztes kommen die zweistelligen Modi an die Reihe. Betrachten wir $\mathtt{R}_{\div}$. Es ist klarerweise $\mathtt{R}^{\mu}_{\div}$ die (partielle!) zweistellige Funktion $\div$. Ferner setzen wir

$$\mathtt{R}^{\varepsilon}_{\div}(\vec{x}, \vec{y}) := (\vec{x} \div \vec{y})$$

sowie

$$\mathtt{R}^{\tau}_{\div}(K, L) := \begin{cases} \mathtt{T} & \text{falls } K = L = \mathtt{T} \\ \star & \text{sonst} \end{cases}$$

Die Zeichenkette

$$\mathtt{R}_{\times}\mathtt{R}_{+}\mathtt{Z}_3\mathtt{Z}_5\mathtt{Z}_7$$

definiert, wie man leicht nachrechnet, ein Zeichen, dessen Exponent $((3+5)\times 7)$ ist. Es repräsentiert dagegen $\mathtt{R}_{\div}\mathtt{Z}_2\mathtt{Z}_0$ kein Zeichen. Denn es entspricht der Zeichenkette $(1 \div 0)$. Diese ist zwar auch syntaktisch wohlgeformt, aber die Bedeutung ist nicht definiert, da wir nicht durch 0 teilen dürfen.

Definition 3.1.6 *Ein **lineares Zeichensystem** über dem Alphabet A, der Typenmenge T und den Bedeutungen M ist eine Teilmenge $\Sigma \subseteq A^* \times T \times M$. Sei ferner $\mathtt{S}$ ein Typ. Dann ist die interpretierte Sprache von Σ bezüglich $\mathtt{S}$ definiert durch*

$$\mathtt{S}(\Sigma) := \{\langle \vec{x}, m\rangle : \langle \vec{x}, \mathtt{S}, m\rangle \in \Sigma\}$$

Wir haben hier den Zusatz 'linear' eingeführt, um uns von Zeichensystemen abzugrenzen, welche nicht unbedingt Zeichenketten als Exponenten nehmen. (Zum Beispiel sind Verkehrsschilder oder Piktogramme nichtlineare Zeichensysteme wie schon in Anschnitt 1.3 besprochen.)

Ein Zeichensystem ist also schlicht nur eine Menge von Zeichen. Die Frage ist nun, ob es gelingt, eine Zeichenalgebra zu konstruieren. Dies ist sicher immer möglich. Man nehme einfach für jedes Zeichen einen nullstelligen Modus an. Da dies keinesfalls befriedigend ist, weil wir dann eine unendliche Signatur haben, wollen wir die Bedingungen verschärfen.

Definition 3.1.7 *Es sei $\Sigma \subseteq A^* \times T \times M$ ein lineares Zeichensystem. Wir sagen, Σ sei **kompositional**, falls es eine endliche Signatur Ω gibt und partielle Ω–Algebren*

$\mathfrak{E} = \langle A^*, \{f^{\mathfrak{E}} : f \in F\}\rangle$, $\mathfrak{T} = \langle T, \{f^{\mathfrak{T}} : f \in F\}\rangle$, $\mathfrak{M} = \langle M, \{f^{\mathfrak{M}} : f \in F\}\rangle$ *derart, daß alle Funktionen berechenbar sind und* Σ *genau die Menge der 0–stelligen Zeichen aus* $\mathfrak{E} \times \mathfrak{T} \times \mathfrak{M}$ *ist.* Σ *heiße* ***schwach kompositional****, falls ein kompositionales Zeichensystem* $\Sigma' \subseteq A' \times T' \times M'$ *existiert mit* $\Sigma = \Sigma' \cap A \times T \times M$*.*

Hierbei sind also zwei Bedingungen wesentlich: die Signatur muß endlich sein und die Funktionen auf den Algebren jeweils berechenbar. Wir werden gleich zeigen, daß auch dies keine wesentliche Einschränkung darstellen muß. Dies liegt unter anderem daran, daß man sich mit extra Typen und extra Bedeutungen aushelfen kann.

Wir ziehen zunächst eine leichte Folgerung aus den Definitionen. Ist σ ein Zeichen, so sagen wir $\langle \varepsilon(\sigma), \tau(\sigma), \mu(\sigma)\rangle$ sei die **Realisierung** von σ. Die **Entfaltungsabbildung** sei diejenige Abbildung, welche zu einem Strukturterm t die Realisierung seines Zeichens ausgibt, sofern t definit ist, und ansonsten das Symbol $\star$ ausgibt.

Proposition 3.1.8 *Es sei* $\langle \mathfrak{A}, \varepsilon, \tau, \mu\rangle$ *eine kompositionale Zeichengrammatik. Dann ist die Entfaltungsabbildung berechenbar.*

Zum Beweis beachte man, daß die Entfaltung eines Strukturterms induktiv über den Aufbau berechnet werden kann. Damit gilt folgender Satz.

Korollar 3.1.9 *Es sei* Σ *kompositional. Dann ist* Σ *rekursiv aufzählbar.*

Dies ist insofern bemerkenswert, als die Menge aller Zeichen über E, T und M keineswegs aufzählbar sein muß. Typischerweise enthält M nämlich überabzählbar viele Bedeutungen (die natürlich nicht alle realisiert, das heißt, durch ein Element der Sprache benannt werden können)! Bemerkenswerterweise gilt auch die Umkehrung.

Theorem 3.1.10 *Eine Sprache ist genau dann schwach kompositional, wenn sie rekursiv aufzählbar ist.*

Beweis. Es sei $\Sigma \subseteq E \times T \times M$ gegeben. Falls Σ schwach kompositional ist, so ist es auch rekursiv aufzählbar. Nehmen wir also an, Σ sei rekursiv aufzählbar, etwa $\Sigma = \{\langle e_i, t_i, m_i\rangle : 0 < i \in \omega\}$. (Man beachte, daß der Laufindex mit 1 beginnt.) Nun sei $\mathtt{V}$ ein Symbol und $\Delta := \{\langle \mathtt{V}^n, \mathtt{V}^n, \mathtt{V}^n\rangle : n \in \omega\}$ eine Sprache. Durch geeignete Wahl von $\mathtt{V}$ kann man es erreichen, daß $\Delta \cap \Sigma = \varnothing$ sowie, daß kein $\mathtt{V}^n$ in E, T oder

M auftritt. Es sei $F = \{\mathtt{Z}_0, \mathtt{Z}_1, \mathtt{Z}_2\}$, $\Omega(\mathtt{Z}_0) = 0$, $\Omega(\mathtt{Z}_1) = 1$ und $\Omega(\mathtt{Z}_2 = 1$.

$$\begin{array}{lcl}
\mathtt{Z}_0 & := & \langle \mathtt{V}, \mathtt{V}, \mathtt{V} \rangle \\
 & := & \\
\mathtt{Z}_1(\sigma) & := & \begin{cases} \langle \mathtt{V}^{i+1}, \mathtt{V}^{i+1}, \mathtt{V}^{i+1} \rangle) & \text{falls } \sigma = \langle \mathtt{V}^i, \mathtt{V}^i, \mathtt{V}^i \rangle \\ \star & \text{sonst} \end{cases} \\
 & := & \\
\mathtt{Z}_2(\sigma) & := & \begin{cases} \langle e^i, t^i, m^i \rangle & \text{falls } \sigma = \langle \mathtt{V}^i, \mathtt{V}^i, \mathtt{V}^i \rangle) \\ \star & \text{sonst.} \end{cases}
\end{array}$$

Dies ist wohldefiniert. Ferner sind die Funktionen sämtlich berechenbar. Zum Beispiel ist die Abbildung $\mathtt{V}^i \mapsto e_i$ berechenbar, da sie die Verkettung der berechenbaren Funktionen $\mathtt{V}^i \mapsto i$, $i \mapsto \langle e_i, t_i, m_i \rangle$ soei $\langle e_i, t_i, m_i \rangle \mapsto e_i$ ist. Wir behaupten: die erzeugte Sprache ist genau $\Delta \cup \Sigma$. Dazu überlegt man sich zunächst, daß ein Strukturterm nur dann definit ist, wenn er die folgende Form hat: (a) $t = \mathtt{Z}_1^i \mathtt{Z}_0$, oder (b) $t = \mathtt{Z}_2 \mathtt{Z}_1^i \mathtt{Z}_0$. Im Fall (a) bekommen wir das Zeichen $\langle \mathtt{V}^{i+1}, \mathtt{V}^{i+1}, \mathtt{V}^{i+1} \rangle$, im Falle (b) das Zeichen $\langle e_{i+1}, t_{i+1}, m_{i+1} \rangle$. Also erzeugen wir genau $\Delta \cup \Sigma$. Damit ist Σ schwach kompositional. $\dashv$

Man beachte, daß die Zeichenalgebra sich erstens eines erweiterten Symbolvorrats bedient, der nur dazu dient, sich natürliche Zahlen als Objekte zu beschaffen, mit denen man rechnen kann. Die hier vorgestellte Zeichenalgebra ist sicher sehr unbefriedigend. (Sie ist auch nicht kompositional.) Daher hat man danach gesucht, eine systematischere Theorie von Typen und von Bedeutungen zu gewinnen. Der erste Schritt in diese Richtung waren die Kategorialgrammatiken. Als Vorbereitung werden wir eine Konstruktion für kontextfreie Grammatiken angeben, welche sich erheblich von der von Satz 3.1.10 unterscheidet. Der Ausgangspunkt ist wieder eine interpretierte Sprache $\mathfrak{I} = \{\langle \vec{x}, f(\vec{x}) \rangle : \vec{x} \in S\}$, wo S jetzt kontextfrei ist und f berechenbar. Dann sei $G = \langle \mathtt{S}, N, A, R \rangle$ eine kontextfreie Grammatik mit $L(G) = S$. Setze $A' := A$, $T' := N \cup \{\mathtt{S}^\heartsuit\}$ und $M' := M \cup A^*$. Wir setzen der Einfachheit halber voraus, daß G in Chomsky–Normalform ist. Für jede Regel der Form $\rho : A \to \vec{x}$ nehmen wir einen nullstelligen Modus $\mathtt{R}_\rho$, der wie folgt definiert ist:

$$\mathtt{N}_\rho = \langle \vec{x}, A, \vec{x} \rangle$$

Für jede Regel der Form $\rho : A \to B\ C$ nehmen wir einen zweistelligen Modus $\mathtt{R}_\rho$ definiert durch

$$\mathtt{R}_\rho(\langle \vec{x}, B, \vec{x} \rangle, \langle \vec{y}, C, \vec{y} \rangle) := \langle \vec{x}\,\vec{y}, A, \vec{x}\,\vec{y} \rangle$$

Als letztes wählen wir noch einen einstelligen Modus $\mathtt{S}$:

$$\mathtt{S}(\langle \vec{x}, \mathtt{S}, \vec{x} \rangle) := \langle \vec{x}, \mathtt{S}^\heartsuit, f(\vec{x}) \rangle$$

Dann ist $\mathfrak{I}$ tatsächlich die Menge der Zeichen mit Typ $\mathtt{S}^\heartsuit$. Wie man sieht, ist diese Zeichenalgebra schon wesentlich übersichtlicher. Die Zeichenketten werden lediglich

aneinandergefügt. Die Bedeutungen sind jedoch nach wie vor wenig natürlich. Auch die Typzuweisung ist nicht strukturiert. Die Grammatik ist nicht streng kompositional, weil sie immer noch Nichtstandard–Bedeutungen verwendet. Deswegen noch ein pathologisches Beispiel.

Definition 3.1.11 *Es sei $Z = \langle \mathfrak{A}, \varepsilon, \tau, \mu \rangle$ ein Zeichensystem. Wir sagen, Z* ***hat natürliche Zahlen vom Typ*** *α, falls gilt:*

1. *$\Sigma_\alpha := \{\sigma : \tau(\sigma) = \alpha\}$ ist unendlich.*
2. *$\Sigma_\alpha = \{\langle E_i, \alpha, M_i \rangle : i \in \omega\}$, wobei $E_i \mapsto i$ und $M_i \mapsto i$ bijektive, berechenbare Funktionen sind.*

Definition 3.1.12 *Es sei Δ eine Zeichensprache. Δ heißt* ***modular entscheidbar****, falls gilt.*

1. *Δ ist entscheidbar.*
2. *Für gegebenes $e \in E$ ist entscheidbar, ob ein $\sigma \in \Delta$ existiert mit $\varepsilon(\sigma) = e$.*
3. *Für gegebenes $t \in T$ ist entscheidbar, ob ein $\sigma \in \Delta$ existiert mit $\tau(\sigma) = t$.*
4. *Für gegebenes $m \in M$ ist entscheidbar, ob ein $\sigma \in \Delta$ existiert mit $\mu(\sigma) = m$.*

Proposition 3.1.13 *Es sei Σ kompositional, und sei $\Delta \subseteq \Sigma$ modular entscheidbar. Dann ist auch Δ kompositional.*

Nun gilt folgendes bemerkenswerte Resultat.

Theorem 3.1.14 (Erweiterung) *Es sei Σ eine rekursiv aufzählbare Menge von Zeichen mit endlich vielen Typen. Es sei $\Delta \subseteq \Sigma$ modular entscheidbar, kompositional, und es habe Δ natürliche Zahlen vom Typ α. Dann existiert eine kompositionale Grammatik für Σ.*

Beweis. Nach Annahme hat Δ natürliche Zahlen. Also existiert ein α und berechenbare bijektive Funktionen z und m mit

$$\Sigma_\alpha := \{\sigma : \tau(\sigma) = \alpha\} = \{\langle z^{-1}(j), \alpha, m^{-1}(j) \rangle : j \in \omega\}$$

Ferner ist Σ rekursiv aufzählbar und es existieren nur endlich viele Typen. Dann überlegt man sich leicht, daß für jeden Typ τ ein Paar berechenbare Funktionen z_τ und m_τ existieren mit

$$\Sigma_\tau = \{\langle z_\tau(j), \alpha, m_\tau(j)\rangle : j \in \omega\} .$$

Wir nehmen eine kompositionale Grammatik für T mit Signatur Ω_1. Nach Proposition 3.1.13 dürfen wir dann annehmen, daß die Grammatik nur Zeichen aus Δ erzeugt. (Dies ist insofern nicht banal, als Δ aus einem kleiner Alphabet oder kleinerer Typenmenge etc. definiert sein kann, die Algorithmen zur Berechnung der Funktionen aber auf größeren Gebieten definiert sein könnten.) Dann sei Ω einfach Ω_1 erweitert um einen einstelligen Modus $\mathtt{E}_\tau$ für jeden Typ τ, der wie folgt definiert ist:

$$\mathtt{E}_\tau^\Sigma(\langle E, \alpha, M\rangle) := \langle z_\tau(z(E)), \tau, m_\tau(m(M))\rangle$$

Da sowohl m, z wie auch m_τ und z_τ berechenbar sind, so sind auch $z_\tau \circ z : E \to E$ und $m_\tau \circ m : M \to M$ berechenbar.

Wir müssen also nur noch verifizieren, daß diese Grammatik tatsächlich Σ erzeugt. Sei dazu $\sigma = \langle E, T, M\rangle$ ein Zeichen und t ein Strukturterm von σ. Durch Induktion über die Länge von t beweisen wir, daß $\sigma \in \Sigma$. Hat t die Länge 1, so ist t ein nullstelliger Modus, also in Ω. Dann ist sogar $\sigma \in \Delta$, nach Annahme über Ω. Nun zum Induktionsschritt. (Fall 1.) $t = \mathtt{E}_\tau s$ für ein gewisses s. Nach Induktionsannahme ist s der Strukturterm eines Zeichens $\sigma' = \langle E', T', M'\rangle \in \Sigma$. Es ist $T' = \alpha$ und somit $\sigma' \in \Delta$. Dann ist nach Konstruktion $\sigma \in \Sigma$. (Fall 2.) Das Hauptsymbol von t ist in Ω. Dann ist $\sigma \in \Delta$, weil Modi aus Ω nur Zeichen aus Δ erzeugen. Dies zeigt, daß unsere Grammatik nicht zu viele Zeichen erzeugt. Daß sie auch nicht zu wenige erzeugt, sieht man so. Es sei $\sigma \in \Sigma$. Dann ist $\sigma = \langle E, \tau, M\rangle$ für ein τ. Es gibt dann eine Zahl j derart, daß $\sigma = \langle z_\tau(j), \tau, m_\tau(j)\rangle$. Setze $E' := z^{-1}(j)$, $M' := m^{-1}(j)$. Dann ist $\langle E', \alpha, M'\rangle \in \Delta$ und besitzt einen Strukturterm s. Dann hat Σ den Strukturterm $\mathtt{E}_\tau s$, wie man leicht nachrechnet. $\dashv$

Der Witz daran ist, daß alle beteiligten Zeichen, das heißt, alle, die man erzeugen kann, tatsächlich in Σ liegen, sofern dies für Δ gilt. Was hier allerdings geschieht, ist in höchstem Grade suspekt: aus der Zeichenkette $\vec{x}$ wird die Zeichenkette $f(\vec{x})$, welche absolut nichts mit $\vec{x}$ zu tun haben muß. Daher muß man die Funktionen, welche in der Zeichenkettenalgebra benutzt werden dürfen, erheblich einschränken.

Übung 64. Zeigen Sie Proposition 3.1.13.

Übung 65. Es heiße Δ τ–**eindeutig**, falls gilt:

1. Ist $\langle e, \tau, m\rangle, \langle e', \tau, m\rangle \in \Delta$, so $e = e'$.
2. Ist $\langle e, \tau, m\rangle, \langle e, \tau, m'\rangle \in \Delta$, so $m = m'$.

Es sei $T = \{\tau\}$ und Δ modular entscheidbar unf τ–eindeutig. Zeigen Sie, daß Δ kompositional ist.

Übung 66. Es sei T endlich, $E = A^*$ und es sei Δ τ–eindeutig für jedes $\tau \in T$ und Δ modular entscheidbar. Zeigen Sie: Δ ist kompositional. *Hinweis.* Definieren Sie für jedes τ einen zweistelligen Modus $\mathtt{C}_\tau$. Es sei dabei $\mathtt{C}^\varepsilon_\tau$ die Verkettung und $\mathtt{C}^\tau_\tau(t, t') := t$, falls $t = t' = \tau$, und undefiniert sonst. Das Kunststück ist die Definition von $\mathtt{C}^\mu_\tau$.

Übung 67. Zeigen Sie, daß das Deutsche den Bedingungen des Satzes 3.1.14 genügt. Also ist Deutsch kompositional! *Hinweis.* Es ist ausreichend, eine Teilmenge Δ mit $\Delta = \Delta_\alpha$ zu finden, welche kompositional ist. Daß Deutsch zumindest rekursiv aufzählbar ist, wollen wir hier nicht beweisen, ist aber ziemlich plausibel.

Übung 68. Konstruieren Sie ein Δ, welches den Bedingungen 2. – 4. der modularen Entscheidbarkeit genügt, aber nicht der Bedingung 1. Konstruieren Sie ebenso ein Δ, welches 1. genügt aber nicht 2. oder 3. oder 4.

3.2 Der Syntaktische Typenkalkül

Die Kategorialgrammatik spezifiziert im Gegensatz zu den Phrasenstrukturgrammatiken keinerlei spezielle Regeln, sondern ordnet jedem Element des Lexikons eine (endliche) Menge von Kontextschemata zu. Diese Kontextschemata können entweder über Zeichenketten oder über Strukturbäumen definiert werden. Der zweite Ansatz ist der ältere und führt zu dem sogenannten Ajdukiewicz–Bar Hillel–Kalkül (**AB**–Kalkül), der erste zum Lambek–Kalkül (**L**–Kalkül). Wir stellen erst den **AB**–Kalkül vor.

Wir nehmen an, daß alle Bäume strikt binär verzweigend sind. In diesem Fall hat jeder Knoten, der kein Blatt ist, genau zwei Töchter. Die Phrasenstrukturregel $X \to YZ$ in einer Grammatik codiert einen lokalen Baum als eine Anweisung, die es erlaubt, X durch YZ zu expandieren. Die Kategorialgrammatik interpretiert dies dadurch, daß Y und Z Bäume repräsentieren, die zu einem Baum mit Wurzel X zusammengebaut werden dürfen. Der Ansatz ist also von unten nach oben. Die Tatsache, daß die Bäume in der genannten Art kombiniert werden dürfen, wird durch eine sogenannte **Typenzuweisung** codiert. Dazu müssen wir erst einmal *Typen* definieren. Diese werden mit Hilfe von Operationszeichen über einem Alphabet B von sogenannten **Basistypen** definiert. In unserem Fall sind / und \ die Operationszeichen. (Später wird noch $\bullet$ hinzukommen.) Ist B gegeben, so sei $Typ_{\backslash,/}(B)$ die Menge der Terme über B und den Operationszeichen. Diese werden in Infixnotation geschrieben, also a/b anstelle von $/ab$. Typen bezeichnen wir üblicherweise mit kleinen griechischen Buchstaben, Basistypen durch kleine lateinische Buchstaben. Ist $B = \{a, b, c\}$, so sind $(a/b)\backslash c$, c/a also Typen. Wir vereinbahren wie bei λ–Termen

Linksklammerung. Es ist also $a/b/c/b/a$ kurz für

$$(((a/b)/c)/b)/a$$

Die Interpretation dieser Terme in Bäumen ist wie folgt. Ein *Baum* ist jetzt ein erschöpfend geordneter Baum mit Marken in $Typ_{\backslash,/}(B)$, welcher von einer Konstituentenstrukter herkommt. Dies bedeutet, daß nichtterminale Knoten genau dann binär verzweigen, wenn sie nicht präterminal sind. Ansonsten sind sie unverzweigt. Die Markierungsfunktion t muß korrekt im Sinne der folgende Definition sein.

Definition 3.2.1 *Es sei A ein Alphabet und $\zeta : A \cup \{\varepsilon\} \to \wp(Typ_{\backslash,/}(B))$ eine Funktion, für welche $\zeta(a)$ stets endlich ist. ζ heißt eine* ***Typenzuweisung****. Ein Baum mit Marken in $Typ_{\backslash,/}(B)$ ist* ***korrekt markiert****, falls (1.) für jedes nichtverzweigende x mit Tochter y gilt $t(x) \in \zeta(t(y))$, (2.) für jedes verzweigende x, welches y_0, y_1 unmittelbar dominiert und $y_0 \sqsubset y_1$ gilt: $t(y_0) = t(x)/t(y_1)$ oder $t(y_1) = t(y_0)\backslash t(x)$.*

Definition 3.2.2 *Eine* ***AB–Kategorialgrammatik*** *über A ist ein Quadrupel $K = \langle B, A, \zeta, S\rangle$, wo B eine Menge von Basistypen ist, $S \in B$, und $\zeta : A \to \wp(Typ_{\backslash,/}(B))$ eine Typenzuweisung. Die Menge der von K akzeptierten Bäume, $L_B(K)$ ist die Menge der endlichen, binär verzweigenden, erschöpfend geordneten Bäume mit korrekter Markierungsfunktion $t : B \to Typ_{\backslash,/}(B)$, derart, daß die Wurzel die Marke S trägt.*

Wir weisen darauf hin, daß aus technischen Gründen auch der leeren Zeichenkette ein Typ oder eine Menge von Typen zugeordnet werden muß. Ansonsten ist keine Sprache, welche ε enthält, eine Kategorialsprache. Im Folgenden werden wir diesen Fall ignorieren, aber in den Übungen wird einiges dazu bewiesen werden.

Die Kategorialgrammatik erlaubt also im Wesentlichen nur, die Abbildung ζ festzulegen. Ist diese einmal gegeben, so läßt sich zu jeder Konstituentenstruktur über einer gegebenen Zeichenkette feststellen, welche Strukturbäume über dieser Zeichenkette existieren. Dazu muß man zunächst alle Konstituenten aufzählen. Anschließend wählt man für jeden Präterminalknoten x ein $\alpha \in \zeta(t(y))$, wo $y \prec x$. Die Markierungsfunktion ist damit auf allen übrigen Knoten eindeutig festgelegt. Denn es muß ja in einem lokalen Baum entweder die linke Tochter eine Marke α/β tragen, und dann hat die rechte Tochter α und die Mutter β, oder aber die rechte Tochter trägt $\alpha\backslash\beta$, und dann hat die linke Tochter die Marke α und die Mutter β. Es existiert somit höchstens eine Markierungsfunktion, die die ursprüngliche fortsetzt. Dieser Algorithmus zur Erkennung von Strukturbäumen ist allerdings wenig effektiv. Stattdessen kann man zeigen, daß jede **AB**–Kategorialgrammatik schon eine stark kontextfreie Menge von Bäumen erzeugt. Dies erlaubt uns, sämtliche verfügbaren Sätze über kontextfreie Sprachen anzuwenden.

Theorem 3.2.3 *Es sei $K = \langle B, A, \zeta, S\rangle$ eine **AB**–Kategorialgrammatik. Dann existiert eine kontextfreie Grammatik G mit $L_B(K) = L_B(G)$.*

Beweis. Wir setzen N die Menge aller Teilterme der Terme in $\zeta(a)$, $a \in A$. Es ist ohne Weiteres einzusehen, daß jeder korrekte Baum tatsächlich nur Marken in N trägt. Wir übernehmen das Startsymbol. Die Regeln haben die Form

$$\begin{array}{lcll} \alpha & \to & \alpha/\beta \quad \beta & \\ \alpha & \to & \beta \quad \beta\backslash\alpha & \\ \alpha & \to & a & \alpha \in \zeta(a) \end{array}$$

wobei X, Y alle Symbole von N durchlaufen und a alle Werte in A. Dies definiert $G := \langle S, N, A, R\rangle$. Falls nun $\mathfrak{B} \in L_B(G)$, so ist die Markierung korrekt, wie man leicht nachprüft. Umgekehrt, falls $\mathfrak{B} \in L_B(K)$, so ist jeder lokale Baum eine Instanz einer Regel aus G, die Wurzel trägt die Marke S, und alle Blätter ein terminales a. Also $\mathfrak{B} \in L_B(G)$. $\dashv$

Umgekehrt kann man allerdings auch jede kontextfreie Grammatik in eine **AB**–Grammatik umformen; allerdings ist diese nicht stark, sondern nur schwach äquivalent. Dazu sei S eine kontextfreie Sprache. Es existiert dann eine Grammatik G in Greibach–Form mit $L(G) = S$. Wir unterscheiden zwei Fälle. Fall 1. $\varepsilon \in S$. Wir nehmen dann an, $\mathtt{S}$ stehe niemals aus der rechten Seite einer Produktion. (Dies kann man einrichten unter Wahrung der Greibach–Form; siehe dazu die Übungen.) Dann wählen wir eine Typenzuweisung wie im Fall 2 und addieren $\varepsilon \mapsto \mathtt{S}$. Fall 2. $\varepsilon \in S$. Jetzt definiere

$$\zeta_G(a) := \{X/Y_{n-1}\ldots/Y_1/Y_0 : X \to a \cdot \prod_{i<n} Y_i \in R\}$$

Setze $K := \langle N_G, A, S, \zeta_G\rangle$. Wir behaupten, daß $L(K) = L(G)$. Dazu werden wir zunächst G transformieren, indem wir jede Regel $\rho : X \to a \cdot \prod_{i<n} Y_i$ durch die Regeln

$$Z_0^\rho \to aY_0, \quad Z_1^\rho \to Z_0Y_1, \quad \ldots, \quad Z_{n-1}^\rho \to Y_{n-2}Y_{n-1}$$

Dies definiert die Grammatik H. Es ist $L(H) = L(G)$. Also genügt der Nachweis, daß $L(K) = L(H)$. Statt K können wir auch eine kontextfreie Grammatik F nehmen; die Nichtterminalsymbole seien N_F. Wir zeigen jetzt sogar, daß F und H dieselben Bäume erzeugen modulo der R–Simulation $\sim \subseteq N_H \times N_F$, welche wie folgt definiert ist; (a) Für $X \in N_G$ ist genau dann $X \sim Y$, wenn $X = Y$. (b) $Z_i^\rho \sim W$ genau dann, wenn $W = X/Y_{n-1}/\ldots/Y_{i+1}$ und $\rho = X \to Y_0 \cdot Y_1 \cdot \ldots \cdot Y_{n-1}$ für gewisse Y_j, $i < j < n$. Dazu genügt, daß die Regeln von F mit $\sim$ den Regeln von H entsprechen. Dies rechnet man aber unmittelbar durch Einsetzen aus.

Theorem 3.2.4 (Bar–Hillel & Gaifman & Shamir) *Es sei S eine Sprache über einem Alphabet A. Genu dann S ist durch eine **AB**–Kategorialgrammatik definierbar, wenn S kontextfrei ist.* $\dashv$

Man bemerke, daß wir von den Typenoperationen nur eine, nämlich /, benutzt haben. Der Leser möge sich davon überzeugen, daß auch \ genügt hätte.

Betrachten wir die Kategorialgrammatik nun vom Standpunkt der Zeichenalgebra. Wir führen auf der Algebra der Typen eine partielle 2–stellige Operation $\cdot$ ein, welche folgenden Gleichungen genügt:

$$\alpha/\beta \cdot \beta = \alpha, \qquad \beta \cdot \beta\backslash\alpha = \alpha$$

Es soll $\beta \cdot \gamma$ also nur dann definiert sein, wenn $\gamma = \beta\backslash\alpha$ oder $\beta = \alpha/\gamma$ für ein α. Betrachten wir nun die Konstruktion einer Zeichenalgebra für kontextfreie Sprachen aus dem letzten Abschnitt. Wir können aufgrund der Ergebnisse dieses Abschnitts annehmen, daß die Menge T' jetzt eine Teilmenge von $Typ_{\backslash,/}(B)$ ist, welche lediglich unter $\cdot$ abgeschlossen sein muß. Wir können dann für unsere Regelmodi wie folgt setzen: ist a vom Typ α, so existiert also eine kontextfreie Regel $\rho : \alpha \to a$, und wir führen wir einen nullstelligen Modus $\mathtt{R}_\rho$ mit

$$\mathtt{R}_\rho = \langle a, \alpha, a\rangle$$

Die übrigen Regeln können wir nun zu einem einzigen (!) Modus zusammenfassen:

$$\mathtt{C}(\langle\vec{x}, \alpha, \vec{x}\rangle, \langle\vec{y}, \beta, \vec{y}\rangle) := \langle\vec{x}\,\vec{y}, \alpha \cdot \beta, \vec{x}\,\vec{y}\rangle$$

Allerdings liefert das noch nicht die intendierten Bedeutungen. Wir müßten jetzt also noch $\mathtt{S}^\heartsuit$ wie in Abschnitt 3.1 einführen, um auch dieses zu beheben. Wir wollen dies jedoch nicht tun, sondern uns mit der Frage befassen, wie man auch die Bedeutungen systematischer berechnen kann. Dies wird im allgemeinen nicht möglich sein, denn wir haben ja nur vorausgesetzt, daß f berechenbar ist. Allerdings stellt sich in der Praxis heraus, daß die syntaktischen Typen in einem sehr engen Verhältnis zu ihren Bedeutungen stehen. Dies ist die Philosophie, welche hinter der Montague Semantik steht. Um zu ihr vordringen zu können, müssen wir allerdings einige Hilfsmittel bereitstellen. Dem Leser sei an dieser Stelle empfohlen, den Abschnitt 3.7 zuerst zu lesen, bevor er hier weitergeht.

Sei eine beliebige Menge B von Basistypen gegeben. Aus diesen Typen können wir nun einerseits Typen im Sinne des getypten λ–Kalküls bilden und andererseits syntaktischen Kategorien im Sinne der Kategorialgrammatik. Diese sind nun durch folgenden Homomorphismus miteinander verbunden.

1. $\sigma(b) := b$.
2. $\sigma(\alpha/\beta) := \sigma(\beta) \to \sigma(\alpha)$.
3. $\sigma(\beta\backslash\alpha) := \sigma(\beta) \to \sigma(\alpha)$.

Es sei nun $\langle M, \Lambda, \mathsf{app}\rangle$ ein Mengen–Modell des λ–Kalküls. Eine **Realisierung** von B ist eine Funktion $\ulcorner - \urcorner$, welche jedem $b \in B$ eine Menge zuordnet. Diese Realisierung kann kanonisch auf die semantischen Typen ausgedehnt werden, und damit indirekt auch auf die syntaktischen Kategorien. Damit bekommen wir

$$\begin{array}{lcl} \ulcorner \alpha/\beta \urcorner & := & \ulcorner \beta \urcorner \to \ulcorner \alpha \urcorner \\ \ulcorner \beta\backslash\alpha \urcorner & := & \ulcorner \beta \urcorner \to \ulcorner \alpha \urcorner \end{array}$$

Wir führen dies für unsere arithmetischen Terme vor. Es gibt einen Basistyp $\mathtt{Z}$, und diesen realisieren wir durch die Menge der Zahlen von 0 bis 9. Ferner gibt es den Typ $\mathtt{T}$ der Terme, welchen wir (zum Beispiel) durch die Menge $\mathbb{Q}$ der rationalen Zahlen realisieren. Es ist also

$$\begin{array}{lcl} \ulcorner \mathtt{Z} \urcorner & = & \{0, 1, \ldots, 9\} \\ \ulcorner \mathtt{T} \urcorner & = & \mathbb{Q} \end{array}$$

Nun ist $+ : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$ eine zweistellige Funktion. Diese können wir, wie in Abschnitt 3.7 gezeigt, zu einer Funktion aus $\mathbb{Q} \to (\mathbb{Q} \to \mathbb{Q})$ umdefinieren. Wir schreiben der Einfachheit halber für diese Funktion auch $+$. Der syntaktische Typ, den wir $+$ zuordnen wollen, muß jetzt dazu passen. Wir wählen $(\mathtt{T}\backslash\mathtt{T})/\mathtt{T}$. In der Tat gilt

$$\ulcorner (\mathtt{T}\backslash\mathtt{T})/\mathtt{T} \urcorner = \mathbb{Q} \to (\mathbb{Q} \to \mathbb{Q}) \; ,$$

wie gewünscht. Nun müssen wir es so einrichten, daß die Bedeutung der Zeichenkette $\mathtt{5+7}$ auch wirklich 12 ist. Dazu verlangen wir einfach, daß wenn $\mathtt{+}$ mit $\mathtt{7}$ zu der Konstituente $\mathtt{+7}$ zusammengefügt wird, die Bedeutung von $\mathtt{+}$ (welche ja eine Funktion ist), auf die Zahl 7 angewendet wird. Tun wir dies, so erhalten wir die Funktion $x \mapsto x + 7$ auf $\mathbb{Q}$. Fassen wir anschließend $\mathtt{+7}$ und $\mathtt{5}$ zusammen, so erhalten wir eine Konstituente vom Typ $\mathtt{T}$, deren Bedeutung gerade die Zahl 12 ist.

Falls die Dinge jetzt so eingerichtet sind, können wir uniform für Kategorialgrammatiken zwei zweistellige Modi, $\mathtt{C}_>$ und $\mathtt{C}_<$ definieren.

$$\begin{array}{lcl} \mathtt{C}_>(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x}\,\vec{y}, \alpha \cdot \beta, (XY)\rangle \\ \mathtt{C}_<(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x}\,\vec{y}, \alpha \cdot \beta, (YX)\rangle \end{array}$$

Ferner nehmen wir an, daß wenn $a \in A$ ist und den Typ α hat, es nur endlich viele $X \in \ulcorner \alpha \urcorner$ geben soll, die Bedeutungen von a vom Typ α sind. Für jede dieser Bedeutungen X nehmen wir einen nullstelligen Modus $\langle a, \alpha, X\rangle$. Damit ist die Kategorialgrammatik vollständig standardisiert. Wir haben in den jeweiligen Algebren $\mathfrak{Z}$, $\mathfrak{T}$ und $\mathfrak{M}$ ein je zweistelliges Produkt definiert, welches der Zusammenfügung von zwei Zeichen zu einem Zeichen entspricht. Die Variabilität ist dann nicht mehr in den mehrstelligen Modi zu finden, sondern nur noch in den nullstelligen Modi. Man spricht deshalb auch von Kategorialgrammatik als einer 'lexikalischen' Theorie, weil sämtliche Information über die Sprache im Lexikon liegt.

Definition 3.2.5 *Eine* ***AB–Zeichengrammatik*** *ist eine Zeichengrammatik*

$$\langle \mathfrak{A}, \zeta, \tau, \mu \rangle \ ,$$

bei der die Signatur aus den zwei zweistelligen Modi $\mathtt{C}_>$ *und* $\mathtt{C}_<$ *und endlich vielen nullstelligen Modi* $\mathtt{M}_i$*,* $i \in n$*, besteht, ferner* $\mathfrak{Z} = \langle A^*, \cdot \rangle$*, sowie* $\mathfrak{T} = \langle U, \cdot \rangle$ *eine Unteralgebra der Algebra der Typen über einer Basismenge* B*, und* $\mathfrak{M} = \langle M, \mathsf{app} \rangle$ *ein Modell des Typenkalküls über der Typenmenge* B*. Ferner gilt: ist* $\mathtt{M} = \langle \vec{x}, \alpha, X \rangle$ *ein nullstelliger Modus, so ist* $\sigma(\alpha)$ *der Typ von* X*.*

Üblicherweise nimmt man für $\mathfrak{M}$ ein volles Modell des Typenkalküls über B, obwohl es mehr Funktionen enthält, als man wirklich benötigt. Man beachte, daß die Algebra der Bedeutungen partiell ist und als einzige Operation die Funktionsanwendung besitzt. (Diese ist ja nicht definiert, wenn die Typen nicht zusammenpassen.) Um genau zu sein, hätten wir bei der Definition von $\mathfrak{Z}$, $\mathfrak{T}$ und $\mathfrak{M}$ jeweils auch 0–stellige Funktionen einführen müssen (für $\mathtt{M}_i^\zeta$, $\mathtt{M}_i^\tau$, $\mathtt{M}_i^\mu$), aber so ist die Definition etwas handlicher. Wie wir noch sehen werden, ist die Konzeption einer Kategorialgrammatik zwar einschränkend in Bezug auf die erzeugte Sprache (sie muß ja kontextfrei sein) und in Bezug auf die Kategorialsymbole, jedoch nicht wirklich einschränkend für die Bedeutungen. Dies zu zeigen, überlassen wir dem Leser.

Wir wollen dies anhand eines Beispiels augenfällig machen. Betrachten wir als Alphabet unsere zehn Ziffern. Jeder nichtleeren Zeichenkette über diesem Alphabet enstpricht in eindeutiger Weise eine Zahl, welche wir durch eben diese Folge benennen. Zum Beispiel benennt die Folge `721` die Zahl 721, welche wir in Binärdarstellung auch 101101001 oder LOLLOLOLOOL schreiben. Wir wollen eine Kategorialgrammatik schreiben, welche in kanonischer Weise jeder Ziffernreihe ihre Zahl zuordnet. Dies ist nicht so einfach, wie es auf dem ersten Blick erscheinen mag. Um das Beispiel nicht trivial erscheinen zu lassen, wollen wir eine Grammatik für Binärzahlen schreiben, mit `L` anstelle von 1 und `O` anstelle von 0. Es ist zunächst klar, daß wir einen Typ `Z` wie im obigen Beispiel benötigen. Diesen realisieren wir jetzt durch die Menge der natürlichen Zahlen. Jede Ziffer hat den Typ `Z`. Dann haben wir folgende 0–stellige Modi:

$$\begin{array}{lcl} \mathtt{Z}_0 & := & \langle \mathtt{O}, \mathtt{Z}, 0 \rangle \\ \mathtt{Z}_1 & := & \langle \mathtt{L}, \mathtt{Z}, 1 \rangle \end{array}$$

Nun vereinbahren wir zusätzlich, daß Ziffern auch den Typ `Z\Z` haben. Damit wird die Zahl `101` wie folgt analysiert:

$$\begin{array}{ccc} \mathtt{L} & \mathtt{O} & \mathtt{L} \\ \mathtt{Z} & \mathtt{Z\backslash Z} & \mathtt{Z\backslash Z} \\ \hline & \mathtt{Z} & \\ \hline & \mathtt{Z} & \end{array}$$

Dies bedeutet, daß sie semantisch gesehen auch Funktionen von ω nach ω sind. Wie man sich leicht überlegt, sind sie folglich die Funktionen $\lambda n.(2n + k)$. Hierbei muß

k gerade der Wert der Ziffer sein. Wir bekommen also noch nullstellige Modi der folgenden Form:

$$\mathtt{M}_0 := \langle \mathtt{0}, \mathtt{Z}\backslash\mathtt{Z}, \lambda n.2n\rangle$$

Dies tut das Gewünschte. Allerdings besitzt diese Grammatik nicht die ideale Form, denn wir haben jetzt jeder Ziffer zwei Bedeutungen gegeben, die nichts miteinander zu tun haben müssen. Wir hätten ja zum Beispiel einen Modus der folgenden Art einführen können:

$$\mathtt{M}_0 := \langle \mathtt{0}, \mathtt{Z}\backslash\mathtt{Z}, \lambda n.(2n+1)\rangle$$

Wir können dies vermeiden, indem wir ein zweites Kategoriesymbol einführen, `T`, welches für eine Ziffernfolge steht, während `Z` nur für Ziffern stehen soll. Wir definieren anstelle von $\mathtt{M}_0$ jetzt einen einstelligen Modus $\mathtt{N}_0$ und einen zweistelligen Modus $\mathtt{N}_1$:

$$\begin{array}{lcl} \mathtt{N}_0 & := & \langle \varepsilon, \mathtt{T}/\mathtt{Z}, \lambda n.n\rangle \\ \mathtt{N}_1 & := & \langle \varepsilon, \mathtt{T}/\mathtt{Z}/\mathtt{T}, \lambda m.\lambda n.(2m+n)\rangle \end{array}$$

Zum Beispiel erhalten wir `LOL` als Exponenten des Terms

$$\mathtt{N}_1\mathtt{N}_1\mathtt{N}_0\mathtt{Z}_1\mathtt{Z}_0\mathtt{Z}_1$$

Der Wert dieses Terms errechnet sich folgendermaßen:

$$\begin{array}{ll} (\mathtt{N}_1\mathtt{N}_1\mathtt{N}_0\mathtt{Z}_1\mathtt{Z}_0\mathtt{Z}_1)^\mu & = \mathtt{N}_1^\mu(\mathtt{N}_1^\mu(\mathtt{N}_0(\mathtt{Z}_1^\mu))(\mathtt{Z}_0^\mu))(\mathtt{Z}_1^\mu) \\ & = \mathtt{N}_1^\mu(\mathtt{N}_1^\mu((\mathtt{N}_0^\mu(1))(0))(1)) \\ & = \mathtt{N}_1^\mu(\mathtt{N}_1^\mu(\lambda n.n(1))(0))(1) \\ & = \mathtt{N}_1^\mu(\mathtt{N}_1^\mu(1)(0))(1) \\ & = \mathtt{N}_1^\mu((\lambda m.\lambda n.2m+n)(1)(0))(1) \\ & = \mathtt{N}_1^\mu(2)(1) \\ & = (\lambda m.\lambda n.2m+n)(2)(1) \\ & = 5 \end{array}$$

Diese Lösung ist weitaus eleganter als die erste. Trotzdem ist das Ergebnis nicht befriedigend. Wir mußten einige Modi postulieren, welche man nicht auf der Zeichenkette sehen kann. Am einfachsten wäre es, man könnte einen zweistelligen Modus definieren, dessen Bedeutung die Funktion g ist. Dies ist in der Kategorialgrammatik dieser Form jedoch nicht erlaubt. Dies ist ein Mangel, den wir in dem nächsten Kapitel beheben wollen. Ein weiteres Problem ist die eingeschränkte Funktionalität im Bereich der Zeichenketten. Am Beispiel der Grammatik T des vorigen Abschnitts mag dies verdeutlicht werden. Wir haben vereinbahrt, daß jeder Term durch Klammern eingeschlossen wird. Die Klammern sind nun Symbole des Alphabets, die lediglich Lesehilfen sind und keinerlei Bedeutung tragen. Um die Klammern korrekt zu setzen, muß man einigen Aufwand treiben. Wir schlagen etwa folgende Grammatik

vor.

$$\begin{array}{lcl}
\mathtt{O}_1 & := & \langle +, (\mathtt{T}\backslash\mathtt{U})/\mathtt{T}, \lambda m.\lambda n.n + m\rangle \\
\mathtt{O}_2 & := & \langle -, (\mathtt{T}\backslash\mathtt{U})/\mathtt{T}, \lambda m.\lambda n.n - m\rangle \\
\mathtt{O}_3 & := & \langle \div, (\mathtt{T}\backslash\mathtt{U})/\mathtt{T}, \lambda m.\lambda n.n/m\rangle \\
\mathtt{O}_4 & := & \langle \times, (\mathtt{T}\backslash\mathtt{U})/\mathtt{T}, \lambda m.\lambda n.nm\rangle \\
\mathtt{O}_5 & := & \langle -, \mathtt{U}/\mathtt{T}, \lambda n. - n\rangle \\
\mathtt{O}_6 & := & \langle \mathtt{)}, \mathtt{L}/\mathtt{U}, \lambda n.n\rangle \\
\mathtt{O}_7 & := & \langle \mathtt{(}, \mathtt{L}\backslash\mathtt{T}, \lambda n.n\rangle \\
\mathtt{Z}_0 & := & \langle \mathtt{L}, \mathtt{T}, 0\rangle \\
\mathtt{Z}_1 & := & \langle \mathtt{O}, \mathtt{T}, 1\rangle
\end{array}$$

Die Vorstellung ist die, daß ein Operationszeichen zunächst einmal einen ungeklammerten Term erzeugt, der dann eine rechte Klammern und anschließend eine linke Klammer benötigt, um zu einem Term zu werden. Eine dieser Zuweisung entsprechende Semantik muß den Typen nun Bedeutungen zuweisen. Dafür nehmen wir jeweils immer $\mathbb{Q}$ als einzigen Basistyp. Die Klammern haben als Bedeutung lediglich die Identität. Falls wir also eine Klammer hinzufügen, so ändern wir den Wert des Terms nicht. Dies ist ein gangbarer Weg, allerdings wird die Anzahl der primitiven Typen vergrößert, ohne daß dem ein erweiterter semantischer Bereich gegenübersteht. Auch hier wirkt sich also die Standardisierung nicht hilfreich aus.

Nun ist die Anwendung einer Funktion auf ein Argument nicht die einzig mögliche Kompositionsregel. Es wurde deswegen insbesondere von Peter Geach in [16] vorgeschlagen, noch weitere Regeln zuzulassen. Dieser Gedanke wurde einerseits durch den Lambek–Kalkül realisiert, auf den wir noch später eingehen werden, und andererseits durch die sogenannte Kombinatorische Kategorialgrammatik. Bei der letzteren ist der Grundgedanke dieser. Jedem Modus in der Kategorialgrammatik enstpricht ein schematischer getypter Kombinator. Nämlich $\mathtt{C}_<$ entspricht der Kombinator **U** (definiert in Anschnitt 3.7) und $\mathtt{C}_>$ der Kombinator **I**. Dies ist — aus der Sicht der kombinatorischen Logik — eine willkürliche Einschränkung. Sehen wir uns nun Beispiele an, wie dies geändert werden kann. Wir nehmen zusätzlich zu den eben genannten Kombinatoren noch an, wir hätten den Kombinator

$$\mathbf{B} := (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\lambda \mathtt{z}.\mathtt{x}(\mathtt{yz}))))$$

Dann ist, wie man leicht sieht, $\mathbf{B}XY$ nichts anderes als die Verkettung der Funktionen X und Y. Denn offensichtlich muß, wenn $\mathtt{z}$ den Typ γ hat, $\mathtt{y}$ den Typ $\beta \to \gamma$ haben und $\mathtt{x}$ den Typ $\alpha \to \beta$. Dann ist aber $\mathbf{Bxy} \equiv (\lambda \mathtt{z}.(\mathtt{x}(\mathtt{yz})))$ vom Typ $\alpha \to \gamma$. Nun definieren wir ein neues Typprodukt $\circ$ durch

$$\begin{array}{lcl}
\gamma/\beta \circ \beta/\alpha & := & \gamma/\alpha \\
\beta/\alpha \circ \beta\backslash\gamma & := & \gamma/\alpha \\
\gamma/\beta \circ \alpha\backslash\beta & := & \alpha\backslash\gamma \\
\alpha\backslash\beta \circ \beta\backslash\gamma & := & \alpha\backslash\gamma
\end{array}$$

Ferner definieren wir zwei neue Modi, $\mathtt{B}_>$ und $\mathtt{B}_<$ wie folgt:

$$\begin{array}{lcl} \mathtt{B}_>(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x} \cdot \vec{y}, \alpha \circ \beta, \mathbf{B}XY\rangle \\ \mathtt{B}_<(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x} \cdot \vec{y}, \alpha \circ \beta, \mathbf{B}YX\rangle \end{array}$$

Anstelle von $\mathbf{B}XY$ hätten wir auch $\mathbf{V}XY$ setzen können, wobei $\mathbf{V}$ definiert ist durch

$$\mathbf{V} := (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\lambda \mathtt{z}.\mathtt{y}(\mathtt{xz}))))$$

Wir bezeichnen diese Erweiterung des **AB**–Kalküls mit CCG(**B**). Diese erlaubt uns also, nicht nur die Kompositionsmodi $\mathtt{C}_>$ und $\mathtt{C}_<$ sondern auch die Modi $\mathtt{B}_>$ und $\mathtt{B}_<$ zu verwenden. Die entstehenden Baummengen sind allerdings von gänzlich neuer Art. Denn jetzt ist es so, daß, falls x ein verzweigender Knoten mit Töchtern y_0 und y_1 ist, auch x den Typ α/γ tragen darf, wenn y_0 den Typ α/β und y_1 den Typ β/γ trägt. Hinter der Definition des Produkts, $\circ$, steckt eine gewisse Willkür. Was man aus Sicht der semantischen Typen erwarten sollte, ist, daß der Typ von $\sigma(\alpha \circ \beta)$ gleich $\eta \to \theta$ ist, wenn $\sigma(\alpha) = \zeta \to \theta$ und $\sigma(\beta) = \eta \to \zeta$ für gewisse η, ζ und θ. Ansonsten sollte das syntaktische Produkt undefiniert sein. Stattdessen wird das syntaktische Produkt semantisch gesehen symmetrisiert, dafür aber die Richtungen spezifiziert. Dies geht im Einzelnen so. Es wird bei der Anwendung eine Kategorie (hier ζ) gekürzt. In der Kategorie η/θ wir die Selektionsrichtung (hier: rechts) als zu θ gehörig aufgefaßt. Wird θ nicht gekürzt, so muß in der Ergebniskategorie wieder $/\theta$ stehen. Wird θ gekürzt, so muß η/θ links stehen. Dies ergibt die obenstehenden Produktregeln. Wir überlassen es dem Leser, für CCG(**B**) nachzuweisen, daß die erzeugten Baummengen auch kontextfrei sind. Insofern ist also nicht viel gewonnen. Deswegen wollen wir im Detail eine etwas andere Erweiterung studieren, nämlich CCG(**P**). Hierbei ist

$$\mathbf{P} := (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\lambda \mathtt{z}.(\lambda \mathtt{u}.\mathtt{x}(\mathtt{yzu})))))$$

Damit dieser richtig getypt ist, darf man den Typ von `z` und `u` frei wählen, etwa β und γ. Dann ist `y` vom Typ $\gamma \to (\beta \to \delta)$ für ein δ, und vom Typ $\delta \to \alpha$ für ein gewisses α. `y` steht also für eine mindestens zweistellige Funktionen, `x` für eine mindestens einstellige. Ist der Kombinator definiert, so liegt der Modus fest, sofern wir noch eine syntaktische Kombinationsregel festlegen. Dazu definieren wir folgendes Produkt.

$$\begin{array}{llcll} \alpha/\delta & \bullet & (\delta/\beta)/\gamma & := & (\alpha/\beta)/\gamma \\ (\delta/\beta)/\gamma & \bullet & \delta\backslash\alpha & := & (\alpha/\beta)/\gamma \\ \alpha/\delta & \bullet & (\beta\backslash\delta)/\gamma & := & (\beta\backslash\alpha)/\gamma \\ (\beta\backslash\delta)/\gamma & \bullet & \delta\backslash\alpha & := & (\beta\backslash\alpha)/\gamma \\ \alpha/\delta & \bullet & \gamma\backslash(\delta/\beta) & := & \gamma\backslash(\alpha/\beta) \\ \gamma\backslash(\delta/\beta) & \bullet & \delta\backslash\alpha & := & \gamma\backslash(\alpha/\beta) \\ \alpha/\delta & \bullet & \gamma\backslash(\beta\backslash\delta) & := & \gamma\backslash(\beta\backslash\alpha) \\ \gamma\backslash(\beta\backslash\delta) & \bullet & \delta\backslash\alpha & := & \gamma\backslash(\beta\backslash\alpha) \end{array}$$

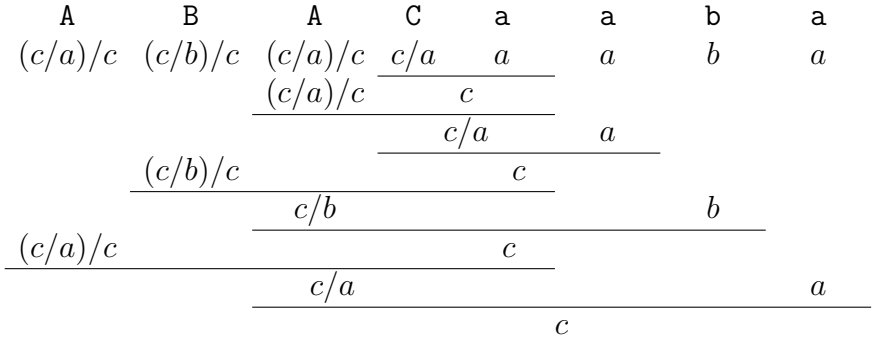
Jetzt definieren wir zwei neue Modi:

$$\begin{array}{lcl} \mathtt{P}_{>}(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x} \cdot \vec{y}, \alpha \bullet \beta, \mathbf{P}XY\rangle \\ \mathtt{P}_{<}(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) & := & \langle \vec{x} \cdot \vec{y}, \alpha \bullet \beta, \mathbf{P}XY\rangle \end{array}$$

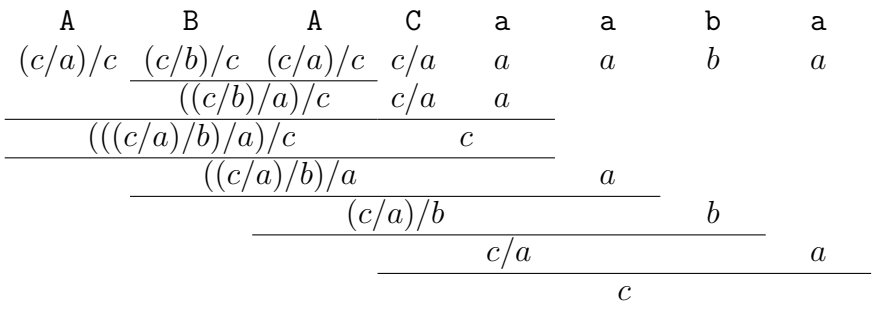
Wir werden diese Grammatik näher studieren. Als Beispiel diene folgende Grammatik.

$$\begin{array}{lcl} \mathtt{M}_0 & = & \langle \mathtt{A}, (c/a)/c, (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\mathtt{x+y})))\rangle \\ \mathtt{M}_1 & = & \langle \mathtt{B}, (c/b)/c, (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\mathtt{x} \cdot \mathtt{y})))\rangle \\ \mathtt{M}_2 & = & \langle \mathtt{a}, a, 1\rangle \\ \mathtt{M}_3 & = & \langle \mathtt{b}, b, 2\rangle \\ \mathtt{M}_4 & = & \langle \mathtt{C}, c/a, (\lambda \mathtt{x}.\mathtt{x})\rangle \end{array}$$

Nehmen wir die Zeichenkette `ABACaaba`. Diese besitzt zum Beispiel folgende Analyse:



Wenn wir gleichzeitig die Bedeutung ausrechnen, so bekommen wir 5. In dieser Analyse wurde nur $\mathtt{C}_{>}$ verwendet. Nun existiert aber auch folgende Analyse:



Auch hier ergibt sich im Übrigen die Bedeutung 5. Diese Analyse benutzt den Modus $\mathtt{P}_{>}$. Man beachte, daß sich im Verlaufe der Ableitung die Kategorien aufblähen.

Theorem 3.2.6 *Es existieren CCG(**P**)–Grammatiken, welche nicht kontextfreie Baummengen erzeugen.*

Wir werden zeigen, daß unsere eben betrachtete Grammatik von dieser Art ist. Dazu einige Beobachtungen.

Lemma 3.2.7 *Es sei $\alpha = \eta_1/\eta_2/\eta_3$, $\beta = \eta_3/\eta_4/\eta_5$ und $\gamma = \eta_5/\eta_6/\eta_7$. Dann ist*

$$\alpha \bullet (\beta \bullet \gamma) = (\alpha \bullet \beta) \bullet \gamma$$

Beweis. Beweis durch direktes Nachrechnen. Es ist etwa $\alpha \bullet \beta = \eta_1/\eta_2/\eta_3/\eta_4/\eta_5$. $\dashv$

Es ist also $\bullet$ assoziativ, sofern es definiert ist (im Gegensatz zu $\cdot$). Betrachten wir eine Zeichenkette der Form $\vec{x}\mathtt{Ca}\vec{y}$, wo $\vec{x} \in (\mathtt{A} \cup \mathtt{B})^*$, $\vec{y} \in (\mathtt{a} \cup \mathtt{b})^*$ und $h(\vec{x}) = \vec{y}^T$, wobei $h : \mathtt{A} \mapsto \mathtt{a}, \mathtt{B} \mapsto \mathtt{b}$. Zum Beispiel ist $\mathtt{AABACaaaba}$ eine solche Zeichenkette. Dann gilt: mit Ausnahme von $\vec{x}\mathtt{C}$ sind alle Präfixe Konstituenten. Denn Präfixe von $\vec{x}$ sind Konstituenten, wie man leicht sieht. Falls dies so ist, so folgt leicht, daß die Baummengen nicht kontextfrei sind. Denn ist $\vec{x} \neq \vec{y}$, so ist $\vec{x}\mathtt{Ca}h(\vec{y}^T)$ nicht ableitbar. Dagegen ist $\vec{x}\mathtt{Ca}h(\vec{x}^T)$ ableitbar. Wäre nun die Baummenge kontextfrei, kann es aber nicht unendlich viele solcher $\vec{x}$ geben, ein Widerspruch.

Somit haben wir schon die kontextfreien Grammatiken hinter uns gelassen. Wir können jedoch die Situation noch verschärfen. Sei $\mathfrak{N}$ die folgende Grammatik.

$$\begin{array}{lcl}
\mathtt{N}_0 & = & \langle \mathtt{A}, c\backslash(c/a), (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\mathtt{x+y}))) \rangle \\
\mathtt{N}_1 & = & \langle \mathtt{B}, c\backslash(c/b), (\lambda \mathtt{x}.(\lambda \mathtt{y}.(\mathtt{x} \cdot \mathtt{y}))) \rangle \\
\mathtt{N}_2 & = & \langle \mathtt{a}, a, 1 \rangle \\
\mathtt{N}_3 & = & \langle \mathtt{b}, b, 2 \rangle \\
\mathtt{N}_4 & = & \langle \mathtt{C}, c, (\lambda \mathtt{x}.\mathtt{x}) \rangle
\end{array}$$

Theorem 3.2.8 *$\mathfrak{N}$ erzeugt eine nicht kontextfreie Sprache.*

Beweis. Sei S die von $\mathfrak{N}$ erzeugte Sprache. Setze $L := \mathtt{C}(\mathtt{A} \cup \mathtt{B})^*(\mathtt{a} \cup \mathtt{b})^*$. Falls S kontextfrei ist, so ist es auch $S \cap L$ (nach Satz 2.1.14). Definiere h durch $h(\mathtt{A}) := h(\mathtt{a}) := \mathtt{a}$, $h(\mathtt{B}) := h(\mathtt{b}) := \mathtt{b}$ sowie $h(\mathtt{C}) := \varepsilon$. Wir zeigen:

$(*)$ $\quad \vec{x} \in S \cap L$ genau dann, wenn $\vec{x} \in L$ und $h(\vec{x}) = \vec{y}\vec{y}$ für ein $\vec{y} \in (\mathtt{a} \cup \mathtt{b})^*$.

Also $h[S \cap L] = \{\vec{y}\vec{y} : \vec{y} \in (\mathtt{a} \cup \mathtt{b})^*\}$. Letztere ist aber nicht kontextfrei. Daraus folgt dann mit Satz 1.5.8, daß $S \cap L$ nicht kontextfrei ist, also auch S nicht kontextfrei. Nun zum Beweis von $(*)$. Es bezeichne für $\Delta = \langle \delta_i : i < n \rangle$ c/Δ die Kategorie $c/\delta_0/\delta_1/\ldots/\delta_{n-1}$. Dann gilt:

$$c\backslash(c/\Delta_1) \bullet c\backslash(c/\Delta_2) = c\backslash(c/\Delta_2;\Delta_1)$$

Sei also $\mathtt{C}\vec{x}\vec{y}$ derart, daß $\vec{x} \in (\mathtt{A} \cup \mathtt{B})^*$ und $\vec{y} \in (\mathtt{a} \cup \mathtt{b})^*$. Es ist nicht schwer zu sehen, daß dann $\mathtt{C}\vec{x}$ eine Konstituente sein muß. Nun sei $x = x_0x_1\ldots x_{n-1}$. Ferner sei $d_i = a$ falls $x_i = \mathtt{A}$ und $d_i = b$, falls $x_i = \mathtt{B}$, $i < n$. Dann ist die Kategorie von $\vec{x}$ gleich $c\backslash(c/\Delta)$, mit $\Delta = \langle d_i : i < n \rangle$. Folglich ist $\mathtt{C}\vec{x}$ eine Konstituente vom Typ c/Δ. Dies

bedeutet aber, daß y_0 den Typ d_0, y_1 den Typ d_1 usw haben muß. Aber falls y_i den Typ d_i hat, so ist $h(x_i) = y_i$, wie man leicht nachprüft. Dies ergibt, daß $h(\vec{x}) = \vec{y}$ sein muß. Falls dies umgekehrt so ist, so ist die Zeichenkette ableitbar. $\dashv$

Wir haben also eine Grammatik, welche nicht kontextfreie Sprachen erzeugt. Kombinatorische Kategorialgrammatiken sind also stärker als Kategorialgrammatiken.

Es gibt noch einen anderen Weg, CCGs einzuführen. Dazu erweitern wir nicht die Kombinationsregeln, sondern führen einige neue nullstellige Modi ein:

$$\begin{array}{lcl} \mathtt{B}_0 & := & \langle \varepsilon, \gamma/\alpha/(\gamma/\beta)/(\beta/\alpha), \mathbf{B}\rangle \\ \mathtt{B}_1 & := & \langle \varepsilon, (\alpha\backslash\gamma)/(\gamma/\beta)/(\alpha\backslash\beta), \mathbf{B}\rangle \\ \mathtt{B}_2 & := & \langle \varepsilon, \gamma/\alpha/(\beta\backslash\gamma)/(\beta/\alpha), \mathbf{V}\rangle \\ \mathtt{B}_3 & := & \langle \varepsilon, (\alpha\backslash\gamma)/(\beta\backslash\gamma)/(\alpha\backslash\gamma), \mathbf{V}\rangle \end{array}$$

Hierbei handelt es sich allerdings gar nicht um vier, sondern um unendlich viele Modi. Denn je nach Instantiierung von α, β und γ muß der Kombinator, **B** oder **V**, den Typ $(\alpha \to \beta) \to ((\beta \to \gamma) \to (\alpha \to \gamma))$ haben. Deswegen ist es überhaupt nur möglich, daß solche Grammatiken nicht kontextfreie Sprachen erzeugen können. Man nennt lexikalische Elemente, welche eine parametrische Schar von Typen (mit entsprechender parametrischer Bedeutung) haben, **polymorph**. Ein besonders wichtiges Beispiel polymorpher Elemente sind `und` und `nicht`. Sie haben syntaktisch gesehen den Typ $(\alpha\backslash\alpha)/\alpha$ und α/α, wo α jeden Typ annehmen kann. Dies bedeutet, daß je zwei Konstituenten gleicher Kategorie durch `und` zu einer Konstituente gleicher Kategorie verbunden werden können, und jede Konstituente durch `nicht` zu einer Konstituente gleicher Kategorie wird.

Übung 69. Es sei $\zeta : A_\varepsilon \to \wp(\mathit{Typ}_{\backslash,/}(B))$ eine Typenzuweisung. Zeigen Sie, daß die korrekt markierten Bäume eine kontextfreie Sprache definieren.

Übung 70. Zeigen Sie, daß zu jeder kontextfreien Sprache eine Grammatik in Greibach–Form existiert, in der `S` niemals auf der rechten Seite einer Produktion auftritt.

Übung 71. Es sei $\zeta : A_\varepsilon \to \wp(\mathit{Typ}_{\backslash,/}(B))$ eine Typenzuweisung. Sei ferner `S` der ausgezeichnete Basistyp. ζ' heißt **normal**, falls $\zeta(\varepsilon) = \mathtt{S}$, und daß kein $\zeta(a)$ ein α enthält, welches die Form $\gamma/\beta_0/\ldots/\beta_{n-1}$ mit $\beta_i = \mathtt{S}$ für ein $i < n$ hat. Zeigen Sie, daß es zu ζ ein normales ζ' gibt, derart, ζ' und ζ die gleiche Sprache haben.

Übung 72. Sei $S \subseteq A^*$ kontextfrei und $f : A^* \to M$. Schreiben Sie eine **AB**–Zeichengrammatik, deren interpretierte Sprache die Menge aller $\langle \vec{x}, f(\vec{x})\rangle$ ist.

Übung 73. Es sei $\langle \mathfrak{A}, \zeta, \tau, \mu\rangle$ eine **AB**–Zeichenkettengrammatik. Zeigen Sie, daß für alle Zeichen $\langle \vec{x}, \alpha, X\rangle$ gilt: X hat den Typ $\sigma(\alpha)$. *Hinweis.* Induktion über die Länge des Strukturterms.

Übung 74. Zeigen Sie, daß die CCG(**B**) Grammatiken stets kontextfreie Zeichen-

kettensprachen erzeugen, sogar kontextfreie Baummengen. *Hinweis.* Zeigen Sie: ist A eine beliebige endliche Menge von Typen, so kann man mit $\mathbf{B}$ höchstens $|A|^n$ viele Kategorien erzeugen.

3.3 Der AB–Kalkül

Wir wollen nun einen Kalkül vorstellen, mit dem die Ableitbarkeitsbeziehungen in der **AB**–Kategorialgrammatik axiomatisch erfaßt werden. Man beachte, daß das einzig variable Element die elementare Typenzuweisung ist. Wir wählen nun ein Alphabet A und eine elementare Typenzuweisung ζ. Wir schreiben $[\alpha]_\zeta$ für die Menge der binären Konstituentenstrukturen über A, für die eine korrekte Markierung existiert, sodaß die Wurzel den Typ α hat. Da Kategorialgrammatiken invertierbar sind, existiert in jedem Fall höchstens eine Markierungsfunktion (die terminalen Regeln ausgenommen). Nun führen wir ein Symbol $\circ$ ein. Mit $\circ$ kann man binär verzweigende Konstituentenstrukturen erzeugen. Es seien $\langle X, \mathfrak{X}\rangle$, $\langle Y, \mathfrak{Y}\rangle$ Konstituentenstrukturen und $X \cap Y = \varnothing$. So ist

$$\langle X, \mathfrak{X}\rangle \circ \langle Y, \mathfrak{Y}\rangle := \langle X \cup Y, \mathfrak{X} \cup \mathfrak{Y} \cup \{X \cup Y\}\rangle$$

(Im Prinzip ist $\circ$ auch dann wohldefiniert, wenn die einzelnen Konstituentenstrukturen nicht binär verzweigen.) Im Falle, wo $X \cap Y \neq \varnothing$ ist, muß man zur disjunkten Summe übergehen. Wir wollen die Einzelheiten nicht ausbuchstabieren. Mit Hilfe von $\circ$ bilden wir nun auch Terme über A, d. h. wir betrachten wieder die Termalgebra über A mit der Signatur $\circ$. Jedem Term wird induktiv eine Konstituentenstruktur wie folgt zugeordnet.

$$\begin{array}{rcl} a^k & := & \langle\{a\}, \{\{a\}\}\rangle \\ (s \circ t)^k & := & s^k \circ t^k \end{array}$$

Man beachte, daß $\circ$ in zwei Bedeutungen gebraucht wurde. Nun wollen wir als letzten Schritt $[\alpha]_\zeta$ betrachten. Diese entsprechen genau den binären Konstituentenstrukturen über A. Es gelten nun folgende Beziehungen

$$\begin{array}{rcl} [\alpha/\beta]_\zeta \circ [\beta]_\zeta & \subseteq & [\alpha]_\zeta \\ [\beta]_\zeta \circ [\beta\backslash\alpha]_\zeta & \subseteq & [\alpha]_\zeta \end{array}$$

Wir abstrahieren nun von A und ζ. Anstatt $\circ$ als Konstruktor für Konstituentenstrukturen über A aufzufassen, interpretieren wir $\circ$ als Konstruktor von Konstituentenstrukturen über $Typ_{\backslash,/}(B)$ für ein gegebenes B. Wir nennen einen Term über τ mit Hilfe von $\circ$ aufgebaut einen **Strukturterm**. Typen werden mit kleinen griechischen Buchstaben, Strukturen mit großen griechischen Buchstaben bezeichnet. Induktiv erweitern wir die Interpretation $[-]_\zeta$ auf Strukturen wie folgt.

$$[\Gamma \circ \Delta]_\zeta := [\Gamma]_\zeta \circ [\Delta]_\zeta$$

Wir führen nun ein neues Symbol ein, $\vdash$. Dies ist eine Relation zwischen Strukturen und Typen. Falls Γ eine Struktur und α ein Typ ist, so bezeichnet $\Gamma \vdash \alpha$ die Tatsache, daß für jede Interpretation in einem Alphabet A mit Typenzuweisung ζ gilt $[\Gamma]_\zeta \subseteq [\alpha]_\zeta$. Die so erhaltene Interpretation werden wir die **Kürzungsinterpretation** nennen. Hierbei werden die Typen als konkrete Marken eingesetzt, welche Knoten zugewiesen werden, und welche der Kürzungsregel genügen. Wir wollen noch eine andere Interpretation vorstellen, welche wir die **Kontextinterpretation** nennen wollen. Hierbei ist die Vorstellung die, daß wir jedem Basistyp b eine Menge $[\![b]\!]$ von Konstituentenstrukturen zuordnen. Diese Abbildung wird wie folgt fortgesetzt.

$$\begin{array}{lcl} [\![\alpha/\beta]\!] & := & [\![\alpha]\!]//[\![\beta]\!] \\ [\![\beta\backslash\alpha]\!] & := & [\![\beta]\!]\backslash\backslash[\![\alpha]\!] \\ [\![\Gamma \circ \Delta]\!] & := & [\![\Gamma]\!] \circ [\![\Delta]\!] \end{array}$$

Hierbei ist

$$\begin{array}{lcl} X//Y & := & \{\Gamma : (\forall \Delta \in Y)(\Gamma \circ \Delta \in X)\} \\ Y\backslash\backslash X & := & \{\Gamma : (\forall \Delta \in Y)(\Delta \circ \Gamma \in X)\} \end{array}$$

Man vergleiche auch die Übung 1.2 aus Abschnitt 1.2. Es gilt demnach

$$[\![\alpha]\!] \subseteq [\![\beta/\alpha]\!]\backslash\backslash[\![\beta]\!] \qquad \Leftrightarrow \qquad [\![\beta/\alpha]\!] \circ [\![\alpha]\!] \subseteq [\![\beta]\!]$$

Diese Interpretationen sind verschieden. In der Kontextinterpretation ist folgendes Gesetz gültig, wie sich aus dem eben Gasagten unschwer ableiten läßt.

$$[\![\alpha]\!] \subseteq [\![(\beta/\alpha)\backslash\beta]\!]$$

(Dieses Gesetz ist auch als *Typenanhebung* bekannt, da es erlaubt, von dem Typ α zu dem "angehobenen" Typ $(\beta/\alpha)\backslash\beta$ überzugehen.) Auf der anderen Seite ist eine Konstituente mit Marke α keine Konstituente mit Marke $(\beta/\alpha)\backslash\beta$, und so gilt dieses Gesetz nicht für die Kürzungsinterpretation.

Wir werden nun einen Kalkül vorstellen, mit welchem $\vdash$ axiomatisiert wird und zwar für beide Interpretationen. Diese Kalküle definieren, wann eine sogenannte *Sequenz* ableitbar ist. Eine **Sequenz** ist ein Paar $\Gamma \vdash \alpha$, wo Γ eine Stuktur ist und α ein Typ. Dieser besteht aus einem einzigen Axiom, nämlich $\alpha \vdash \alpha$, sowie einer Reihe von Regeln, mit Hilfe derer aus schon abgeleiteten Tatsachen neue abgeleitet werden können. Bei der Formulierung der Regeln benützen wir die Schreibweise $\Gamma[\alpha]$. Dies bedeutet in diesem Zusammenhang, daß Γ eine Struktur ist, und α in Γ vorkommt. Es wird dabei ein Vorkommen von α fixiert; wenn wir also anschließend $\Gamma[\Delta]$ schreiben, so meinen wir das Resultat der Ersetzung des vorher fixierten Vorkommens von α durch Δ.

$$\text{(ax)} \quad \alpha \vdash \alpha \qquad\qquad \text{(schn)} \quad \frac{\Gamma \vdash \alpha \qquad \Delta[\alpha] \vdash \beta}{\Delta[\Gamma] \vdash \beta}$$

$$\textbf{(I}\text{–}/\textbf{)} \quad \frac{\Gamma \circ \alpha \vdash \beta}{\Gamma \vdash \beta/\alpha} \qquad\qquad \textbf{(I}\text{–}\backslash\textbf{)} \quad \frac{\alpha \circ \Gamma \vdash \beta}{\Gamma \vdash \alpha\backslash\beta}$$

$$\textbf{(}/\text{–}\textbf{I)} \quad \frac{\Gamma \vdash \alpha \qquad \Delta[\beta] \vdash \gamma}{\Delta[\beta/\alpha \circ \Gamma] \vdash \gamma} \qquad\qquad \textbf{(}\backslash\text{–}\textbf{I)} \quad \frac{\Gamma \vdash \alpha \qquad \Delta[\beta] \vdash \gamma}{\Delta[\Gamma \circ \alpha\backslash\beta] \vdash \gamma}$$

Sequenzen, welche Axiome sind, heißen auch **Anfansgsequenzen**. Die Regel (schn) wird auch die **Schnittregel** genannt. Wir bezeichnen den obenstehenden Kalkül mit **AB** + (schn), und mit **AB** den Kalkül ohne (schn). Ferner heißt der Kalkül aus den Regeln **(\–I)** und **(/–I)** $\mathbf{AB}^-$. Eine Sequenz $\Gamma \vdash \alpha$ ist nun in einem Kalkül $\mathcal{S}$ **ableitbar** — wir sagen dann auch, sie sei $\mathcal{S}$–ableitbar —, wenn sie entweder ein Axiom ist, oder aus ableitbaren Sequenzen mit Anwendung einer Regel des Kalküls hervorgeht.

Proposition 3.3.1 (Korrektheit) *(1.) Falls $\Gamma \vdash \alpha$ im Kalkül $\boldsymbol{AB}^-$ ableitbar ist, so gilt $[\Gamma]_\zeta \subseteq [\alpha]_\zeta$ für jedes A und jede Typenzuweisung ζ. (2.) Falls $\Gamma \vdash \alpha$ in $\boldsymbol{AB}$ herleitbar ist, so gilt $[\![\Gamma]\!] \subseteq [\![\alpha]\!]$ für jede Kontextinterpetation.*

Eine wichtige Eigenschaft dieser Kalküle ist ihre Entscheidbarkeit. Gegeben Γ und α läßt sich in endlicher Zeit entscheiden, ob $\Gamma \vdash \alpha$ ist oder nicht. Dies ist nun allerdings nicht unmittelbar klar. Dazu eine Vorüberlegung. Ist die Schnittregel keine Regel des Kalküls, so ist die Entscheidbarkeit des Kalküls, in diesem Falle **AB**, unmittelbar einleuchtend. Die letzten vier Regeln sind nämlich so beschaffen, daß die Anzahl der Symbole stets zunimmt. Man kann also die Länge einer Ableitung nach oben durch die Anzahl der in Γ und α auftauchenden Symbole abschätzen. Die Schnittregel stellt also das einzige Problem dar. Wir wollen nun zeigen, daß die Schnittregel entbehrlich ist; das heißt zu jeder Ableitung von eine Sequenz $\Gamma \vdash \alpha$, welche (schn) verwendet, existiert eine Ableitung von $\Gamma \vdash \alpha$, welche (schn) nicht verwendet.

Definition 3.3.2 *Eine Regel R heißt **zulässig** für einen Ableitungskalkül $\mathcal{S}$, falls für jede $\mathcal{S} + R$–Ableitung einer Sequenz $\Gamma \vdash \alpha$ eine $\mathcal{S}$–Ableitung von $\Gamma \vdash \alpha$ existiert.*

Theorem 3.3.3 (Schnittelimination) *Es existiert ein Verfahren, zu jedem Beweis von $\Gamma \vdash \alpha$ in $\boldsymbol{AB}$ + (schn) einen Beweis von $\Gamma \vdash \alpha$ in $\boldsymbol{AB}$ zu konstruieren. Also ist (schn) für $\boldsymbol{AB}$ zulässig.*

Korollar 3.3.4 $\boldsymbol{AB}$ **+ (schn)** *ist entscheidbar.* $\dashv$

Um das Theorem zu beweisen, definieren wir zunächst den *Grad* eines Schnittes. Für eine Struktur Γ sei $d(\Gamma)$ die Anzahl der Vorkommen von $\backslash$ und $/$. Ferner sei in (schn) der Grad des Schnittes genau $d(\Gamma) + d(\alpha) + d(\Delta) + d(\gamma)$. Wir zeigen nun folgendes Lemma, aus welchem der Schnitteliminationssatz folgt.

Lemma 3.3.5 *Zu jedem Beweis von $\Gamma \vdash \alpha$, welcher genau n Schnitte vom Grade g enthält und p Anwendungen von Schnitt vom Grade $< g$, existiert ein Beweis von $\Gamma \vdash \alpha$, welcher höchstens $n-1$ Anwendungen von (schn) vom Grade g enthält, und höchstens $p+2$ Anwendungen von Schnitt vom Grad $< g$.*

Beweis. Es sei R eine Regel. In einer Anwendung einer Regel werden die Strukturvariablen zu Strukturtermen instantiiert. Wir nennen die darin vokommenden Typen *Seitenformeln*, alle anderen *Hauptformeln*. Es sei nun eine Anwendung von (schn) gegeben. Falls sie die höchste Anwendung einer Regel ist, so sind die Prämissen dieser Regel schon Axiome. In diesem Falle ist die Anwendung von der Form

$$\frac{\alpha \vdash \alpha \qquad \alpha \vdash \alpha}{\alpha \vdash \alpha}$$

und kann durch $\alpha \vdash \alpha$ ersetzt werden. Analoges gilt für den Fall, daß nur eine der Prämissen ein Axiom ist. Wir können also jetzt annehmen, daß beide Prämissen des Schnittes durch Anwendung von Regeln gewonnen wurden. Wir zeigen nun, wie man im Falle eines Schnittes auf Hauptformeln vorgeht und überlassen dem Leser den (einfacheren) Fall, daß der Schnitt über eine Seitenformel der linken Prämissen erfolgt. Wir wenden uns also gleich dem Fall zu, daß der Schnitt über eine Hauptformel der linken Prämissen erfolgt. Der erste Fall ist, daß die Schnittformel durch (**I**–/) eingeführt worden ist.

$$\frac{\dfrac{\Gamma \circ \alpha \vdash \beta}{\Gamma \vdash \beta/\alpha} \qquad \Delta[\beta/\alpha] \vdash \gamma}{\Delta[\Gamma] \vdash \gamma}$$

Betrachten wir nun die Regelanwendungen, welche unmittelber über $\Delta[\beta/\alpha] \vdash \gamma$ steht. Es gibt nun mehrere Möglichkeiten für die rechte Prämisse. (0.) Die Prämisse ist ein Axiom. Dann ist $\gamma = \beta/\alpha$, und der Schnitt entbehrlich. (1.) β/α ist Hauptformel der rechten Prämisse. Dann ist $\Delta[\beta/\alpha] = \Theta[\beta/\alpha \circ \Xi]$ für gewisse Θ und Ξ, und die Anwendung der Regel sah wie folgt aus

$$\frac{\Xi \vdash \alpha \qquad \Theta[\beta] \vdash \gamma}{\Theta[\beta/\alpha \circ \Xi] \vdash \gamma}$$

Dann können wir die Ableitung wie folgt umstrukturieren.

$$\frac{\dfrac{\Gamma \circ \alpha \vdash \beta \qquad \Theta[\beta] \vdash \gamma}{\Theta[\Gamma \circ \alpha] \vdash \gamma} \qquad \Xi \vdash \alpha}{\Theta[\Gamma \circ \Xi] \vdash \gamma}$$

Wir haben $d(\Delta) = d(\Xi) + d(\Theta)$. Der Schnittgrad des ersten Schnittes ist nun

$$\begin{array}{rl} & d(\Gamma \circ \alpha) + d(\beta) + d(\Theta) + d(\gamma) \\ = & d(\Gamma) + d(\beta) + d(\alpha) + d(\Theta) + d(\gamma) \\ < & d(\Gamma) + d(\beta/\alpha) + d(\Delta) + d(\gamma)\ . \end{array}$$

Der Schnittgrad des zweiten Schnittes ist

$$\begin{array}{rl} & d(\Theta) + d(\Gamma) + d(\alpha) + d(\Xi) + d(\gamma) \\ \leq & d(\Delta) + d(\Gamma) + d(\alpha) + d(\gamma) \\ < & d(\Delta) + d(\Gamma) + d(\beta/\alpha) + d(\gamma)\ . \end{array}$$

Der Schnitt ist also durch zwei Schnitte von kleinerem Grad ersetzt worden. Man beachte, daß sich die Anzahl der übrigen Schnitte nicht verändert hat. Nun nehmen wir also an, die Schnittformel ist nicht Hauptformel der rechten Prämisse. (2.) $\gamma = \zeta/\varepsilon$, und die Prämisse ist durch Anwenden von (**I**–/) entstanden.

$$\frac{\Delta[\beta/\alpha] \circ \varepsilon \vdash \zeta}{\Delta[\beta/\alpha] \vdash \zeta/\varepsilon}$$

Dann ersetzen wir diesen Beweis durch

$$\dfrac{\dfrac{\Gamma \circ \alpha \vdash \beta}{\Gamma \vdash \beta/\alpha} \qquad \Delta[\beta/\alpha] \circ \varepsilon \vdash \zeta}{\dfrac{\Delta[\Gamma] \circ \varepsilon \vdash \zeta}{\Delta[\Gamma] \vdash \varepsilon/\zeta}}$$

Der Grad des Schnittes ist jetzt

$$\begin{array}{rl} & d(\Gamma) + d(\beta/\alpha) + d(\Delta) + d(\varepsilon) + d(\zeta) \\ < & d(\Gamma) + d(\beta/\alpha) + d(\Delta) + d(\gamma) \end{array}$$

(3.) $\gamma = \varepsilon\backslash\zeta$ und ist durch Anwendung der Regel **(I**–**)** enstanden. Dann verfahre man analog wie in (2.). (4.) Die Regelanwendung führt eine Formel ein, welche in Δ vorkommt. Dieser Fall sei dem Leser überlassen.

Falls nun die linke Prämisse durch Anwendung von **(**\–**I)** enstanden ist, so verfähre man ganz analog. Nun nehmen wir an, die linke Prämisse sei wie durch Anwendung von **I**–**I**) entstanden.

$$\dfrac{\dfrac{\Gamma \vdash \alpha \quad \Delta[\beta] \vdash \gamma}{\Delta[\beta/\alpha \circ \Gamma] \vdash \gamma} \quad \Theta[\gamma] \vdash \delta}{\Theta[\Delta[\beta/\alpha \circ \Gamma]] \vdash \delta}$$

Dann können wir diesen Beweis wie folgt umstrukturieren.

$$\dfrac{\Gamma \vdash \alpha \quad \dfrac{\Delta[\beta] \vdash \gamma \quad \Theta[\gamma] \vdash \delta}{\Theta[\Delta[\beta]] \vdash \delta}}{\Theta[\Delta[\beta/\alpha \circ \Gamma]] \vdash \delta}$$

Auch hier rechnet man leicht nach, daß der Grad des neuen Schnittes kleiner ist als der Grad des alten Schnittes. Der Fall, wo die linke Prämisse durch Anwendung von **(**\–**I)** entstanden ist, ist ganz analog. Damit sind alle Fälle behandelt worden. ⊣

Der Schnitteliminationssatz wird nun wie folgt bewiesen. Jedem Beweis B ordnen wir zwei Maßzahlen zu, $a(B)$ und $s(B)$. Dabei ist $a(B)$ die Anzahl der Schnitte in B und $s(B)$ die Summe aller 3^g, wo g der Schnittgrad eines Schnittes im Beweis ist. Dabei werden alle Schnitte gezählt, also kann der Summand 3^g durchaus mehrfach

auftreten. Immer ist $a(B) \leq s(B)$. Solange $a(B) < s(B)$, existiert ein Schnitt mit Schnittgrad $\neq 0$, und das obenstehende Lemma kann angewendet werden. Mit jeder Anwendung der Konstruktion im vorigen Lemma wird $s(B)$ kleiner. Denn ein Summand 3^g wird ersetzt durch zwei Summanden $\leq 3^{g-1}$. Das Verfahren endet also in endlicher Zeit, und wir haben dann $a(B) = s(B)$. Falls $a(B) \neq 0$, so haben wir noch Schnitte vom Grade 0. In diesem Fall ist aber $d(\Gamma) = d(\alpha) = d(\Delta) = d(\gamma) = 0$, also sind α, γ Basistypen, und Γ und Δ Strukturen aus Basistypen. Wir betrachten einen Schnitt, dessen Prämissen ohne (schn) hergeleitet worden sind. Da alle anderen Regeln den Grad der Sequenz echt erhöhen, sind die Prämissen bereits Axiome. Dies bedeutet aber, daß $\Gamma = \alpha = \gamma = \Delta = b$ für ein gewisses $b \in B$. Die Konklusion ist dann von der Form $b \vdash b$. Dies bedeutet, wir können auf diese Anwendung von (schn) verzichten. Mit dieser Operation verringert sich $a(B)$ und $s(B)$ um eins. Auch dieses Verfahren kann nur endlich oft angewendet werden und hört auf, wenn $a(B) = 0$. In diesem Fall haben wir einen Beweis ohne (schn), wie gewünscht.

Der **AB**–Kalkül liefert also eine Methode, Strukturterme auf ihren syntaktischen Typ zu testen. Wir erwarten nun, daß die Bedeutungen der Terme systematisch mit ihrem syntaktischen Aufbau zusammenhängen, daß wir also auch sagen können, welche Bedeutung vom Typ α der Term Γ hat. Den syntaktischen Kategorien werden semantische Typen mittels σ wie in Abschnitt 3.2 zugeordnet. Betrachten wir nun die Regeln unseres **AB**–Kalküls. Wir beginnen mit den einfachsten Regeln. Wir schreiben $\vec{x} : \alpha : X$, um zu sagen, daß $\vec{x}$ eine Zeichenkette vom Typ α mit Bedeutung X ist, und nennen dies wie gehabt ein Zeichen. Falls $\vec{x}$ unerheblich ist, so lassen wir es weg. Zunächst einmal wollen wir die Zeichenketten weglassen, da es Probleme gibt, auf die wir noch zurückkommen werden. Für die Bedeutungen schreiben wir λ–Terme, welche jedoch nur Bezeichnungen für die "wirklichen" Bedeutungen sein sollen. Deswegen schreiben wir jetzt $(\lambda x.xy)$ anstelle von `(λx.xy)`. Letzteres ist die Zeichenkette, welche die Funktion $(\lambda x.xy)$ bezeichnet. Zu diesem Unterschied, welcher uns hier nicht interessieren muß, siehe Abschnitt 4.7. Eine **Struktur** ist ein Term aus Zeichen, aufgebaut mit Hilfe von $\circ$. Sequenzen sind Paare $\Gamma \vdash \zeta$, wo Γ eine Struktur und ζ ein Zeichen ist. Dann gilt

$$\alpha : x_{\sigma(\alpha)} \vdash \alpha : x_{\sigma(\alpha)}$$

Hierbei ist $x_{\sigma(\alpha)}$ eine beliebige Variable vom Typ α. Um die Notation nicht unnötig zu überfrachten, werden wir jetzt x_α anstelle von $x_{\sigma(\alpha)}$ schreiben. Dies bedeutet, daß im Typ–Index einer Variable der Unterschied zwischen $/$, $\backslash$ und $\rightarrow$ nur notationell ist. Als erstes behandeln wir Schnitt.

$$\text{(schn)} \qquad \frac{\Gamma \vdash \alpha : Y \qquad \Delta[\alpha : x_\alpha] \vdash \beta : X}{\Delta[\Gamma] \vdash \beta : [Y/x_\alpha]X}$$

Dies ist so zu interpretieren. Ist $\Delta[\alpha]$ eine Struktur mit markiertem Vorkommen von α, so sei f diejenige Funktion, welche Bedeutung von Δ ist und in der x_α auftritt.

Diese Variable (welche nur einmal vorkommt), wird nun durch g ersetzt. Der Schnitt entspricht also gerade der Substitution. Die anderen Regeln des Kalküls sind etwas komplizierter.

$$(/\text{–}\mathbf{I}) \qquad \frac{\Gamma \vdash \alpha : X \qquad \Delta[\beta : x_\beta] \vdash \gamma : Y}{\Delta[\beta/\alpha : x_{\alpha\to\beta} \circ \Gamma] \vdash \gamma : [(x_{\alpha\to\beta}X)/x_\beta]Y}$$

Dies entspricht dem Ersetzen einer primitiven Konstituente durch eine komplexe Konstituente oder auch dem Ersetzen eines Werts $X(x)$ durch das Paar $\langle X, x\rangle$. Dabei wird eine Variable $x_{\alpha\to\beta}$ eingeführt, welche für eine beliebige Funktion von α–Bedeutungen nach β–Bedeutungen steht. Die Variable x_β ist dagegen verschwunden. Dies ist ein schwerwiegender Mangel dieses Kalküls (der dagegen andere Vorzüge hat). Wir werden weiter unten einen anderen entwickeln. Analog die Regel (\–**I**). Nun zu den Regeln (**I**–/) und (**I**–\). Diese sind in folgender Weise semantisch interpretierbar. Angenommen, Γ sei eine Konstituente vom Typ α/β. Dann haben wir ganz einfach:

$$(\mathbf{I}\text{–}/) \qquad \frac{\Gamma \circ \alpha : x_\alpha \vdash \beta : X}{\Gamma \vdash \beta/\alpha : (\lambda x_\alpha.X)}$$

Hierbei ist die Bedeutung von Γ also von der Form X und die Bedeutung von α gleich Y. Man beachte, daß uns auf diese Weise der **AB**–Kalkül dazu zwingt, die Bedeutung eines Wortes vom Typ α/β als Funktion von α–Bedeutungen nach β–Bedeutungen anzusetzen. Denn es steht ja förmlich geschrieben, daß Γ die Bedeutung einer Funktion haben muß. Wir nennen die Regeln (**I**–/) und (**I**–\) auch **Abstraktionsregeln**.

Falls wir bei unserer Interpretation im λ–Kalkül bleiben, muß die Regel jedoch eingeschränkt werden. Dazu ein Beispiel. Man kann in der Regel (\–**I**) $\Delta = \beta$ setzen, und $\gamma = \beta$.

$$\frac{\alpha : x_\alpha \vdash \alpha : x_\alpha \qquad \beta : x_\beta \vdash \beta : x_\beta}{\alpha/\beta : x_{\beta\to\alpha} \circ \beta : x_\beta \vdash \alpha : (x_{\beta\to\alpha}x_\beta)}$$

Daraus bekommen wir durch Anwenden von (**I**–/)

$$\frac{\alpha/\beta : x_{\beta\to\alpha} \circ \beta : x_\beta \vdash \alpha : (x_{\beta\to\alpha}x_\beta)}{\alpha/\beta : x_{\beta\to\alpha} \vdash \alpha/\beta : (\lambda x_\beta.(x_{\beta\to\alpha}x_\beta))}$$

Nun ist $\lambda x_\beta.x_{\beta\to\alpha}x_\beta$ das Redex von $x_{\beta\to\alpha}$, und deswegen sind beide dieselben Funktionen. Durch Anwenden von (**I**–\) bekommen wir aber andererseits

$$\frac{\alpha/\beta : x_{\beta\to\alpha} \circ \beta : x_\beta \vdash \alpha : (x_{\beta\to\alpha}x_\beta)}{\beta : x_\beta \vdash (\alpha/\beta)\backslash\alpha : (\lambda x_{\beta\to\alpha}.(x_{\beta\to\alpha}x_\beta))}$$

Dies ist die Typanhebung, von der wir oben gesprochen hatten. Ist x_β vom Typ β, so können wir es auch als Funktion ansehen, welches zu jeder Funktion f geeigneten Typs das Element $f(x_\beta)$ ausgibt. Allerdings ist x_β nicht dieselbe Funktion wie

$(\lambda x_{\beta\to\alpha}.(x_{\beta\to\alpha}x_\beta))$. Deswegen ist die Anwendung der Regel in diesem Fall semantisch inkorrekt. Um den Fehler zu beheben, müssen wir verlangen, daß die Variable, welche abstrahiert wird, auf der linken Seite von $\vdash$ in der Prämisse als Argumentvariable auftritt und nicht als Funktionsvariable. Also nehmen wir die folgenden Regeln an:

$$(\mathbf{I}\text{–}/) \qquad \frac{\Gamma \circ \alpha : x_\alpha \vdash \beta : X}{\Gamma \vdash \beta/\alpha : (\lambda x_\alpha.X)}, \qquad \text{falls } x_\alpha \text{ Argumentvariable ist}$$

Wie kann man nun feststellen, wann x_α Argumentvariable ist? Dazu verlangen wir, daß in dem Kalkül $\mathbf{AB}^-$ die Sequenz $\Gamma \vdash \beta/\alpha$ herleitbar ist. Dies ist beim ersten Hinsehen paradox: denn dadurch wird der Kalkül genauso schwach wie $\mathbf{AB}^-$. Wozu sollte man die Regel (**I**–/) einsetzen, wenn man sowieso schon die Sequenz hergeleitet hat? Dazu muß man den Unterschied zwischen dem uninterpretierten Kalkül und dem interpretierten beachten. Wir erlauben im interpretierten Kalkül die Verwendung der (interpretierten) Regel (**I**–/), falls im uninterpretierten Kalkül $\Gamma \vdash \beta/\alpha$ herleitbar ist; oder, prosaischer, wenn Γ die Kategorie β/α und damit den Typ $\alpha \to \beta$ hat. Daß dies den Kalkül verstärkt, kann man so sehen. Im interpretierten $\mathbf{AB}^-$–Kalkül ist die folgende Sequenz nicht ableitbar.

$$\beta/\alpha : x_{\alpha\to\beta} \vdash \beta/\alpha : \lambda x_\alpha.x_{\beta\to\alpha}x_\alpha$$

Der Nachweis ist als Übung überlassen. Jedoch haben wir gerade diese Sequenz im interpretierten **AB**–Kalkül abgeleitet.

Wir wollen sehen, warum die Beschränkung die Sache behebt. Dazu erst einmal ein Rekurs auf die Interpretation von Sequenzen. Wir weisen noch einmal darauf hin, daß in einer Ableitung eine Anfangssequenz stets nur einmal vorkommen darf. Also bindet der λ–Operator stets nur ein Vorkommen einer freien Variablen. Wir definieren jetzt eine Abbildung von Strukturtermen nach λ–Termen wie folgt.

$$\begin{array}{lcl} h(\alpha : f) & := & f \\ h(\Gamma \circ \Delta) & := & \begin{cases} h(\Gamma)h(\Delta) & \text{falls dies wohlgeformt ist} \\ h(\Delta)h(\Gamma) & \text{falls dies wohlgeformt ist} \\ \text{undefiniert} & \text{sonst} \end{cases} \end{array}$$

Theorem 3.3.6 *Es sei $\Gamma \vdash \alpha : X$ im **AB**–Kalkül ableitbar. Dann ist $h(\Gamma)$ definiert hat den Typ $\sigma(\alpha)$, und $h(\Gamma) \equiv X$. Ferner tritt jede freie Variable von Γ auch frei in X auf, sofern keine Anfangssequenz zweimal auftritt.*

Der Beweis erfolgt durch Induktion über die Ableitung der Sequenz. Man macht dabei wesentlich von der Tatsache Gebrauch, daß eine Variable jeweils nur einmal frei auftritt. Dazu weiter unten ein Beispiel.

Der so definierte Kalkül ist allerdings unschön. Er zwingt uns, im Verlaufe der Ableitung neue Variablen einzuführen. Der folgende Kalkül, **N**, vermeidet dies.

$$\text{(ax)} \quad \alpha : x_\alpha \vdash \alpha : x_\alpha$$

$$(\mathbf{I}\text{–}/) \quad \frac{\Gamma \circ \alpha : x_\alpha \vdash \beta : X}{\Gamma \vdash \beta/\alpha : (\lambda x_\alpha.X)} \qquad (\mathbf{I}\text{–}\backslash) \quad \frac{\alpha : x_\alpha \circ \Gamma \vdash \beta : X}{\Gamma \vdash \alpha\backslash\beta : (\lambda x_\alpha.X)}$$

$$(\mathbf{E}\text{–}/) \quad \frac{\Gamma \vdash \alpha : X \qquad \Delta \vdash \beta/\alpha : Y}{\Delta \circ \Gamma \vdash \beta : (XY)} \qquad (\mathbf{E}\text{–}\backslash) \quad \frac{\Gamma \vdash \alpha : X \qquad \Delta \vdash \alpha\backslash\beta : Y}{\Gamma \circ \Delta \vdash \beta : (XY)}$$

Dieser Kalkül unterscheidet sich von dem ersten dadurch, daß jetzt von oben nach unten Formeln auch abgebaut werden (wie in den Regeln (**E**–/) und (**E**–\)). Auch in diesem Kalkül ist Schnitt zulässig. Dies nachzuweisen, ist als Übung überlassen. Auch hier muß man im Übrigen annehmen, daß eine Anfangssequenz nur einmal auftritt, sodaß also eine gegebene Variable nicht mehr als einmal in einer Anfansgssequenz auftreten kann. Ansonsten liefert der Kalkül falsche Resultate in der Interpretation. Zum Beispiel ist die folgende Sequenz ableitbar.

$$\beta/\alpha/\alpha : X \circ \alpha : x_\alpha \vdash \beta/\alpha : \lambda x_\alpha.((Xx_\alpha)x_\alpha)$$

Der Typ des λ–Terms auf der rechten Seite ist β, weil X vom Typ $\alpha \to (\alpha \to \beta)$ ist. Aber der syntaktische Typ ist β/α, und $\sigma(\beta/\alpha) = \alpha \to \beta \neq \beta$. Grob gesprochen entspricht eine solche Sequenz dem Fall eines syntaktischen Arguments, welches semantisch gesehen leer ist, weil ein anderes bereits seine semantische Funktion übernommen hat.

Gehen wir nun zu den Zeichenketten über. Wir lassen dabei die Interpretation weg, weil diese bereits in den obenstehenden Regeln erklärt ist. Unsere Objekte werden jetzt also notiert als $\vec{x} : \alpha$, wo $\vec{x}$ eine Zeichenkette, α ein syntaktischer Typ ist. Der Leser möge sich an dieser Stelle erinnern, daß $\vec{y}/\vec{x}$ diejenige Zeichenkette ist, welche entsteht, wenn von $\vec{y}$ das Postfix $\vec{x}$ weggenommen wird. Dies ist also nur dann definiert, wenn $\vec{y} = \vec{u} \cdot \vec{x}$ für ein $\vec{u}$, und ist dann gleich $\vec{u}$. Analog für $\vec{x}\backslash\vec{y}$.

$$\text{(ax)} \quad \vec{x} : \alpha \vdash \vec{x} : \alpha$$

$$((\mathbf{I}\text{–}/) \quad \frac{\Gamma \circ \vec{x} : \alpha \vdash \vec{y} : \beta}{\Gamma \vdash \vec{y}/\vec{x} : \beta/\alpha} \qquad ((\mathbf{I}\text{–}\backslash) \quad \frac{\vec{x} : \alpha \circ \Gamma \vdash \vec{y} : \beta}{\Gamma \vdash \vec{x}\backslash\vec{y} : \alpha\backslash\beta}$$

$$(\mathbf{E}\text{–}/) \quad \frac{\Gamma \vdash \vec{x} : \alpha \qquad \Delta \vdash \vec{y} : \beta/\alpha}{\Delta \circ \Gamma \vdash \vec{y} \cdot \vec{x} : \beta} \qquad (\mathbf{E}\text{–}\backslash) \quad \frac{\Gamma \vdash \vec{x} : \alpha \qquad \Delta \vdash \vec{y} : \alpha\backslash\beta}{\Gamma \circ \Delta \vdash \vec{x} \cdot \vec{y} : \beta}$$

Die Schnittregel ist allerdings nun gar nicht formulierbar. Sie müßte nämlich etwa wie folgt aussehen.

$$\text{(schn)} \quad \frac{\Gamma \vdash \vec{x} : \alpha \qquad \Delta[\vec{y} : \alpha] \vdash \vec{z} : \beta}{\Delta[\Gamma] \vdash [\vec{y}/\vec{x}]\vec{z} : \beta}$$

Hierbei bedeutet $[\vec{y}/\vec{x}]\vec{z}$ das Resultat der Ersetzung von $\vec{x}$ durch $\vec{y}$ in $\vec{z}$. Dabei darf allerdings nur ein einzelnes Vorkommen von $\vec{y}$ ersetzt werden, und dieses ist nicht

eindeutig bestimmt. Die Operation auf der rechten Seite ist also nicht eindeutig bestimmt, da die Zeichenkette im Gegensatz zum Strukturterm zu wenig Information hergibt. Folglich ist die Regel so nicht formulierbar. Man wird nicht umhinkommen, auch Variable für Zeichenketten einzuführen, um dann die Zeichenketten als λ–Terme auffassen zu können, die man analog den eigentlichen Bedeutungen manipuliert. Auf der anderen Seite ist Schnitt ohnehin zulässig, sodaß man auf diese Regel auch verzichten kann.

Es heiße nun **NZ** der volle Zeichenkalkül. Dieser besitzt also die Regeln (**I**–/), (**I**–\) mit den Beschränkungen, daß nur Argumentvariable abstrahiert werden dürfen, sowie die Regeln (**E**–/) und (**E**–\). Es gibt keine Schnittregel. Aus einer **AB**–Zeichengrammatik $\mathfrak{A}$ läßt sich nun ein **NZ**–Kalkül wie folgt definieren. Anstelle der Anfangssequenzen, die in **NZ** zulässig sind, lassen wir nur die folgenden Anfangssequenzen zu:

$(\mathrm{ax}_{\mathfrak{A}})$ $\quad \langle \vec{x}, \alpha, X\rangle \vdash \langle \vec{x}, \alpha, X\rangle, \quad$ wo $\langle \vec{x}, \alpha, X\rangle$ nullstelliger Modus von $\mathfrak{A}$

Üblicherweise besitzt eine **AB**–Grammatik keine Modi der Form $\langle \vec{x}, \alpha, x_\alpha\rangle$, wo x_α eine Variable ist. Zum Beispiel ist dies bei natürlichen Sprachen der Fall. Man überlegt sich leicht, daß in diesem Fall die Regeln (**I**–/) und (**I**–\) niemals zum Zuge kommen, da ja x_α eine Variable sein muß, welche in $\mathfrak{A}$ allerdings nicht bereitsteht. Es ist dann ein leichtes nachzuweisen, daß der modifizierte **NZ**–Kalkül und die Grammatik $\mathfrak{A}$ äquivalent sind. Beim Lambek–Kalkül, den wir im nächsten Abschnitt besprechen, sieht die Sache allerdings anders aus.

Übung 75. Beweisen Sie den Korrektheitssatz, Proposition 3.3.1.

Übung 76. Es sei p wie folgt definiert.

$$\begin{array}{lcl} p(\alpha) & := & \alpha \\ p(\Gamma \circ \Delta) & := & p(\Gamma) \cdot p(\Delta) \end{array}$$

Zeigen Sie: genau dann ist $\Gamma \vdash \alpha$ in $\mathbf{AB}^-$ ableitbar, wenn $p(\Gamma) = \alpha$. Zeigen Sie, daß die gleiche Behauptung auch für $\mathbf{AB}^- + (\mathrm{schn})$ gilt. Folgern Sie daraus, daß die Schnittregel für $\mathbf{AB}^-$ zulässig ist. (Dies kann man im Prinzip auch aus dem Beweis für **AB** herausfiltern, aber der Beweis hier ist denkbar einfach.)

Übung 77. Beweisen Sie die Vollständigkeit des Kalküls **AB**. Das bedeutet, falls für alle A und alle ζ, $[\Gamma]_\zeta \subseteq [\alpha]_\zeta$, so ist $\Gamma \vdash \alpha$ in **AB** ableitbar.

Übung 78. Zeigen Sie, daß jede kontextfreie Sprache durch eine Kategorialgrammatik über den Basistypen $B = \{s, t\}$ erzeugt werden kann.

Übung 79. Zeigen Sie: im interpretierten $\mathbf{AB}^-$–Kalkül sind keine Sequenzen herleitbar, in welchen gebundene Variablen auftreten.

Übung 80. Zeigen Sie, daß für **N** Schnitt zulässig ist.

3.4 Der Lambek–Kalkül

Der Lambek–Kalkül stellt in mehrfacher Hinsicht eine Erweiterung des **AB**–Kalküls dar. Zum ersten Mal wurde er in Lambek [33] eingeführt. Im Gegensatz zum **AB**–Kalkül werden darin Typen nicht als Mengen von markierten Bäumen interpretiert, sondern als Zeichenketten. Dies hat zur Folge, daß in dem Ableitungskalkül andere Gesetze gelten. Ferner gibt es im Lambek–Kalkül einen weiteren Typenkonstruktor, die *Verkettung*; dieser Typenkonstruktor wird mit $\bullet$ bezeichnet. Auf der Bedeutungsebene entspricht $\bullet$ der Paarbildung, wie wir weiter unter noch sehen werden. Die Typenkonstruktoren für den (klassischen) Lambek–Kalkül sind also $\backslash$, $/$ und $\bullet$. Es sei nun ein Alphabet A gegeben, sowie eine elementare Typenzuweisung ζ. Wir bezeichnen nun mit $[\alpha]_\zeta$ die Menge aller Zeichenketten über A, welche bezüglich ζ den Typ α haben. Dann gilt

$$\begin{array}{lcl} [\alpha \bullet \beta]_\zeta & := & [\alpha]_\zeta \cdot [\beta]_\zeta \\ [\Gamma \circ \Delta]_\zeta & := & [\Gamma]_\zeta \cdot [\Delta]_\zeta \end{array}$$

Da wir $\bullet$ explizit als Konstruktor zur Verfügung haben, können wir im Prinzip auf das externe Symbol $\circ$ verzichten. Wir tun dies aber nicht und formulieren den Kalkül wie vorher mit $\circ$, welches jetzt aber genauso wie $\bullet$ interpretiert wird. Dementsprechend unterscheiden wir wieder zwischen *Termen* und *Strukturen*. Analog zu dem Fall der Konstituentenstrukturen gibt es zwei Interpretationen, nämlich die Kürzungsinterpretation sowie die Kontextinterpretation. In der Kürzungsinterpretation bauen wir die Markierungsfunktion auf aus der Typenzuweisung, in der Kontextinterpretation bauen wir die Interpretation auf aus den Werten für die elementaren Typen. Im Folgenden werden wir uns mit der Kontextinterpretation befassen und nur gelegentlich auf die Kürzungsinterpretation eingehen.

Wir schreiben $\Gamma \vdash \beta$, falls jede Zeichenkette in $[\![\Gamma]\!]$ auch in $[\![\alpha]\!]$ ist. Wir wollen den L–Kalkül $\vdash$ axiomatisieren. Dazu fügen wir folgende Regeln zu dem bisherigen Kalkül **AB** (ohne (schn)) hinzu.

$$\begin{array}{llll} (\mathbf{ass1}) & \dfrac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash \alpha}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash \alpha} & (\mathbf{ass2}) & \dfrac{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash \alpha}{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash \alpha} \\ (\bullet\mathbf{–I}) & \dfrac{\Gamma[\alpha \circ \beta] \vdash \gamma}{\Gamma[\alpha \bullet \beta] \vdash \gamma} & (\mathbf{I}\text{–}\bullet) & \dfrac{\Gamma \vdash \alpha \qquad \Delta \vdash \beta}{\Gamma \circ \Delta \vdash \alpha \bullet \beta} \end{array}$$

Weiterhin bleibt $\alpha \vdash \alpha$ das einzige Axiom. Diesen Kalkül nennen wir den **Lambek**–Kalkül, oder kurz **L**. Der nichtassoziative Lambek–Kalkül, genannt **NL**, ist der Kalkül, der zusätzlich zu **AB** nur die Regeln (**I**–$\bullet$) und ($\bullet$–**I**) enthält. Der Kalkül für die Kürzungsinterpretation ist $\mathbf{L}^- = \mathbf{AB}^- + (\bullet\text{–}\mathbf{I}) + (\mathbf{I}\text{–}\bullet)$. Zunächst wollen wir noch den Begriff einer *Lambek*–Grammatik definieren.

Definition 3.4.1 *Eine **Lambek–Grammatik**, oder kurz **L–Grammatik**, **über** A ist ein Quadrupel $\mathbb{L} = \langle B, A, \zeta, S\rangle$, wo B eine endliche Menge ist, $S \in B$ und ζ eine Typenzuweisung. Es sei $\vec{x} = x_0 \cdot x_1 \cdot \ldots x_{n-1}$ eine Zeichenkette der Länge n über A. Wir schreiben $\mathbb{L} \vdash \vec{x}$, falls für gewisse $\alpha_i \in \zeta(x_i)$, $i < n$, die folgende Sequenz in **L** ableitbar ist*

$$\alpha_0 \circ \alpha_1 \circ \ldots \circ \alpha_{n-1} \vdash S.$$

*Wir schreiben dann $L(\mathbb{L}) = \{\vec{x} : \mathbb{L} \vdash \vec{x}\}$ und nennen es die von $\mathbb{L}$ **erzeugte** oder **akzeptierte** Sprache.*

Theorem 3.4.2 (Lambek) *Die Schnittregel ist zulässig für **L**.*

Korollar 3.4.3 (Lambek) *Der Kalkül **L** ist entscheidbar.*

Zum Beweis müssen wir Anwendungen von (schn) betrachten, in welchen eine der Prämissen durch eine der neuen Regeln entstanden ist. Der Grad ist analog definiert als die Anzahl der Vorkommen von /, \ und •. Es sei die linke Prämisse durch (ass1) entstanden.

$$\dfrac{\dfrac{\Gamma[\Theta_1 \circ (\Theta_2 \circ \Theta_2)] \vdash \alpha}{\Gamma[(\Theta_1 \circ \Theta_2) \circ \Theta_2] \vdash \alpha} \qquad \Delta[\alpha] \vdash \beta}{\Delta[\Gamma[(\Theta_1 \circ \Theta_2) \circ \Theta_3]] \vdash \beta}$$

Diesen Beweis formulieren wir um in den folgenden Beweis

$$\dfrac{\dfrac{\Gamma[\Theta_1 \circ (\Theta_2 \circ \Theta_3)] \vdash \alpha \qquad \Delta[\alpha] \vdash \beta}{\Delta[\Gamma[\Theta_1 \circ (\Theta_2 \circ \Theta_3)]] \vdash \beta}}{\Delta[\Gamma[(\Theta_1 \circ \Theta_2) \circ \Theta_3]] \vdash \beta}$$

Ganz analog im Fall, wo die linke Prämisse durch (ass2) erzeugt worden ist. Wir überlassen dem Leser den Fall, wo die rechte Prämisse durch (ass1) bzw. (ass2) erzeugt worden ist. Es ist nun allerdings zu bemerken, daß im Gegensatz zu den anderen Anwendungen der Schnittgrad nicht reduziert worden ist. Der ursprüngliche Beweis läßt sich also nicht so ohne weiteres übertragen. Allerdings hat sich die *Tiefe* der Anwendung des Schnittes reduziert; die Tiefe ist dabei definiert als die Länge des längsten Pfades im Beweisbaum von einem Axiom bis zu der Anwendung der Regel. Wenn wir annehmen, daß $\Gamma[\Theta_1 \circ (\Theta_2 \circ \Theta_3)] \vdash \alpha$ die Tiefe i hat, und $\Delta[\alpha] \vdash \beta$ die Tiefe j, so hat im ersten Baum die Anwendung von (schn) die Tiefe $max\{i, j\} + 1$, im zweiten dagegen die Tiefe $max\{i, j\}$.

Betrachten wir nun die Fälle der Einführung von •. Der Fall von (•–**I**) auf die linke Prämisse ist einfach.

$$\dfrac{\dfrac{\Gamma[\theta_1 \circ \theta_2] \vdash \alpha}{\Gamma[\theta_1 \bullet \theta_2] \vdash \alpha} \qquad \Delta[\alpha] \vdash \gamma}{\Delta[\Gamma[\theta_1 \bullet \theta_2]] \vdash \gamma} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma[\theta_1 \circ \theta_2] \vdash \alpha \qquad \Delta[\alpha] \vdash \gamma}{\Delta[\Gamma[\theta_1 \circ \theta_2]] \vdash \gamma}}{\Delta[\Gamma[\theta_1 \bullet \theta_2]] \vdash \gamma}$$

Nun der Fall von (**I**–•) auf die rechte Prämisse.

$$\dfrac{\Gamma \vdash \alpha \qquad \dfrac{\Theta_1 \vdash \theta_1 \qquad \Theta_2 \vdash \theta_2}{\Delta[\alpha] \vdash \gamma}}{\Delta[\Gamma] \vdash \gamma}$$

In diesem Fall ist $\gamma = \theta_1 \bullet \theta_2$. Außerdem ist $\Delta = \Theta_1 \circ \Theta_2$, und das markierte Vorkommen von α ist entweder in Θ_1 oder in Θ_2. Ohne Beschränkung der Allgemeinheit sei es in Θ_1. Dann können wir den Beweis ersetzen durch

$$\dfrac{\dfrac{\Gamma \vdash \alpha \qquad \Theta_1[\alpha] \vdash \theta_1}{\Theta_1[\Gamma] \vdash \alpha} \qquad \Theta_2 \vdash \theta_2}{\Theta_1[\Gamma] \circ \Theta_2 \vdash \theta_1 \bullet \theta_2}$$

Es ist $\Theta_1[\Gamma] \circ \Theta_2 = \Delta[\Gamma]$ nach Voraussetzung über das Vorkommen von α. Nun betrachten wir den Fall, wo die linke Prämisse des Schnitts durch (**I**–•) erzeugt wurde. Wir können dann annehmen, daß die rechte Prämisse durch (•–**I**) erzeugt worden ist. Der Fall, wo α Seitenformel ist, ist nämlich wiederum einfach. Also sei α Hauptformel. Wir erhalten folgenden lokalen Baum.

$$\dfrac{\dfrac{\Theta_1 \vdash \theta_1 \qquad \Theta_2 \vdash \theta_2}{\Theta_1 \circ \Theta_2 \vdash \theta_1 \bullet \theta_2} \qquad \dfrac{\Delta[\theta_1 \circ \theta_2] \vdash \gamma}{\Delta[\theta_1 \bullet \theta_2] \vdash \gamma}}{\Delta[\Theta_1 \circ \Theta_2] \vdash \gamma} \quad \rightsquigarrow$$

$$\dfrac{\Theta_2 \vdash \theta_2 \qquad \dfrac{\Theta_1 \vdash \theta_1 \qquad \Delta[\theta_1 \circ \theta_2] \vdash \gamma}{\Delta[\Theta_1 \circ \theta_2] \vdash \gamma}}{\Delta[\Theta_1 \circ \Theta_2] \vdash \gamma}$$

In allen Fällen ist der Schnittgrad reduziert worden. Die Zulässigkeit des Schnittes folgt nun so. Jedes der angegebenen Verfahren verringert entweder den Grad des Schnittes, oder läßt den Grad des Schnittes unverändert und verringert dessen Tiefe. Das Verfahren ist also in jedem Falle endlich, und endet damit, daß sowohl Grad als auch Tiefe aller Schnitte 0 sind, das heißt, alle Schnitte eliminiert werden können (siehe die Diskussion im vorigen Abschnitt).

In Gegenwart der Regeln (ass1) und (ass2) verhält sich $\circ$ genau wie die Verkettungsoperation, d. h. es ist eine voll assoziative Operation. Darum gehen wir im Folgenden zu einer anderen Notation über. Anstatt von Strukturen bestehend aus Typen betrachten wir endliche Folgen von Typen, d. h. Zeichenketten über $Typ_{\backslash,\bullet,/}(B)$. Wir notieren aber, wie in der Beweistheorie auch üblich, die Verkettung schlicht durch Komma.

Es sei $\mathfrak{G} := \langle G, \cdot, ^{-1}, 1\rangle$ eine Gruppe, und $\gamma : B \to G$. Wir erweitern γ auf alle Typen und Strukturen wie folgt:

$$\begin{array}{lcl}
\gamma(\alpha \bullet \beta) & := & \gamma(\alpha) \cdot \gamma(\beta) \\
\gamma(\alpha/\beta) & := & \gamma(\alpha) \cdot \gamma(\beta)^{-1} \\
\gamma(\beta\backslash\alpha) & := & \gamma(\beta)^{-1} \cdot \gamma(\alpha) \\
\gamma(\Gamma \circ \Delta) & := & \gamma(\Gamma) \cdot \gamma(\Delta)
\end{array}$$

Wir nennen γ eine **gruppenwertige Interpretation**.

Theorem 3.4.4 (Roorda) *Falls $\Gamma \vdash \alpha$, so gilt für alle gruppenwertigen Interpretationen γ $\gamma(\Gamma) = \gamma(\alpha)$.*

Der Beweis erfolgt durch Induktion über die Länge der Ableitung und ist dem Leser als Übung überlassen.

Kommen wir nun wieder zu der Bedeutungstheorie zurück. Wir haben im vorigen Abschnitt gezeigt, wie man den **AB**–Kalkül um eine Komponente erweitert, welche zu Sequenzen gleichzeitig auch deren Bedeutungen errechnet. Wir wollen dies hier auch tun. Dabei müssen wir als erstes klären, was die Realisierung von $\alpha \bullet \beta$ sein soll. Hier vereinbahren wir:

$$\ulcorner \alpha \bullet \beta \urcorner := \ulcorner \alpha \urcorner \times \ulcorner \beta \urcorner$$

Die Regeln werden in Bezug auf diese Interpretation zugeschnitten. Die gängigen Regeln sind wie folgt. Die Interpretation der ersten zwei Regeln ist denkbar einfach:

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash \alpha : X}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash \alpha : X}$$

Dies bedeutet also, daß die Umstrukturierung des Terms auf seine Bedeutung keinen Einfluß hat. Ebenso haben wir

$$\frac{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash \alpha : X}{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash \alpha : X}$$

Für die letzten zwei Regeln benötigen wir noch ein klein wenig Notation. Falls $x = \langle y, z \rangle$ ein Paar ist, so sei $p_1(x) = y$ und $p_2(x) = z$. Die Funktionen p_1 und p_2 heißen im Übrigen die **Projektionsfunktionen**.

$$\frac{\Gamma[\alpha : x_\alpha \circ \beta : x_\beta] \vdash \gamma : X}{\Gamma[\alpha \bullet \beta] \vdash \gamma : [p_1(z_{\alpha\bullet\beta})/x_\alpha, p_2(z_{\alpha\bullet\beta})/x_\beta]X}$$

Diese Regel besagt, daß wir anstelle einer Funktion mit zwei Argumenten vom Typ α und β auch eine Funktion mit einem einzigen Argument vom Typ $\alpha \bullet \beta$ nehmen können. Die zwei Argumente können wir uns durch Anwendung der Projektionsfunktionen wieder besorgen. Die vierte Regel schließlich sagt uns, wie der Typ $\alpha \bullet \beta$ interpretiert wird:

$$\frac{\Gamma \vdash \alpha : X \qquad \Delta \vdash \beta : Y}{\Gamma \circ \Delta \vdash \alpha \bullet \beta : \langle X, Y \rangle}$$

Hier stellt sich wie im vorigen Kapitel das Problem, daß die Bedeutungsregeln nicht im Einklang mit der angegebenen Interpretation stehen. Der Term **x(yz)** bezeichnet nicht dieselbe Funktion wie der Term **(xy)z**. (Üblicherweise ist einer von beiden

gar nicht wohldefiniert.) Es stellt sich also die Frage, ob man überhaupt so vorgehen darf. Wir stellen daher einen völlig anderen Kalkül vor, der auf den Kalkül **N** des vorigen Abschnitts aufbaut und diese Schwierigkeiten vermeidet. Die Regeln (**ass1**) und (**ass2**) werden gestrichen. Ferner wird (•–**I**) auf $\Gamma = \varnothing$ beschränkt. Die Beschränkungen für die Abstraktionsregeln werden übernommen.

$$\text{(ax)} \quad \vec{x} : \alpha : x_\alpha \vdash \vec{x} : \alpha : x_\alpha$$

$$(\mathbf{I}\text{–}/) \quad \frac{\Gamma \circ \vec{x} : \alpha : x_\alpha \vdash \vec{y} : \beta : Y}{\Gamma \vdash \vec{y}/\vec{x} : \beta/\alpha : (\lambda x_\alpha.Y)}$$

$$(\mathbf{I}\text{–}\backslash) \quad \frac{\vec{x} : \alpha : x_\alpha \circ \Gamma \vdash \vec{y} : \beta : Y}{\Gamma \vdash \vec{x}\backslash\vec{y} : \alpha\backslash\beta : (\lambda x_\alpha.Y)}$$

$$(\mathbf{E}\text{–}/) \quad \frac{\Gamma \vdash \vec{x} : \alpha : X \qquad \Delta \vdash \vec{y} : \beta/\alpha : Y}{\Delta \circ \Gamma \vdash \vec{y} \cdot \vec{x} : \beta : (XY)}$$

$$(\mathbf{E}\text{–}\backslash) \quad \frac{\Gamma \vdash \vec{x} : \alpha : X \qquad \Delta \vdash \vec{y} : \alpha\backslash\beta : Y}{\Gamma \circ \Delta \vdash \vec{x} \cdot \vec{y} : \beta : (XY)}$$

$$(\bullet\text{–}\mathbf{I}) \quad \frac{\vec{x} : \alpha : x_\alpha \circ \vec{y} : \beta : y_\beta \vdash \vec{z} : \gamma : X}{\vec{x} \cdot \vec{y} : \alpha \bullet \beta : z_{\alpha\bullet\beta} \vdash \vec{z} : \gamma : [p_1(z_{\alpha\bullet\beta})/x_\alpha, p_2(z_{\alpha\bullet\beta})/y_\beta]X}$$

$$(\mathbf{I}\text{–}\bullet) \quad \frac{\Gamma \vdash \vec{x} : \alpha : X \qquad \Delta \vdash \vec{y} : \beta : Y}{\Gamma \circ \Delta \vdash \vec{x} \cdot \vec{y} : \alpha \bullet \beta : \langle X, Y \rangle}$$

Dieser Kalkül ist zwar überhaupt nicht so elegant wie der Lambek–Kalkül, ist aber semantisch korrekt. Falls man die Assoziativität wiederhaben möchte, kann man entweder eine Kombinatorische Erweiterung vornehmen, indem man, grob gesagt, **C** als Modus nimmt, wo **C** definiert ist durch $\mathbf{C}XYZ = X(YZ)$. Oder aber, man nimmt **C** als polymorphes leeres Symbol auf. Die Einzelheiten wollen wir hier allerdings nicht durchgehen.

Übung 81. Es seien anstatt der Sequenzen $\alpha \vdash \alpha$ für beliebiges α nur noch Sequenzen $b \vdash b$, $b \in B$, als Axiome zugelassen. Zeigen Sie, daß sich mittels der Regeln von **L** $\alpha \vdash \alpha$ für jedes α herleiten läßt.

Übung 82. Beweisen Sie das Theorem 3.4.4.

Übung 83. Man definiere analog zur **L**–Grammatik eine $\mathbf{L}^-$–Grammatik. Im Unterschied zu der **L**–Grammatik gilt $\mathbb{L} \vdash \vec{x}$ nur dann, wenn die folgende Sequenz in $\mathbf{L}^-$ ableitbar ist.

$$\alpha_0 \circ \alpha_1 \circ \ldots \circ \alpha_{n-1} \vdash S.$$

Zeigen Sie, daß die von $\mathbf{L}^-$ akzeptierten Sprachen kontextfrei sind.

Übung 84. Zeigen Sie, daß die folgenden Regeln im Lambek–Kalkül zulässig sind

$$(\bullet\text{–}\mathbf{E}) \ \frac{\Gamma[\theta_1 \bullet \theta_2] \vdash \alpha}{\Gamma[\theta_1 \circ \theta_2] \vdash \alpha} \qquad (\mathbf{E}\text{–}/) \ \frac{\Gamma \vdash \alpha/\beta}{\Gamma \circ \beta \vdash \alpha} \qquad (\mathbf{E}\text{–}\backslash) \ \frac{\Gamma \vdash \beta\backslash\alpha}{\beta \circ \Gamma \vdash \alpha}$$

Übung 85. Zeigen Sie, daß die Sequenz $\alpha/\beta \circ \beta/\gamma \vdash \alpha/\gamma$ im Lambek–Kalkül ableitbar ist, nicht jedoch im **AB**–Kalkül. Welche Semantik hat die Struktur $\alpha/\beta \circ \beta/\gamma$?

3.5 Der Satz von Pentus

Im Gegensatz zum **AB**–Kalkül ist es ungewöhnlich schwer zu zeigen, daß die durch den Lambek–Kalkül beschreibbaren Sprachen kontextfrei sind. Dieser Nachweis wurde zuerst von Mati Pentus (siehe [39]) geführt. Es sei B gegeben und $b \in B$. Für einen Typ α über B definieren wir

$$\begin{array}{lcl} \sigma_b(c) & := & \begin{cases} 1 & \text{wenn } b = c \\ 0 & \text{sonst} \end{cases} \\ \sigma_b(\alpha \bullet \beta) & := & \sigma_b(\alpha) + \sigma_b(\beta) \\ \sigma_b(\alpha/\beta) & := & \sigma_b(\alpha) + \sigma_b(\beta) \\ \sigma_b(\beta\backslash\alpha) & := & \sigma_b(\alpha) + \sigma_b(\beta) \end{array}$$

Ebenso definieren wir

$$\begin{array}{lcl} |\alpha| & := & \sum_{b \in B} \sigma_b(\alpha) \\ \pi(\alpha) & := & \{b \in B : \sigma_b(\alpha) > 0\} \end{array}$$

Diese Definitionen werden auf Strukturen kanonisch erweitert. Es sei Δ eine nichtleere Struktur (also $\Delta \neq \varepsilon$) und $\Gamma[-]$ eine Struktur mit einem markierten Vorkommen einer Teilstruktur in Γ. Eine **Interpolante** für eine Sequenz $\Gamma[\Delta] \vdash \alpha$ in einem Kalkül **K** bezüglich Δ ist ein Typ θ derart, daß

1. $\sigma_b(\theta) \leq min\{\sigma_b(\Gamma) + \sigma_b(\alpha), \sigma_b(\Delta)\}$ für alle $b \in B$,
2. $\Delta \vdash \theta$ ist ableitbar in **K**
3. $\Gamma[\theta] \vdash \alpha$ ist ableitbar in **K**

Falls θ diese Bedingungen erfüllt, so gilt insbesondere $\pi(\theta) \subseteq \pi(\Gamma \circ \alpha) \cap \pi(\Delta)$. Wir sagen, **K** habe **Interpolation**, falls für jedes ableitbare $\Gamma[\Delta] \vdash \alpha$ eine Interpolante bezüglich Δ existiert.

Wir interessieren uns für die Kalküle **AB** und **L**. Im Fall von **L** wollen wir noch bemerken, daß in Gegenwart von voller Assoziativität die Interpolationseigenschaft auch wie folgt formuliert werden kann. Wir gehen von Sequenzen der Form $\Gamma \vdash \alpha$ aus, wo Γ schlicht eine Zeichenkette von Typen ist. Falls $\Gamma = \Theta_1, \Delta, \Theta_2$, mit $\Delta \neq \varepsilon$, so existiert eine Interpolante bezüglich Δ. Denn sei etwa Δ° eine Struktur in $\circ$, welche Δ entspricht (nach Weglassen der $\circ$). Dann existiert eine Sequenz $\Gamma^\circ \vdash \alpha$, welche ableitbar ist, in welcher Δ° als Teilstruktur auftritt.

Die Interpolationseigenschaft wird typischerweise durch Induktion über die Länge der Ableitung gezeigt. Im Falle eines Axioms ist nichts zu zeigen. Denn da haben wir eine Sequenz $\alpha \vdash \alpha$, und die markierte Struktur Δ muß α sein. In diesem Fall ist

also α eine Interpolante. Nehmen wir also an, die Regel (**I**–/) sei angewendet worden. Für die Prämissen sei bereits die Behauptung gezeigt. Wir haben dann folgende Konstellation

$$\frac{\Gamma[\Delta] \circ \alpha \vdash \beta}{\Gamma[\Delta] \vdash \beta/\alpha}$$

Wir haben eine Interpolante bezüglich Δ zu finden. Aufgrund der Induktionsannahme existiert eine Interpolante θ derart, daß $\Gamma[\theta] \circ \alpha \vdash \beta$ und $\Delta \vdash \theta$ ableitbar sind und $\sigma_b(\theta) \leq min\{\sigma_b(\Gamma \circ \alpha \circ \beta), \sigma_b(\Delta)\}$ für alle $b \in B$. Dann ist aber $\Gamma[\theta] \vdash \beta/\alpha$ sowie $\Delta \vdash \theta$ ableitbar, und es ist $\sigma_b(\theta) \leq min\{\sigma_b(\Gamma \circ \beta/\alpha), \sigma_b(\Delta)\}$. Also ist θ auch eine Interpolante für $\Gamma[\Delta] \vdash \beta/\alpha$ bezüglich Δ. Der Fall (**I**–\) ist ganz analog.

Nun betrachten wir den Fall, daß die letzte Regel, die angewandt wurde, (/–**I**) ist.

$$\frac{\Gamma \vdash \beta \qquad \Delta[\alpha] \vdash \gamma}{\Delta[\alpha/\beta \circ \Gamma] \vdash \gamma}$$

Nun wählen wir aus $\Delta[\alpha/\beta \circ \Gamma]$ eine Teilstruktur Z. Mehrere Fälle müssen unterschieden werden. (1.) Z ist eine Teilstruktur von Γ, d. h. $\Gamma = \Gamma'[Z]$. Dann existiert eine Interpolante θ bezüglich $\Gamma'[Z] \vdash \beta$. Man rechnet leicht nach, daß θ auch eine Interpolante für $\Delta[\alpha/\beta \circ \Gamma'[Z]] \vdash \gamma$ bezüglich Z ist. (2.) Z ist disjunkt mit dem markierten Vorkommen von $\alpha/\beta \circ \Gamma$. Dann ist $\Delta[\alpha] = \Delta'[Z, \alpha]$ und es existiert eine Interpolante θ bezüglich Z für $\Delta'[Z, \alpha] \vdash \gamma$. Auch hier rechnet man leicht nach, daß θ die gesuchte Interpolante ist. (3.) $Z = \alpha/\beta$. Nach Induktionsvorraussetzung existiert eine Interpolante θ_ℓ für $\Gamma \vdash \beta$ bezüglich Γ, sowie eine Interpolante θ_r für $\Delta[\alpha] \vdash \gamma$ bezüglich α. Dann ist $\theta := \theta_r/\theta_\ell$ eine Interpolante. Es gilt nämlich

$$\begin{array}{rcl} \sigma_b(\theta) & = & \sigma_b(\theta_r) + \sigma_b(\theta_\ell) \\ & \leq & min\{\sigma_b(\Delta \circ \gamma), \sigma_b(\alpha)\} + min\{\sigma_b(\beta), \sigma_b(\Gamma)\} \\ & \leq & min\{\sigma_b(\Delta \circ \Gamma \circ \gamma), \sigma_b(\alpha/\beta)\} \end{array}$$

Ferner gilt

$$\frac{\Gamma \vdash \theta_\ell \qquad \Delta[\theta_r] \vdash \gamma}{\Delta[\theta_r/\theta_\ell \circ \Gamma] \vdash \gamma} \qquad\qquad \frac{\dfrac{\theta_\ell \vdash \beta \qquad \alpha \vdash \theta_r}{\alpha/\beta \circ \theta_\ell \vdash \theta_r}}{\alpha/\beta \vdash \theta_r/\theta_\ell}$$

Also haben wir eine Interpolante für ein Θ. (4.) $Z = \Theta[\alpha/\beta \circ \Gamma]$. Dann ist $\Delta[\alpha/\beta \circ \Gamma] = \Delta'[\Theta[\alpha/\beta \circ \Gamma]]$ für ein gewisses Δ'. Dann existiert nach Voraussetzung eine Interpolante für $\Delta'[\Theta[\alpha]] \vdash \gamma$ bezüglich $\Theta[\alpha]$. Wir zeigen, daß θ die gesuchte Interpolante ist.

$$\frac{\Gamma \vdash \beta \qquad \Theta[\alpha] \vdash \theta}{\Theta[\alpha/\beta \circ \Gamma] \vdash \theta} \qquad\qquad \Delta'[\theta] \vdash \gamma$$

Außerdem ist $\sigma_b(\theta) \leq min\{\sigma_b(\Delta' \circ \gamma), \sigma_b(\Theta[\alpha])\} \leq min\{\sigma_b(\Delta' \circ \gamma), \sigma_b(\Theta[\alpha/\beta \circ \Gamma])\}$. Dies beendet den Fall (/–**I**). Der Fall (\–**I**) ist wiederum analog.

Theorem 3.5.1 *Der Kalkül **AB** hat Interpolation.* ⊣

Wir gehen nun über zum Lambek–Kalkül. Wir können den alten Beweis weiter benutzen, müssen aber weitere Fälle diskutieren. Betrachten wir zunächst nur den Fall, wo wir $\bullet$ hinzufügen mit den Einführungsregeln. Anschließend diskutieren wir den vollen Lambek–Kalkül. Sei also die letzte Regel ($\bullet$–**I**).

$$\frac{\Gamma[\alpha \circ \beta] \vdash \gamma}{\Gamma[\alpha \bullet \beta] \vdash \gamma}$$

(1.) Z enthält nicht das markierte Vorkommen von $\alpha \bullet \beta$. Dann ist $\Gamma[\alpha \bullet \beta] = \Gamma'[Z, \alpha \bullet \beta]$, und nach Induktionsvorraussetzung erhalten wir eine Interpolante θ für $\Gamma'[Z, \alpha \circ \beta] \vdash \gamma$ bezüglich Z. Man rechnet leicht nach, daß θ auch eine Interpolante für $\Gamma'[Z, \alpha \bullet \beta] \vdash \gamma$ bezüglich Z ist. (2.) Es sei $Z = \Theta[\alpha \bullet \beta]$. Dann ist $\Gamma[\alpha \bullet \beta] = \Gamma'[\Theta[\alpha \bullet \beta]]$. Nach Induktionsvorraussetzung existiert eine Interpolante θ für $\Gamma'[\Theta[\alpha \circ \beta]] \vdash \gamma$ bezüglich $\Theta[\alpha \circ \beta]$, und diese ist auch eine Interpolante für $\Gamma'[\Theta[\alpha \bullet \beta]] \vdash \gamma$ bezüglich $\Theta[\alpha \bullet \beta]$. In beiden Fällen haben wir also eine Interpolante gefunden.

Nun wenden wir uns dem Fall ($\bullet$–**I**) zu.

$$\frac{\Gamma \vdash \alpha \qquad \Delta \vdash \beta}{\Gamma \circ \Delta \vdash \alpha \bullet \beta}$$

Es gibt nun drei Fälle für Z. (1.) $\Gamma = \Gamma'[Z]$. Nach Induktionsannahme existiert eine Interpolante θ_ℓ für $\Gamma'[Z] \vdash \alpha$ bezüglich Z. Diese ist schon die gewünschte Interpolante. (2.) $\Delta = \Delta'[Z]$. Analog wie (1.). (3.) $Z = \Gamma \circ \Delta$. Nach Voraussetzung existiert eine Interpolante θ_ℓ für $\Gamma \vdash \alpha$ bezüglich Γ und eine Interpolante θ_r für $\Delta \vdash \beta$ bezüglich Δ. Setze $\theta := \theta_\ell \bullet \theta_r$. Dies ist die gesuchte Interpolante. Denn

$$\frac{\Gamma \vdash \theta_\ell \qquad \Delta \vdash \theta_r}{\Gamma \circ \Delta \vdash \theta_\ell \bullet \theta_r} \qquad\qquad \frac{\dfrac{\theta_\ell \vdash \alpha \qquad \theta_r \vdash \beta}{\theta_\ell \circ \theta_r \vdash \alpha \bullet \beta}}{\theta_\ell \bullet \theta_r \vdash \alpha \bullet \beta}$$

Außerdem rechnet man leicht nach, daß $\sigma_b(\theta) \leq min\{\sigma_b(\alpha \bullet \beta), \sigma_b(\Gamma \circ \Delta)\}$ ist.

Zu guter letzt müssen wir noch den vollen Lambek–Kalkül studieren. Die Regel(n) (**ass1**), (**ass2**) stellen uns vor ein technisches Problem. Wir können den Beweis jetzt nicht mehr induktiv über die Länge des Beweises führen, da im Moment der Anwendung von (**ass1**) bzw. (**ass2**) der Aufbau der Struktur geändert wurde. Wir wechseln daher nun unsere Strukturen und betrachten Sequenzen von der Form $\Gamma \vdash \alpha$, wo Γ schlicht eine endliche Folge von Typen ist. In diesem Fall sind die Regeln (**ass1**) und (**ass2**) herauszunehmen. Allerdings müssen wir im Beweis der Interpolationseigenschaft mehr Fallunterscheidungen vornehmen. Die Regeln (**I**–/) sowie (**I**–\) sind weiterhin unproblematisch. Betrachten wir daher einen komplizierten Fall, nämlich eine Anwendung der Regel (/–**I**).

$$\frac{\Gamma \vdash \beta \qquad \Delta[\alpha] \vdash \gamma}{\Delta[\alpha/\beta, \Gamma] \vdash \gamma}$$

Wir können die Struktur $\Delta[\alpha/\beta, \Gamma]$ zerlegen in $\Delta', \alpha/\beta, \Gamma, \Delta''$. Sei eine Teilsequenz Z in $\Delta', \alpha/\beta, \Gamma, \Delta''$ ausgezeichnet. Der Fall, wo Z ganz in Δ' enthalten ist, ist relativ einfach, ebenso der Fall, wo Z ganz in Δ'' enthalten ist. Es bleiben nun folgende Fälle übrig. (1.) $Z = \Delta_1, \alpha/\beta, \Gamma_1$, wo $\Delta' = \Delta_0, \Delta_1$ für ein gewisses Δ_0, und $\Gamma = \Gamma_1, \Gamma_2$ für ein gewisses Γ_2. Auch wenn Z nichtleer ist, können sowohl Δ_1 als auch Γ_1 leer sein. Sei hier nun noch $\Gamma_1 \neq \varepsilon$ vorausgesetzt. In diesem Fall sei θ_ℓ eine Interpolante für $\Gamma \vdash \beta$ bezüglich Γ_2 und θ_r eine Interpolante von $\Delta[\alpha] \vdash \gamma$ bezüglich Δ_1, α. (Hier wird klar, warum wir $\Delta_1 \neq \varepsilon$ nicht voraussetzen müssen.) Folgende Sequenzen sind daher ableitbar.

$$\begin{array}{ll} \Gamma_2 \vdash \theta_\ell & \Gamma_1, \theta_\ell \vdash \beta \\ \Delta_1, \alpha \vdash \theta_r & \Delta_0, \theta_r, \Delta'' \vdash \gamma \end{array}$$

Nun setze $\theta := \theta_r/\theta_\ell$. Dann haben wir

$$\dfrac{\dfrac{\Delta_1, \alpha \vdash \theta_r \qquad \Gamma_1, \theta_\ell \vdash \beta}{\Delta_1, \alpha/\beta, \Gamma_1, \theta_\ell \vdash \theta_r}}{\Delta_1, \alpha/\beta, \Gamma_1 \vdash \theta_r/\theta_\ell} \qquad\qquad \dfrac{\Gamma_2 \vdash \theta_\ell \qquad \Delta_0, \theta_r, \Delta'' \vdash \gamma}{\Delta_0, \theta_r/\theta_\ell, \Gamma_2, \Delta'' \vdash \gamma}$$

Die Bedingung über die Anzahl der Symbolvorkommen ist leicht nachzuprüfen. (2.) Wie im Fall (2.), mit Γ_1 leer. Sei dann θ_ℓ eine Interpolante für $\Gamma \vdash \beta$ bezüglich Γ und θ_r eine Interpolante für $\Delta_0, \Delta_1, \alpha, \Delta'' \vdash \gamma$ bezüglich Δ_1, α. Dann setze $\theta := \theta_r/\theta_\ell$. θ ist eine Interpolante für die Endsequenz bezüglich Z.

$$\dfrac{\dfrac{\theta_\ell \vdash \beta \qquad \Delta_1, \alpha \vdash \theta_r}{\Delta_1, \alpha/\beta, \theta_\ell \vdash \theta_r}}{\Delta_1, \alpha/\beta \vdash \theta_r/\theta_\ell} \qquad\qquad \dfrac{\Gamma \vdash \theta_\ell \qquad \Delta_0, \theta_r, \Delta'' \vdash \gamma}{\Delta_0, \theta_r/\theta_\ell, \Gamma, \Delta'' \vdash \gamma}$$

(3.) Z enthält das markierte Vorkommen von α/β nicht. In diesem Fall ist $Z = \Gamma_2, \Delta_1$ für ein Endstück Γ_2 von Γ und ein Anfangsstück Δ_1 von Δ''. Sowohl Γ_2 als auch Δ_1 können wir nun als nichtleer voraussetzen, da wir ansonsten einen bereits diskutierten Fall erhalten. Die Situation ist also wie folgt, für $Z = \Gamma_2, \Delta_1$.

$$\dfrac{\Gamma_1, \Gamma_2 \vdash \beta \qquad \Delta', \alpha, \Delta_1, \Delta_2 \vdash \gamma}{\Delta', \alpha/\beta, \Gamma_1, \Gamma_2, \Delta_1, \Delta_2 \vdash \gamma}$$

Es sei θ_ℓ eine Interpolante für $\Gamma_1, \Gamma_2 \vdash \beta$ bezüglich Γ_2 und θ_r eine Interpolante für $\Delta', \alpha, \Delta_1, \Delta_2 \vdash \gamma$ bezüglich Δ_1. Dann sind ableitbar

$$\begin{array}{ll} \Gamma_2 \vdash \theta_\ell & \Gamma_1, \theta_\ell \vdash \beta \\ \Delta_1 \vdash \theta_r & \Delta', \alpha, \theta_r, \Delta_2 \vdash \gamma \end{array}$$

Nun wählen wir $\theta := \theta_\ell \bullet \theta_r$. Dann haben wir

$$\dfrac{\Gamma_2 \vdash \theta_\ell \qquad \Delta_1 \vdash \theta_r}{\Gamma_2, \Delta_1 \vdash \theta_\ell \bullet \theta_r} \qquad\qquad \dfrac{\dfrac{\Gamma_1, \theta_\ell \vdash \beta \qquad \Delta', \alpha, \theta_r, \Delta_2 \vdash \gamma}{\Delta', \alpha/\beta, \Gamma_1, \theta_\ell, \theta_r, \Delta_2 \vdash \gamma}}{\Delta', \alpha/\beta, \Gamma_1, \theta_\ell \bullet \theta_r, \Delta_2 \vdash \gamma}$$

Auch hier prüft man die Abzählbedingung direkt nach. Damit sind nun alle Fälle erschöpft. Man beachte, daß wir $\bullet$ zur Konstruktion der Interpolante benutzt haben. Der Fall der Regeln (**I**–$\bullet$) und ($\bullet$–**I**) bietet gegenüber dem Fall von **AB** keine Überraschungen.

Theorem 3.5.2 (Roorda) *Der Lambek–Kalkül hat Interpolation.* $\dashv$

Nun wollen wir den Nachweis der Kontextfreiheit des Lambek–Kalküls angehen. Dazu führen wir eine Schar von erheblich beschränkteren Kalkülen ein, von denen gezeigt wird, daß sie zusammengenommen nicht schwächer sind als **L**. Diese Kalküle heißen $\mathbf{L}_m$, $m < \omega$. Die Axiome des Kalküls $\mathbf{L}_m$ sind Sequenzen $\Gamma \vdash \alpha$ derart, daß

1. $\Gamma = \beta_1, \beta_2$ oder $\Gamma = \beta_1$ für gewisse Typen β_1, β_2
2. $\Gamma \vdash \alpha$ ist ableitbar in **L**
3. $|\alpha|, |\beta_1|, |\beta_2| < m$

Die einzige Schlußregel ist (schn). Die Hauptarbeit liegt im Beweis des folgenden Satzes.

Theorem 3.5.3 (Pentus) *Es sei $\Gamma = \beta_0, \beta_1, \ldots, \beta_{n-1}$. Genau dann ist $\Gamma \vdash \alpha$ ableitbar in $\boldsymbol{L}_m$, wenn $|\beta_i| < m$ für alle $i < m$, $|\alpha| < m$ und $\Gamma \vdash \alpha$ in $\boldsymbol{L}$ ableitbar ist.*

Wir zeigen zunächst, wie man aus diesem Resultat beweisen kann, daß **L**–Grammatiken kontextfrei sind. Wir schwächen dazu den Kalkül noch ein wenig mehr ab. Der Kalkül $\mathbf{L}_m^{\boxminus}$ hat die Axiome von $\mathbf{L}_m$, aber (schn) darf nur dann angewendet werden, wenn die linke Prämisse ein Axiom ist.

Lemma 3.5.4 *Für alle Sequenzen $\Gamma \vdash \alpha$ gilt: $\Gamma \vdash \alpha$ ist in $\boldsymbol{L}_m^{\boxminus}$ ableitbar genau dann, wenn $\Gamma \vdash \alpha$ in $\boldsymbol{L}_m$ ableitbar ist.*

Der Beweis ist relativ einfach und als Übung überlassen.

Theorem 3.5.5 *Die von L–Grammatiken akzeptierten Sprachen sind kontextfrei.*

Beweis. Es sei $\mathbb{L} = \langle B, A, \zeta, S\rangle$ gegeben. Es sei m größer als die das Maximum aller $|\alpha|$, $\alpha \in \zeta(a)$, $a \in A$. Da sowohl A als auch $\zeta(a)$ endlich sind, existiert m. Der

Einfachheit halber nehmen wir an, daß $B = \bigcup\langle \pi(\alpha) : \alpha \in \zeta(a), a \in B\rangle$. Wir setzen nun $N := \{\alpha : |\alpha| < m\}$. $G := \langle N, A, S, R\rangle$, wobei

$$\begin{aligned} R := \quad & \{\alpha \to a : a \in \zeta(a)\} \\ \cup \quad & \{\alpha \to \beta : \alpha, \beta \in N, \beta \vdash \alpha \text{ ist L–ableitbar}\} \\ \cup \quad & \{\alpha \to \beta_0\beta_1 : \alpha, \beta_0, \beta_1 \in N, \beta_0, \beta_1 \vdash \alpha \text{ ist L–ableitbar}\} \end{aligned}$$

Nun sei $\mathbb{L} \vdash \vec{x}$, $\vec{x} = x_0 \cdot x_1 \cdot \ldots x_{n-1}$. Dann existieren $\alpha_i \in \zeta(x_i)$ für alle $i < n$, derart daß $\Gamma \vdash S$ in $\mathbf{L}$ ableitbar ist, wobei $\Gamma := \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$. Nach Theorem 3.5.3 und Lemma 3.5.4 ist dann $\Gamma \vdash S$ auch in $\mathbf{L}_m^{\boxminus}$ ableitbar. Induktion über die Länge der Ableitung liefert nun, daß $\vdash_G \alpha_0 \cdot \alpha_1 \cdot \ldots \cdot \alpha_{n-1}$, und somit auch $\vdash_G \vec{x}$. Jetzt sei umgekehrt $\vdash_G \vec{x}$. Wir erweitern die Typenzuweisung ζ zu $\zeta^+ : A \cup N \to \mathit{Typ}_{\backslash,\bullet,/}(B)$, indem wir setzen $\zeta^+(\alpha) := \{\alpha\}$, während $\zeta^+ \restriction A = \zeta$. Induktiv über die Länge der Ableitung von $\vec{\alpha}$ beweist man nun, daß aus $\vdash_G \vec{\alpha}$ folgt $\mathbb{L} \vdash \vec{\alpha}$. $\dashv$

Nun zum Beweis von Satz 3.5.3.

Definition 3.5.6 *Ein Typ α heißt **dünn**, falls $\sigma_b(\alpha) \leq 1$ für alle $b \in B$. Eine Sequenz $\Gamma \vdash \alpha$ heißt **dünn**, falls gilt:*

1. *$\Gamma \vdash \alpha$ ist in $\boldsymbol{L}$ ableitbar.*
2. *Alle Typen, die in Γ und α vorkommen, sind dünn.*
3. *$\sigma_b(\Gamma, \alpha) \leq 2$ für alle $b \in B$.*

Für einen dünnen Typ α gilt stets $|\alpha| = |\pi(\alpha)|$. Wir merken an, daß für eine dünne Sequenz nur $\sigma_b(\Gamma, \alpha) = 0$ oder $= 2$ in Frage kommt, da $\sigma_b(\Gamma, \alpha)$ immer eine gerade Zahl ist. Betrachten wir nun eine dünne Sequenz $\Gamma[\Delta] \vdash \alpha$ und eine Interpolante θ bezüglich Δ. Dann gilt $\sigma_b(\theta) \leq \sigma_b(\Delta) \leq 1$. Denn entweder ist $b \notin \pi(\Delta)$, und dann ist $b \notin \pi(\theta)$, also $\sigma_b(\theta) = 0$. Oder $b \in \pi(\Delta)$, aber dann $b \in \pi(\Gamma, \alpha)$, und so nach Voraussetzung $\sigma_b(\Delta) = 1$.

$$\sigma_b(\Delta, \theta) \leq \sigma_b(\Delta) + \sigma_b(\theta) \leq \sigma_b(\Gamma[\Delta], \alpha) + \sigma_b(\theta) \leq 2 + 1$$

Nun ist $\sigma_b(\Delta) + \sigma_b(\theta)$ eine gerade Zahl, also entweder 0 oder 2. Also ist $\Delta \vdash \theta$ dünn. Ebenso zeigt man, daß auch $\Gamma[\theta] \vdash \alpha$ dünn ist.

Lemma 3.5.7 *Es sei $\Gamma, \Theta, \Delta \vdash \alpha$ eine Sequenz, und $b, c \in B$ zwei verschiedene elementare Typen. Ferner sei $b \in \pi(\Gamma) \cap \pi(\Delta)$ sowie $c \in \pi(\Theta) \cap \pi(\alpha)$. Dann ist $\Gamma, \Theta, \Delta \vdash \alpha$ nicht dünn.*

Beweis. Es sei $\mathfrak{F}_G(B)$ die von den elementaren Typen frei erzeugte Gruppe. Die Elemente dieser Gruppe sind endliche Produkte $b_0^{s_0} \cdot b_2^{s_2} \cdot \ldots \cdot b_{n-1}^{s_{n-1}}$, wo $b_i \neq b_{i+1}$ ist für $i < n-1$, und $s_i \in \mathbb{Z}$. (Falls $n = 0$, so bezeichne das leere Produkt gerade die 1.) Falls nämlich z. B. $b_0 = b_1$ ist, so kann der Term $b_0^{s_0} \cdot b_1^{s_1}$ gekürzt werden zu $b_0^{s_0+s_1}$. Betrachte die gruppenwertige Interpretation $\gamma : b \mapsto b$. Wäre die Sequenz dünn, so müßte gelten $\gamma(\Gamma) \cdot \gamma(\Theta) \cdot \gamma(\Delta) = \gamma(\alpha)$. Nach Voraussetzung ist die linke Seite von der Form $w \cdot b^{\pm 1} \cdot x \cdot c^{\pm 1} \cdot y \cdot b^{\pm 1} \cdot z$ für irgendwelche Produkte w, x, y, z. Die rechte Seite ist gleich $t \cdot c^{\pm 1} \cdot u$ für gewisse t, u. Wir wissen ferner, daß Terme, die für w, x, y, z sowie t und u stehen, in maximal gekürzter Form nicht b und c enthalten. Dann kann aber die Gleichheit nicht gelten. $\dashv$

Lemma 3.5.8 *Es sei $n > 0$, und die Sequenz $\alpha_0, \alpha_1, \ldots, \alpha_n \vdash \alpha_{n+1}$ sei dünn. Dann existiert ein k mit $0 < k < n+1$ derart, daß $\pi(\alpha_k) \subseteq \pi(\alpha_{k-1}) \cup \pi(\alpha_{k+1})$.*

Beweis. Der Beweis erfolgt durch Induktion über n. Wir beginnen mit $n = 1$. Hier hat die Sequenz die Form $\alpha_0, \alpha_1 \vdash \alpha_2$. Es sei $b \in \pi(\alpha_1)$. Dann ist $\sigma_b(\alpha_1) = 1$, da die Sequenz dünn ist, und da $\sigma_b(\alpha_0, \alpha_1, \alpha_2) = 2$, so ist $\sigma_b(\alpha_0, \alpha_2) = 1$, also $b \in \pi(\alpha_0) \cup \pi(\alpha_2)$. Dies beendet den Fall $n = 1$. Nun sei $n > 1$ und die Behauptung für alle $m < n$ bereits gezeigt. Wir betrachten zwei Fälle. **Fall a.** $\pi(\alpha_0, \alpha_1, \ldots, \alpha_{n-2}) \cap \pi(\alpha_n) = \emptyset$. Dann wählen wir $k = n$. Denn wenn $b \in \pi(\alpha_n)$, so ist $\sigma_b(\alpha_0, \ldots, \alpha_{n-2}) = 0$, und so $\sigma_b(\alpha_{n-1}) + \sigma_b(\alpha_{n+1}) = 1$. Also $b \in \pi(\alpha_{n-1})$ oder $b \in \pi(\alpha_{n+1})$. **Fall b.** $\pi(\alpha_0, \alpha_1, \ldots, \alpha_{n-2}) \cap \pi(\alpha_n) \neq \emptyset$. Dann existiert ein elementarer Typ b mit $b \in \pi(\alpha_0, \ldots, \alpha_{n-2})$ und $b \in \pi(\alpha_n)$. Setze $\Gamma := \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$, $\Delta := \alpha_n$. Es sei θ eine Interpolante für $\Gamma, \Delta \vdash \alpha_{n+1}$ bezüglich Γ. Dann sind $\Gamma \vdash \theta$ sowie $\theta, \alpha_n \vdash \alpha_{n+1}$ dünn. Nach Induktionsvoraussetzung existiert ein k derart, daß $\pi(\alpha_k) \subseteq \pi(\alpha_{k-1}) \cup \pi(\alpha_{k+1})$, falls $k < n-1$, bzw. $\pi(\alpha_k) \subseteq \pi(\alpha_{k-1}) \cup \pi(\theta)$, falls $k = n-1$. Im Falle $k < n-1$ ist k schon die gesuchte Zahl für die Ausgangssequenz. Sei nun $k = n-1$. Dann ist $\pi(\alpha_{n-1}) \subseteq \pi(\alpha_{n-2}) \cup \pi(\theta) \subseteq \pi(\alpha_{n-2}) \cup \pi(\alpha_n) \cup \pi(\alpha_{n+1})$. Wir zeigen, daß k auch in diesem Fall die gesuchte Zahl für die Ausgangssequenz ist. Sei $\pi(\alpha_{n-1}) \cap \pi(\alpha_{n+1}) \neq \emptyset$, etwa $c \in \pi(\alpha_{n-1}) \cap \pi(\alpha_{n+1})$. Dann ist sicherlich $c \notin \pi(\alpha_n)$, also $c \neq b$. Also ist die Sequenz wegen Lemma 3.5.7 nicht dünn. Deswegen gilt $\pi(\alpha_{n-1}) \cap \pi(\alpha_{n+1}) = \emptyset$, und so $\pi(\alpha_{n-1}) \subseteq \pi(\alpha_{n-2}) \cup \pi(\alpha_n)$. $\dashv$

Lemma 3.5.9 *Es sei $\Gamma \vdash \gamma$ eine dünne Sequenz, in der alle Typen eine Länge $< m$ haben. Dann ist $\Gamma \vdash \gamma$ schon in $\boldsymbol{L}_m$ ableitbar.*

Beweis. Sei $\Gamma = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$; setze $\alpha_n := \gamma$. Falls $n \leq 2$, so ist $\Gamma \vdash \gamma$ bereits ein Axiom von $\mathbf{L}_m$. Also sei $n > 2$. Nach dem vorangegangenen Lemma existiert ein k mit $\pi(\alpha_k) \subseteq \pi(\alpha_{k-1}) \cup \pi(\alpha_{k+1})$. **Fall 1.** $k < n$. **Fall 1a.** $|\pi(\alpha_{k-1}) \cap \pi(\alpha_k)| \geq |\pi(\alpha_{k+1}) \cap \pi(\alpha_k)|$. Setze $\Xi := \alpha_0, \alpha_1, \ldots, \alpha_{k-2}$, $\Delta := \alpha_{k-1}, \alpha_k$, $\Theta := \alpha_{k+1}, \ldots, \alpha_{n-1}$. Es sei θ eine Interpolante für $\Xi, \Delta, \Theta \vdash \alpha_n$ bezüglich Δ. Dann ist die Sequenz

$$\alpha_0, \ldots, \alpha_{k-2}, \theta, \alpha_{k+1}, \ldots, \alpha_{n-1} \vdash \alpha_n$$

dünn. Ferner ist

$$\begin{array}{rcl}\pi(\theta) & \subseteq & (\pi(\alpha_{k-1}) \cup \pi(\alpha_k)) \cap \pi(\Xi, \Theta, \alpha_n) \\ & = & (\pi(\alpha_{k-1}) \cap \pi(\Xi, \Theta, \alpha_n)) \cup (\pi(\alpha_k) \cap \pi(\Xi, \Theta, \alpha_n)).\end{array}$$

Sei $b \in \pi(\alpha_{k-1})$. Dann $\sigma_b(\alpha_{k-1}) = 1$, $\sigma_b(\Xi, \alpha_{k-1}, \alpha_k, \Theta, \alpha_n) = 2$, also $\sigma_b(\Xi, \alpha_k, \Theta, \alpha_n) = 1$. Also ist entweder $\sigma_b(\alpha_k) = 1$ oder $\sigma_b(\Xi, \Theta, \alpha_n) = 1$. Da b beliebig war, gilt $\pi(\alpha_k) \cap \pi(\Xi, \Theta, \alpha_n) = \pi(\alpha_{k-1}) - (\pi(\alpha_{k-1}) \cap \pi(\alpha_k))$. Nach Wahl von k ist $\pi(\alpha_k) \cap \pi(\Xi, \Theta, \alpha_n) = \pi(\alpha_k) \cap \pi(\alpha_{k+1})$. Deswegen gilt jetzt

$$\begin{array}{rcl}\pi(\theta) & \subseteq & (\pi(\alpha_{k-1}) \cap \pi(\Xi, \Theta, \alpha_n)) \cup (\pi(\alpha_k) \cap \pi(\Xi, \Theta, \alpha_n)) \\ & \subseteq & (\pi(\alpha_k) - (\pi(\alpha_{k-1}) \cap \pi(\alpha_k)) \cup (\pi(\alpha_k) \cap \pi(\alpha_{k+1})).\end{array}$$

Also

$$\begin{array}{rcl}|\pi(\theta)| & = & |\pi(\alpha_{k-1})| + |\pi(\alpha_{k-1}) \cap \pi(\alpha_k)| + |\pi(\alpha_k) \cap \pi(\alpha_{k+1})| \\ & \leq & |\pi(\alpha_{k-1})| \\ & < & m.\end{array}$$

(Man beachte, daß $|\pi(\alpha_{k-1})| = |\alpha_k|$ ist.) Deswegen ist auch $|\theta| < m$ und so $\alpha_{k-1}, \alpha_k \vdash \theta$ ein Axiom von $\mathbf{L}_m$. Nach Induktionsvoraussetzung ist $\Xi, \theta, \Theta \vdash \alpha_n$ nun in $\mathbf{L}_m$ ableitbar. Einmalige Anwendung aus beiden Sequenzen ergibt aber die Ausgangssequenz. Diese ist also in $\mathbf{L}_m$ ableitbar. **Fall 1b.** $|\pi(\alpha_{k-1}) \cap \pi(\alpha_k)| < |\pi(\alpha_k) \cap \pi(\alpha_{l+1})|$. Hier setze man $\Xi = \alpha_0, \ldots, \alpha_{k-1}$, $\Delta := \alpha_k, \alpha_{k+1}$, $\Theta := \alpha_{k+1}, \ldots, \alpha_{n-1}$ und verfahre wie im Fall 1a. **Fall 2.** $k = n-1$. Dies bedeutet $\pi(\alpha_{n-1}) \subseteq \pi(\alpha_{n-2}) \cup \pi(\gamma)$. Auch hier unterscheiden wir zwei Fälle. **Fall 2a.** $|\pi(\alpha_{n-2}) \cap \pi(\alpha_{n-1})| \geq |\pi(\alpha_{n-1}) \cap \pi(\alpha_n)|$. Dieser Fall ist ähnlich zum Fall 1a. **Fall 2b.** $|\pi(\alpha_{n-2}) \cap \pi(\alpha_{n-1})| < |\pi(\alpha_{n-1}) \cap \pi(\alpha_n)|$. Hier setze $\Delta := \alpha_0, \ldots, \alpha_{n-2}$, $\Theta := \alpha_{n-1}$. Sei θ eine Interpolante für $\Delta, \Theta \vdash \alpha_n$ bezüglich Δ. Dann ist $\Delta \vdash \theta$ sowie $\theta, \alpha_{n-1} \vdash \alpha_n$ dünn. Ferner gilt

$$\begin{array}{rcl}\pi(\theta) & \subseteq & \pi(\Delta) \cap (\pi(\alpha_{n-1}) \cup \pi(\alpha_n)) = (\pi(\Delta) \cap \pi(\alpha_{n-1})) \cup (\pi(\Delta) \cap \pi(\alpha_n)) \\ & = & (\pi(\alpha_{n-2}) \cap \pi(\alpha_{n-1})) \cup (\pi(\alpha_n) - (\pi(\alpha_{n-1}) \cap \pi(\alpha_n))).\end{array}$$

Ebenso wie im Fall 1a folgern wir, daß

$$\begin{array}{rcl}|\pi(\theta)| & = & |\pi(\alpha_{n-2}) \cap \pi(\alpha_{n-1})| + |\pi(\alpha_n)| - |\pi(\alpha_{n-1}) \cap \pi(\alpha_n)| \\ & < & |\pi(\alpha_n)| \\ & < & m.\end{array}$$

Also ist $\theta, \alpha_{n-1} \vdash \alpha_n$ ein Axiom von $\mathbf{L}_m$. Nach Induktionsvoraussetzung ist $\Delta \vdash \theta$ in $\mathbf{L}_m$ ableitbar. Einmalige Anwendung von (schn) ergibt die Ausgangssequenz, welche also in $\mathbf{L}_m$ ableitbar ist. $\dashv$

Nun schließlich zum Beweis des Satzes 3.5.3. Es sei $|\beta_i| < m$ für alle $i < n$, und $|\alpha| < m$. Schließlich sei $\beta_0, \beta_1, \ldots, \beta_{m-1} \vdash \alpha$ in $\mathbf{L}$ ableitbar. Wir wählen einen Beweis dieser Sequenz. Wir können dabei annehmen, daß wir als Axiome nur Sequenzen der Form $b \vdash b$ verwendet haben. Für jedes Vorkommen eines Axioms $b \vdash b$ wählen wir

einen neuen elementaren Typ $\widehat{b}$ und ersetzen dieses Vorkommen von $b \vdash b$ durch $\widehat{b} \vdash \widehat{b}$. Wir führen diese Ersetzung in dem ganzen Beweis durch, sodaß wir nun einen neuen Beweis einer Sequenz $\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_{n-1} \vdash \widehat{\alpha}$ erhalten. Wir haben $\sigma_b(\widehat{\alpha}) + \sum_{i<n} \sigma_b(\widehat{\beta}_i) = 2$, falls b in der Sequenz überhaupt vorkommt. Trotzdem muß die Sequenz nicht dünn sein, denn es können Typen in ihr enthalten sein, die nicht dünn sind. Falls aber $\sigma_b(\delta) = 2$ für ein δ und ein b, so ist b in keinem weiteren Typ enthalten. Wir nutzen dies wie folgt aus. Durch Sukzessive Anwendung von Interpolation bekommen wir folgende in **L** ableitbare Sequenzen.

$$\begin{array}{rl}
\widehat{\beta}_0 \vdash \theta_0 & \theta_0, \widehat{\beta}_1, \widehat{\beta}_2, \ldots, \widehat{\beta}_{n-1} \vdash \widehat{\alpha} \\
\widehat{\beta}_1 \vdash \theta_1 & \theta_0, \theta_1, \widehat{\beta}_2, \ldots, \widehat{\beta}_{n-1} \vdash \widehat{\alpha} \\
\vdots & \vdots \\
\widehat{\beta}_{n-1} \vdash \theta_{n-1} & \theta_0, \theta_1, \ldots, \theta_{n-1} \vdash \widehat{\alpha} \\
\theta_0, \theta_1, \ldots, \theta_{n-1} \vdash \gamma & \gamma \vdash \widehat{\alpha}
\end{array}$$

Es ist nicht schwer zu zeigen, daß $\sigma_b(\theta_i) \leq 1$ für alle b und alle $i < n$ ist. Also ist die Sequenz $\theta_0, \theta_1, \ldots, \theta_{n-1} \vdash \gamma$ dünn. Sicher ist $|\gamma| \leq |\widehat{\alpha}| = |\alpha| < m$ sowie $|\theta_i| \leq |\widehat{\alpha}_i| = |\alpha_i| < m$ für alle $i < n$. Nach Lemma 3.5.9 ist also $\theta_0, \theta_1, \ldots, \theta_{n-1} \vdash \gamma$ in $\mathbf{L}_m$ ableitbar. Die Sequenzen $\widehat{\beta}_i \vdash \theta_i$, $i < n$, sowie $\gamma \vdash \widehat{\alpha}_n$ sind Axiome von $\mathbf{L}_m$. Also ist $\widehat{\beta}_0, \widehat{\beta}_0, \ldots, \widehat{\beta}_{n-1} \vdash \widehat{\alpha}$ in $\mathbf{L}_m$ ableitbar. Wir machen nun die Ersetzung in der Ableitung rückgängig. Dies können wir durch eine Substitution t tun, welche $\widehat{b}$ auf b abbildet. Dadurch bekommen wir nun eine Ableitung von $\beta_0, \beta_1, \ldots, \beta_{n-1} \vdash \beta_n$ in $\mathbf{L}_m$. Dies beendet den Beweis von Satz 3.5.3.

Wir merken an, daß Pentus in [38] gezeigt hat, daß der Lambek–Kalkül vollständig in Bezug auf die Kürzungsinterpretation auf Halbgruppen ist.

Übung 86. Es sei $\Gamma \vdash \alpha$ ableitbar. Dann ist $\sigma_b(\Gamma) + \sigma_b(\alpha)$ eine gerade Zahl.

Übung 87. Man beweise Lemma 3.5.4.

3.6 Montague–Semantik

Bis Anfang der 70er Jahre galt die Semantik natürlicher Sprachen als ein wenig aussichtsreiches Gebiet. Natürliche Sprache galt (und gilt vielerorts noch) als derart unstrukturiert, daß es unmöglich schien, eine formale Bedeutungstheorie aufzubauen. Montague hat dagegen die These vertreten, daß man natürliche Sprachen genauso behandeln könne wie formale Sprachen. Auch wenn diese Behauptung überzogen sein mag (wahrscheinlich hat Montague bewußt ein wenig übertrieben), so enthält sie doch genügend Wahres. Montague hat selbst den Beweis für seine These angetreten und einen Entwurf für ein kleines Fragment des Englischen vorgelegt. Wir wollen hier nur einen kleinen Einblick in die von Montague geschaffene Theorie werfen. Dazu müssen wir zunächst über Prädikatenlogik und ihre Modelle reden. Denn

Montague hat seine Semantik etwas anders aufgezogen, als wir das bisher getan haben. Anstatt eine Interpretation in einem Modell direkt zu definieren, definiert Montague eine Übersetzung in den λ–Kalkül über Prädikatenlogik, deren Interpretation bereits festliegt.

Eine Sprache der Prädikatenlogik erster Stufe mit Gleichheit besitzt folgende Symbole:

1. eine Menge R von Relationssymbolen eine Menge F von Funktionssymbolen,
2. eine unendliche, abzählbare Menge $V := \{x_i : i \in \omega\}$ von Variablen,
3. die Gleichheit $\doteq$,
4. die Aussagenjunktoren $\neg$, $\wedge$, $\vee$, $\rightarrow$,
5. die Quantoren $\exists$, $\forall$.

Hierbei sei r jeweils $\Xi(r)$–stellig und f $\Omega(f)$–stellig. Die Gleichheit ist ein 2–stelliges Relationssymbol. Wir definieren zuerst Terme:

* x ist ein Term für $x \in V$.
* Sind t_i, $i < \Omega(f_j)$, Terme, so ist auch $f_j(t_0, \ldots, t_{\Omega(f_j)-1})$ ein Term.

Als nächstes definieren wir Formeln (siehe dazu auch Abschnitt 2.6):

* Sind t_i, $i < \Xi(r_j)$, Terme, so ist $r_j(t_0, \ldots, t_{\Xi(r_j)-1})$ eine Formel.
* Sind t_0 und t_1 Terme, so ist $t_0 \doteq t_1$ eine Formel.
* Sind φ und ψ Formeln, so auch $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$ und $\varphi \rightarrow \psi$.
* Ist φ eine Formel und $x \in V$, so auch $(\forall x)\varphi$ und $(\exists x)\varphi$.

Ein **Modell** für diese Sprache ist ein Tripel $\langle M, \{f^{\mathfrak{M}} : f \in F\}, \{r^{\mathfrak{M}} : r \in R\}\rangle$ derart, daß $f^{\mathfrak{M}} : M^{\Omega(f)} \rightarrow M$ für jedes $f \in F$ und $r^{\mathfrak{M}} \subseteq M^{\Xi(r)}$ für jedes $r \in R$. Sei jetzt $\beta : V \rightarrow M$. Dann definieren wir $\langle \mathfrak{M}, \beta\rangle \models \varphi$ für eine Formel induktiv. Zunächst ordnen wir jedem Term t seinen Wert $[t]^\beta$ zu. Dazu ist

$$\begin{aligned} [x]^\beta &:= \beta(x) \\ [f(t_0, \ldots, t_{\Omega(f)-1}])^\beta &:= f^{\mathfrak{M}}([t_0]^\beta, \ldots, [t_{\Omega(f)-1}]^\beta) \end{aligned}$$

Jetzt gehen wir zu den Formeln über. (Dazu ist $\gamma \sim_x \beta$, $x \in V$, falls $\beta(y) \neq \gamma(y)$ nur dann, wenn $y = x$.)

$$\begin{array}{lcl}
\langle \mathfrak{M}, \beta\rangle \models s_0 \doteq s_1 & \Leftrightarrow & [s_0]^\beta = [s_1]^\beta \\
\langle \mathfrak{M}, \beta\rangle \models r(t_0, \ldots, t_{\Xi(r)}) & \Leftrightarrow & \langle [t_i] : i < \Xi(r)\rangle \in r^{\mathfrak{M}} \\
\langle \mathfrak{M}, \beta\rangle \models \neg\varphi & \Leftrightarrow & \langle \mathfrak{M}, \beta\rangle \not\models \varphi \\
\langle \mathfrak{M}, \beta\rangle \models \varphi \wedge \psi & \Leftrightarrow & \langle \mathfrak{M}, \beta\rangle \models \varphi \text{ und } \langle \mathfrak{M}, \beta\rangle \models \psi \\
\langle \mathfrak{M}, \beta\rangle \models \varphi \vee \psi & \Leftrightarrow & \langle \mathfrak{M}, \beta\rangle \models \varphi \text{ oder } \langle \mathfrak{M}, \beta\rangle \models \psi \\
\langle \mathfrak{M}, \beta\rangle \models \varphi \to \psi & \Leftrightarrow & \text{falls } \langle \mathfrak{M}, \beta\rangle \models \varphi \text{ so } \langle \mathfrak{M}, \beta\rangle \models \psi \\
\langle \mathfrak{M}, \beta\rangle \models (\exists x)\varphi & \Leftrightarrow & \text{existiert } \beta' \sim_x \beta : \langle \mathfrak{M}, \beta'\rangle \models \varphi \\
\langle \mathfrak{M}, \beta\rangle \models (\forall x)\varphi & \Leftrightarrow & \text{für alle } \beta' \sim_x \beta : \langle \mathfrak{M}, \beta'\rangle \models \varphi
\end{array}$$

Auf diese Weise werden also Formeln in Strukturen interpretiert. Zum Beispiel können wir unsere arithmetischen Terme mit $+$, 0 und $\times$ sowie der Relation $<$ in der Struktur $\mathbb{N}$ interpretieren, wo $+^{\mathbb{N}}$ und $\times^{\mathbb{N}}$ die üblichen Operationen $+$ und $\times$ sind, $0^{\mathbb{N}} = 0$ und $<^{\mathbb{N}} = <$. Dann gilt zum Beispiel für $\beta(z) = 7$:

$$(*) \qquad \langle \mathbb{N}, \beta\rangle \models (\forall x)(\forall y)(x \times y \doteq z \to x \doteq 1 \vee y \doteq 1)$$

Denn die Formel besagt nichts anderes, als daß $\beta(z)$ eine Primzahl ist. Denn eine Zahl w ist eine Primzahl, wenn für alle Zahlen u und v gilt: ist $u \times v = w$, so ist $u = 1$ oder $v = 1$. Wir vergleichen dies mit $(*)$. $(*)$ gilt, wenn für alle $\beta' \sim_x \beta$ gilt:

$$\langle \mathbb{N}, \beta'\rangle \models (\forall y)(x \times y \doteq z \to x \doteq 1 \vee y \doteq 1)$$

Dies wiederum ist der Fall, wenn für alle $\beta'' \sim_y \beta'$ gilt

$$\langle \mathbb{N}, \beta''\rangle \models x \times y \doteq z \to x \doteq 1 \vee y \doteq 1$$

Dies bedeutet: ist $u := \beta''(x)$, $v := \beta''(y)$ und $w := \beta''(z)$ und haben wir $w = u \times v$, so ist $u = 1$ oder $v = 1$. Dies gilt nun für alle u und v. Da andererseits $w = \beta(z)$, gilt also $(*)$ genau dann, wenn $\beta(z)$, also 7, Primzahl ist.

Der Leser möge sich überzeugen, daß für jedes β gilt

$$\langle \mathbb{N}, \beta\rangle \models (\forall z)(\exists u)(\forall x)(\forall y)(x \times y \doteq u \to x \doteq 1 \vee y \doteq 1)$$

Dieser Satz besagt, daß zu jeder Zahl eine noch größere Primzahl existiert. Für spätere Zwecke führen wir nun den Typ e ein. Dies ist der Typ der Terme. Diese haben Werte in M. Also wird e stets realisiert durch M.

Bevor wir uns daran machen können, eine Semantik für natürliche Sprache zu entwerfen, werden wir die Relationen aus der Prädikatenlogik eliminieren. Dazu werden wir einen neuen Basistyp schaffen, den wir t nennen. t wird durch die Menge der Wahrheitswerte realisiert, also $\{0, 1\}$. Eine n–stellige Relation r wird durch die

n–stellige Funktion $r^{\spadesuit}$ von Objekten nach Wahrheitswerten ersetzt, welche definiert ist durch

$$r^{\spadesuit}(x_0, x_1, \ldots, x_{\Xi(r)-1}) = 1 \text{ gdw } r(x_0, \ldots, x_{\Xi(r)-1}) \; .$$

Dies erlaubt uns, den λ–Kalkül zur expliziten Verwaltung der Argumentstellen von r einzusetzen. Zum Beispiel können wir bei einer 2–stelligen Relation r jetzt folgende Funktionen r_1 und r_2 definieren:

$$\begin{array}{lcl} r_1 & := & \lambda x_e.\lambda y_e.r^{\spadesuit}(x_e, y_e) \\ r_2 & := & \lambda x_e.\lambda y_e.r^{\spadesuit}(y_e, x_e) \end{array}$$

Dies erlaubt uns also, Funktionen zu definieren, deren erstes Argument wahlweise dem ersten Argument von $r^{\spadesuit}$ oder dem zweiten Argument von $r^{\spadesuit}$ entspricht.

Ferner werden wir $\neg$, $\wedge$, $\vee$ und $\rightarrow$ als Funktionen auffassen:

	$\neg$
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

Syntaktisch gesehen hat $\neg$ die Kategorie t/t und $\wedge$, $\vee$ und $\rightarrow$ die Kategorie $(t\backslash t)/t$. Schließlich werden auch die Quantoren noch als Funktionen aufgefaßt. Dazu definieren wir Funktionen Π und Σ vom Typ $e \rightarrow (t \rightarrow t)$. Es gilt

$$\begin{array}{lll} \Pi(x)(X) = 1 & \text{gdw} & \text{für alle Werte von } x : X(x) = 1 \\ \Sigma(x)(X) = 1 & \text{gdw} & \text{für einen Wert von } x : X(x) = 1 \end{array}$$

Als Interpretation von $(\forall x)\varphi$ wählen wir $\Pi(x)(\varphi)$, und als Interpretation von $(\exists x)\varphi$ wählen wir $\Sigma(x)(\varphi)$. Es ist also $(\forall x) = \lambda x.\Pi(x)(X)$ und $(\exists x) = \lambda x.\Sigma(x)(X)$. Wir werden allerdings weiterhin $\forall x.\varphi$ und $\exists x.\varphi$ schreiben.

Wir beginnen nun mit der Montague Semantik. Der Einfachheit halber wählen wir nur eine kleine Sprache. Beginnen wir mit

$$\{\texttt{Paul}, \texttt{Peter}, \texttt{schläft}, \texttt{sieht}\}$$

Der semantische Typ von `Paul` und `Peter` ist jeweils e, der semantische Typ von `schläft` ist $e \rightarrow t$, der Typ von `sieht` $e \rightarrow (e \rightarrow t)$. Dies bedeutet: Namen werden durch Individuen interpretiert, intransitive Verben durch einstellige Relationen und transitive Verben durch 2–stellige Relationen. Das (finite) Verb `schläft` wird durch die Relation $\mathsf{schläft}'$ interpretiert und `sieht` durch die Relation sieht'. Aufgrund unserer Vereinbahrung ist dann ein transitives Verb eine Funktion vom Typ $e \rightarrow (e \rightarrow t)$. Also ist die Bedeutung von diesen Verben in der Tat

$$\begin{array}{lcl} \texttt{schläft} & \mapsto & \lambda x_e.\mathsf{schläft}'(x_e) \\ \texttt{sieht} & \mapsto & \lambda x_e.\lambda y_e.\mathsf{sieht}'(y_e, x_e) \end{array}$$

Wir halten diese Konvention (von Montague eingeführt) durch und vereinbahren, daß der Wert von `Paul` gerade paul' und der Wert von `Peter` das Individuum peter' ist. Als letztes vereinbahren wir die syntaktischen Kategorien.

$$\begin{array}{lcl} \texttt{Paul} & : & e \\ \texttt{Peter} & : & e \\ \texttt{schläft} & : & e\backslash t \\ \texttt{sieht} & : & (e\backslash t)/e \end{array}$$

Dies also sind unsere Modi:

$$\begin{array}{lll} \langle \texttt{Paul}, & e, & \mathsf{paul}' \rangle \\ \langle \texttt{Peter}, & e, & \mathsf{peter}' \rangle \\ \langle \texttt{schläft}, & e\backslash t, & \lambda x_e.\mathsf{schläft}'(x_e) \rangle \\ \langle \texttt{sieht}, & (e\backslash t)/e, & \lambda x_e.\lambda y_e.\mathsf{sieht}'(y_e, x_e) \rangle \end{array}$$

Die Sätze `Peter schläft` oder `Peter sieht Peter` sind grammatisch, und ihre Bedeutung ist — wie man leicht bestätigt — $\mathsf{schläft}'(\mathsf{paul}')$ und $\mathsf{sieht}'(\mathsf{peter}', \mathsf{peter}')$.

Die syntaktischen Kategorien besitzen auch eine Entsprechung in der grammatischen Terminologie. e zum Beispiel ist die Kategorie der Eigennamen. Die Kategorie $e\backslash t$ ist die Kategorie der intransitiven Verben und die Kategorie $(e\backslash t)/e$ die Kategorie der transitiven Verben.

Diese Minisprache kann man zum Beispiel durch logische Junktoren erweitern. Zum Beispiel können wir das Wort `nicht` über folgendes Zeichen einführen:

$$\langle \texttt{nicht}, (e\backslash t)\backslash(e\backslash t), \lambda e_{e\to t}.\lambda x_e.\neg x_{e\to t}(x_e) \rangle$$

Der Leser ist aufgefordert nachzurechnen, daß nunmehr `schläft nicht` ein intransitives Verb ist, dessen Bedeutung gerade das Gegenteil von `schläft` ist. Also ist `Paul schläft nicht` wahr genau dann, wenn `Paul schläft` falsch ist. Ferner können wir das Wort `und` durch folgenden Modus einführen:

$$\langle \texttt{und}, ((e\backslash t)\backslash(e\backslash t))/(e\backslash t), \lambda x_{e\to t}.\lambda y_{e\to t}.\lambda z_e.x_{e\to t}(z_e) \wedge y_{e\to t}(z_e) \rangle$$

Auf diese Weise haben wir schon eine kleine Sprache definiert, welche unendlich viele grammatische Sätze erzeugt und ihnen die richtige Bedeutung zuweist. Allerdings wird man recht bald sehen, daß das Deutsche unermeßlich viel komplizierter ist als die gerade vorgestellte Sprache.

Das wirkliche Verdienst Montagues ist es aber, die Behandlung von Quantoren ermöglicht zu haben. Sehen wir uns an, wie dies gehen kann. Nomina wie `Katze` und `Maus` sind keine Eigennamen, sondern semantisch gesehen 1–stellige Prädikate. Denn `Katze` ist kein einzelner Gegenstand, vielmehr ist ein bestimmter Gegenstand eine Katze oder nicht. Getreu unseren obigen Definitionen setzen wir also für den

semantischen Typ von `Katze` und `Maus` $e \to t$. Syntaktisch entspricht das t/e oder $e\backslash t$. Hier ist keine Entscheidung möglich, denn weder ist `Katze Paul` noch `Paul Katze` grammatisch. Sicher sind sie, falls grammatisch, keine Aussagen. Montague hat dieses Problem nicht wirklich gelöst. Er hat einen neuen Typkonstruktor, //, eingeführt, mit dem man eine Kategorie $t//e$ bilden kann, welche sich syntaktisch von t/e (dem intransitiven Verb) unterscheidet. Ehrlicher wäre es gewesen, schlicht eine Kategorie n anzunehmen, und ihr den semantischen Typ $e \to t$ zu geben. Wir wollen letzteres tun. Nun sagen wir, der Subjektsquantor `jede` hat die syntaktische Kategorie $(t/(e\backslash t))/n$. Dies bedeutet: er formt mit einem Nomen eine Konstituente, welche die Kategorie $t/(e\backslash t)$ hat. Diese benötigt ihrerseits ein intransitives Verb, um letztlich einen Satz zu formen. Wir haben also folgendes Zeichen:

$$\langle \texttt{jede}, (t/(e\backslash t))/n, \lambda x_{e\to t}.\lambda y_{e\to t}.\forall x_e.(x_{e\to t}(x_e) \to y_{e\to t}(x_e))\rangle$$

Rechnen wir dies anhand des folgenden Satzes vor:

`Jede Katze sieht Peter`

Die syntaktische Analyse ist wie folgt:

$$\dfrac{\dfrac{\begin{array}{cc}\texttt{Jede} & \texttt{Katze}\\ (t/(e\backslash t))/n & n\end{array}}{t/(e\backslash t)} \qquad \dfrac{\begin{array}{cc}\texttt{sieht} & \texttt{Peter}\\ (e\backslash t)/e & e\end{array}}{e\backslash t}}{t}$$

Daraus ergibt sich also diese Konstituentstruktur.

$$((\texttt{Jede Katze}) \quad (\texttt{sieht Peter}))$$

Jetzt müssen wir nur noch die Bedeutungen einsetzen und 'ausrechnen'. Ausrechnen bedeutet hier, daß wir die Funktion auf ihr Argument anwenden. Denn laut Konvention wird eine Konstituente immer aus einer Funktion mit einem passenden Argument gebildet. Dafür sorgt die Korrespondenz zwischen syntaktischen und semantischen Typen. Wir rechnen in mehreren Schritten. Es ist `sieht Peter` eine Konstituente und ihre Bedeutung ist

$$(\lambda x_e.\lambda y_e.\mathsf{sieht}'(y_e, x_e))(\mathsf{peter}') = \lambda y_e.\mathsf{sieht}'(y_e, \mathsf{peter}')$$

Ferner ist `jede Katze` eine Konstituente mit folgender Bedeutung

$$\begin{array}{ll} & (\lambda x_{e\to t}.\lambda y_{e\to t}.(\forall x_e.x_{e\to t}(x_e) \to y_{e\to t}(x_e)))(\lambda x_e.\mathsf{katze}'(x_e))\\ = & \lambda y_{e\to t}.\forall x_e.((\lambda x_e.\mathsf{katze}'(x_e))(x_e) \to y_{e\to t}(x_e))\\ = & \lambda y_{e\to t}.\forall x_e.(\mathsf{katze}'(x_e) \to y_{e\to t}(x_e))\end{array}$$

Diese beiden kombinieren wir jetzt ihrerseits:

$$\begin{aligned} & (\lambda y_{e\to t}.\forall x_e.\mathsf{katze}'(x_e) \to y_{e\to t}(x_e))(\lambda y_e.\mathsf{sieht}'(y_e, \mathsf{peter}')) \\ = \;& \forall x_e.(\mathsf{katze}'(x_e) \to (\lambda y_e.\mathsf{sieht}'(y_e, \mathsf{peter}'))(x_e)) \\ = \;& \forall x_e.(\mathsf{katze}'(x_e) \to \mathsf{sieht}'(x_e, \mathsf{peter}')) \end{aligned}$$

Dies ist in der Tat das gewünschte Ergebnis. Ähnlich wie `jede` ist `eine` definiert:

$$\langle \texttt{eine}, (t/(e\backslash t))/n, \lambda x_{e\to t}.\lambda y_{e\to t}.\exists x_e.(x_{e\to t}(x_e) \wedge y_{e\to t}(x_e))\rangle$$

Falls wir auch in einem Akkusativobjekt einen Quantor unterbringen wollen, müssen wir folgende neuen Zeichen vereinbahren:

$$\langle \texttt{jede}, ((e\backslash t)/((e\backslash t)/e)/n, \lambda x_{e\to t}.\lambda y_{e\to(e\to t)}.\lambda y_e.\forall x_e.(x_{e\to t}(x_e) \to y_{e\to(e\to t)}(x_e)(y_e))\rangle$$
$$\langle \texttt{eine}, ((e\backslash t)/((e\backslash t)/e)/n, \lambda x_{e\to t}.\lambda y_{e\to(e\to t)}.\lambda y_e.\exists x_e.(x_{e\to t}(x_e) \to y_{e\to(e\to t)}(x_e)(y_e))\rangle$$

Denn `jede Katze` als Akkusativobjekt wird analysiert als eine Konstituente, welche ein transitives Verb zu einem intransitiven Verb macht. Dies muß also die Kategorie $(e\backslash t)/((e\backslash t)/e)$ haben. Daraus folgt unmittelbar die Kategorie des Quantors `jede`.

Gehen wir dies anhand des Beispiels

```
Eine Katze sieht jede Maus.
```

durch. Die Konstituentenstruktur ist wie folgt:

$$((\texttt{Eine Katze})\ (\texttt{sieht}\ (\texttt{jede Maus})))$$

Die Bedeutung von `jede Maus` ist, wie man jetzt leicht nachrechnet, diese:

$$\lambda y_{e\to(e\to t)}.\lambda y_e.\forall x_e(\mathsf{maus}'(x_e) \to y_{e\to(e\to t)}(x_e)(y_e))$$

Damit bekommen wir für `sieht jede Maus`

$$\begin{aligned} & \lambda y_e.\forall x_e(\mathsf{maus}'(x_e) \to (\lambda x_e.\lambda y_e.\mathsf{sieht}'(y_e, x_e))(x_e)(y_e)) \\ = \;& \lambda y_e.\forall x_e(\mathsf{maus}'(x_e) \to \mathsf{sieht}'(y_e, x_e)) \end{aligned}$$

`eine Katze` ist analog zu `jede Katze`:

$$\lambda y_{e\to t}.\exists x_e.(\mathsf{katze}'(x_e) \wedge y_{e\to t}(x_e))$$

Diese kombinieren wir jetzt:

$$\begin{aligned} & \lambda y_{e\to t}.\exists x_e.(\mathsf{katze}'(x_e) \wedge y_{e\to t}(x_e))(\lambda y_e.\forall x_e.(\mathsf{maus}'(x_e) \to \mathsf{sieht}'(y_e, x_e))) \\ = \;& \exists x_e.(\mathsf{katze}'(x_e) \wedge (\lambda y_e.\forall x_e.(\mathsf{maus}'(x_e) \to \mathsf{sieht}'(y_e, x_e)))(x_e)) \\ = \;& \exists x_e.(\mathsf{katze}'(x_e) \wedge \forall z_e.(\mathsf{maus}'(z_e) \to \mathsf{sieht}'(x_e, z_e))) \end{aligned}$$

An diesem Beispiel kann man sehen, daß das Ausrechnen von λ–Termen einiges Geschick erfordert. Gelegentlich kann es zu Variablenkollisionen kommen, und dann muß man Variablen umbenennen. Dies ist der Fall, wenn wir einen Term einsetzen, welcher freie Variable enthält, die dann gebunden werden, oder wenn wir einen Term einsetzen, dessen gebundene Variablen in dem größeren Term ebenfalls vorkommen.

3.7 Anhang: Grundzüge des λ–Kalküls und der Kombinatorischen Logik

Schon Frege hat bemerkt, daß ein Unterschied besteht zwischen einem Term und einer Funktion. Der *Term* $x^2 + 2xy$ ist etwas, welches einen konkreten Wert hat, wenn x und y einen konkreten Wert haben. Betrachten wir dagegen die Funktion $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} : \langle x, y\rangle \mapsto x^2 + 2xy$. Diese Funktion benötigt keine Werte für irgendwelche Variablen, sondern sie benötigt lediglich ein Paar konkreter Zahlen, damit sie einen Wert ergibt. Daß in der Definition von f auch Variablen auftauchen, ist eine notwendige Sache, aber ihr Name ist unerheblich. Wir hätten diese Funktion auch so beschreiben können $f : \langle x, z\rangle \mapsto x^2 + 2xz$. Nicht so beim Term $x^2 + 2xy$. Dieser ist verschieden vom Term x^2+2xz. Denn wenn, sagen wir, x durch 3, y durch 4 und z durch 2 ersetzt wird, so bekommen wir für den ersten Term $3^2 + 2 \cdot 3 \cdot 4 = 33$, und für den zweiten Term $3^2 + 2 \cdot 3 \cdot 2 = 21$. Um diesem Unterschied Rechnung zu tragen, wurde der λ–Kalkül entwickelt. Der λ–Kalkül erlaubt es, aus beliebigen Termen Funktionen zu definieren. Im vorliegenden Fall schreibt man die Funktion f als

$$\lambda xy.x^2 + 2xy$$

Dieser Ausdruck benennt die Funktion f und sagt gleichzeitig, was sie leistet. Das Präfix 'λxy' besagt, daß wir es mit einer Funktion von Paaren $\langle x, y\rangle$ zu tun haben, und daß diese Funktion jedem solchen Paar den Wert $x^2 + 2xy$ zuordnet. Dies ist also genau dasselbe, was $\langle x, y\rangle \mapsto x^2 + 2xy$ ausdrückt. Nun können wir aber auch folgende Funktionen definieren:

$$\lambda x.\lambda y.x^2 + 2xy, \qquad \lambda y.\lambda x.x^2 + 2xy$$

Die erste ist eine Funktion, welche zu jeder Zahl m Funktion $\lambda y.m^2 + 2my$ auswirft; diese wiederum ergibt für jedes n den Wert $m^2 + 2mn$. Die zweite ist eine Funktion, welche zu jeder Zahl m die Funktion $\lambda x.x^2 + 2xm$ auswirft; diese ergibt für n den Wert $n^2 + 2nm$. Da im allgemeinen $m^2 + 2mn \neq n^2 + 2nm$, so sind diese Funktionen verschieden.

Im λ–Kalkül verzichtet man üblicherweise auf die gleichzeitige Abstraktion, das heißt, man erlaubt nur Präfixe der Form 'λx', nicht aber solche der Form 'λxy'. Dies wollen wir jetzt auch tun (wobei wir uns augenblicklich der Möglichkeit berauben, über den Lambek–Kalkül zu reden). Wir wollen nun die allgemeine Definition von λ–Termen angeben. Das Alphabet besteht außer aus F, λ, den Variablen $\{x_i : i \in \omega\}$ noch aus den Klammern `(`, `)` und dem Punkt `.`.

Definition 3.7.1 *Die Menge der* λ–***Terme über einer Signatur*** Ω*, kurz die Menge der* λ–Ω–***Terme****, ist die kleinste Menge* M*, für die gilt:*

1. *Jeder Ω–Term ist in M.*
2. *Sind $X, Y \in M$, so auch $(XY) \in N$.*
3. *Ist $X \in M$ und x eine Variable, so ist $(\lambda x\,.\,X) \in M$.*

*Geht die Signatur aus dem Kontext hervor, so sprechen wir auch schlicht von der Menge der λ–**Terme**.*

Da der Operator nicht geschrieben wird, existiert keine Polnische Notation. Deswegen verwenden wir noch Klammern (und). Streng genommen müssen wir auch noch die Menge V der Variablen festlegen. Für unsere Zwecke enthält V gewisse Symbole, mindestens jedoch x, y, z und u. Wir vereinbahren hier nun, daß x eine Metavariable für Elemente von V ist, während x eine konkrete Variable ist, also $\mathtt{x} \in V$. Ferner sind X, Y Metavariable für λ–Terme. In den Büchern setzt man meist $F := \varnothing$. Das Interessante an dem λ–Kalkül sind ja nicht die Grundfunktionen, sondern die Mechanik der Funktionalabstraktion. Falls also $F = \varnothing$, so sprechen wir auch von **reinen λ–Termen**. Üblicherweise läßt man Klammern weg, wenn der Term linksgeklammert ist. So ist $WXYZ$ kurz für $(((WX)Y)Z)$ und $\lambda x.XY$ kurz für $((\lambda x\,.\,X)Y)$. Dabei muß man allerdings Vorsicht walten lassen, weil die Klammern wirkliche Symbole der Sprache sind. Insofern ist xxx keine Zeichenkette, sondern nur ein Kürzel für ((xx)x), was wir jedoch nach einiger Zeit vernachlässigen werden. Man beachte, daß (xx) ein Term ist. Dieser bezeichnet die Anwendung von x auf sich selbst. *Vorkommen* einer Zeichenkette $\vec{x}$ in einer Zeichenkette $\vec{y}$ haben wir als Kontexte $\langle \vec{u}, \vec{v} \rangle$ definiert, für die $\vec{u}\vec{x}\vec{v} = \vec{y}$ ist. Ω–Terme werden in Polnischer Notation notiert gedacht.

Definition 3.7.2 *Es sei x eine Variable. Wir definieren die Menge der **Vorkommen** von x in λ–Termen induktiv wie folgt.*

1. *Ist X ein Ω–Term t, so ist die Menge der Vorkommen von x in X gerade die Menge der Vorkommen von $\vec{x}$ in t.*
2. *Die Menge der Vorkommen von x in (XY) ist die Vereinigung aus der Menge der Paare $\langle (\vec{u}, \vec{v}Y) \rangle$, wo $\langle \vec{u}, \vec{v} \rangle$ ein Vorkommen von x in X ist und der Menge der Paare $\langle (X\vec{u}, \vec{v}) \rangle$, wo $\langle \vec{u}, \vec{v} \rangle$ ein Vorkommen von x in Y ist.*
3. *Die Menge der Vorkommen von x in $(\lambda x\,.\,X)$ ist die Menge aller $\langle (\lambda x\,.\,\vec{u}, \vec{v}) \rangle$, wo $\langle \vec{u}, \vec{v} \rangle$ ein Vorkommen von x in X ist.*

Man beachte, daß — technisch gesehen — das Auftreten von x in dem λ–Präfix nicht als Vorkommen gilt. Also kommt x in $(\lambda$x.y) nicht vor!

Definition 3.7.3 *Es sei X ein λ–Term und x eine Variable. x kommt an einer Stelle in X* ***frei*** *vor, falls es nicht in einem Term der Form $\lambda x.Y$ für irgendein Y vorkommt. Falls x an dieser Stelle nicht frei vorkommt, so kommt es dort* ***gebunden*** *vor. Ein λ–Term heißt* ***geschlossen****, falls keine Variable frei auftritt. Die Menge der in X frei vorkommenden Variablen bezeichnen wir mit $f(X)$.*

Ein paar Beispiele mögen dies erläutern. In $X = (\lambda \mathtt{x}.\mathtt{xy})$ tritt `x` stets gebunden auf, da es immer in einem Teilterm der Form $(\lambda \mathtt{x}.Y)$ auftritt (etwa $Y := \mathtt{xy}$). Dagegen kommt `y` frei vor. Es ist für eine Variable möglich, in einem Term sowohl gebunden wie auch frei aufzutreten, zum Beispiel `x` in $\mathtt{x}(\lambda \mathtt{x}.\mathtt{x})$.

Gebundene und freie Vorkommen verhalten sich unterschiedlich bei Ersetzung. Ist X ein λ–Term und x eine Variable, so bezeichne $[N/x]X$ das Resultat der Ersetzung von x durch N. Bei der Ersetzung wird allerdings nicht jedes Vorkommen von x kurzerhand durch ein Vorkommen von N ersetzt; die Definition der Ersetzung erfordert einige Vorsicht. Deswgen ist $[N/x]X$ verschieden von $X[x \mapsto N]$, welches das Ergebnis der Ersetzung jedes Vorkommens von x in X durch N bezeichnet.

(a)	$[N/x]t$	$:=$	$t[x \mapsto N]$	t ein Ω–Term
(b)	$[N/x](XY)$	$:=$	$([N/x]X)([N/x]Y))$	
(c)	$[N/x](\lambda x.Y)$	$:=$	$(\lambda x.Y)$	
(d)	$[N/x](\lambda y.Y)$	$:=$	$(\lambda y.[N/x]Y)$	falls $y \neq x$, $y \notin f(N)$ oder $x \notin f(Y)$
(e)	$[N/x](\lambda y.Y)$	$:=$	$(\lambda z.[N/x][z/y]Y)$	falls $y \neq x$, $y \in f(N)$ und $x \in f(Y)$

In (e) muß z so gewählt sein, daß es nicht frei in N oder Y auftritt. Damit die Substitution eindeutig bestimmt ist, nehmen wir an, die Variablenmenge sei wohlgeordnet, und dann wählen wir z bezüglich dieser Wohlordnung minimal. Die Vorsichtmaßnahme der zusätzlichen Substitution in (e) ist notwendig. Sei nämlich $Y = \mathtt{x}$. Dann wäre ohne diese Maßregel (das heißt mit (d)),

$$[\mathtt{y}/\mathtt{x}](\lambda \mathtt{x}.\mathtt{y}) = (\lambda \mathtt{y}.[\mathtt{y}/\mathtt{x}]\mathtt{x}) = (\lambda \mathtt{y}.\mathtt{y})$$

Dies widerspricht aber unserer Intuition. Denn $(\lambda \mathtt{y}.\mathtt{x})$ ist die Funktion, welche zu gegebenem Wert für `y` den Wert von `x` auswirft. Aber $(\lambda \mathtt{y}.\mathtt{y})$ ist die Identitätsfunktion und somit davon verschieden. Nun soll aber die Substitution einer Variablen durch eine andere Variable soll nicht den Wertverlauf einer Funktion ändern.

Definition 3.7.4 *Es seien X und Y λ–Terme. Wir sagen, Y* ***gehe aus*** *X* ***durch Umbenennung gebundener Variablen hervor****, falls es einen Teilterm $\lambda y.Z$ von X und eine Variable z, welche nicht in Z vorkommt, sodaß Y das Resultat des Ersetzens eines Vorkommens von $\lambda y.Z$ durch $\lambda v.[v/y]Z$ ist. Y ist* ***kongruent zu*** *X, falls Y aus X durch eine endliche Serie gebundener Umbenennungen hervorgeht.*

Nun zur Definition von β–Konversion.

Definition 3.7.5 *Es sei X ein λ–Term. Wir schreiben $X \rightsquigarrow_\beta Y$ und sagen, X **kontrahiert zu** Y, falls Y das Resultat einer einmaligen Ersetzung eines Vorkommens von $\lambda x.ZU$ in X durch $[U/x]Z$ ist. Ferner schreiben wir $X \triangleright Y$, falls Y aus X durch eine endliche Serie von Umbenennungen oder Kontraktionen aus X hervorgeht, und $X \equiv Y$, falls $X \triangleright Y$ und $Y \triangleright X$.*

Ein Term der Form $((\lambda x.X)Y)$ heißt **Redex** und $[Y/x]X$ sein **Kontraktum**. Der Übergang von Redex zu Kontraktum repräsentiert hier das Auswerten einer Funktion auf ihr Argument. Insofern ist ein λ–Term *ausgerechnet*, wenn er kein Redex enthält.

Definition 3.7.6 *Ein λ–Term heißt in **Normalform**, falls er kein Redex enthält. Ist $X \triangleright Y$ und Y in Normalform, so heißt Y eine **Normalform von** X.*

Ohne Beweis teilen wir hier den folgenden wichtigen Satz mit.

Theorem 3.7.7 (Church, Rosser) *Es sei X ein λ–Term und $X \triangleright U, V$. Dann existiert ein Y mit $U \triangleright Y$ und $V \triangleright Y$.*

Den Beweis kann man in Büchern zum λ–Kalkül finden.

Korollar 3.7.8 *Es seien Y und Z Normalformen von X. Dann sind Y und Z kongruent.*

Der Beweis ist einfach. Denn nach dem obigen Satz existiert ein U mit $Y \triangleright U$ und $Z \triangleright U$. Da aber sowohl Y als auch Z kein Redex mehr enthalten, und Umbenennungen keine Redexe einführen, so kann U aus Y sowie Z nur durch Umbenennungen hervorgehen. Also ist U mit Y und Z kongruent, und deshalb Y und Z.

Es ist nicht immer der Fall, daß ein λ–Term eine Normalforn hat. Man bekommt zum Beispiel

$$((\lambda x.(xx))(\lambda x.(xx))) \triangleright ((\lambda x.(xx))(\lambda x.(xx)))$$

Oder auch

$$\begin{aligned} & ((\lambda x.((xx)y))(\lambda x.((xx)y))) \\ \triangleright\ & (((\lambda x.((xx)y))(\lambda x.((xx)y)))y) \\ \triangleright\ & ((((\lambda x.((xx)y))(\lambda x.((xx)y)))y)y) \end{aligned}$$

Der getypte λ–Kalkül unterscheidet sich von dem eben vorgestellten durch eine wesentliche Einschränkung.

Definition 3.7.9 *Es sei B eine Menge. Die Menge der* ***Typen*** *über B, $Typ_{\rightarrow}(B)$, ist die kleinste Menge M, für die gilt:*

1. $B \subseteq M$.
2. *Ist* $\alpha \in M$ *und* $\beta \in M$, *so* $\alpha \rightarrow \beta \in M$.

Mit anderen Worten: Typen sind nichts anderes als Terme der Signatur $\{\rightarrow\}$ mit $\Omega(\rightarrow) = 2$ über einer Menge von Basistypen. Es wird jedem Term ein Typ zugewiesen, und der Termaufbau wird durch die Typenzuweisung eingeschränkt. Es werden weiter alle Ω–Terme zugelassen. Ihr Typ steht bereits fest. Es gelten folgende Regeln.

1. Ist (XY) Term vom Typ γ, so existiert ein Typ α derart, daß X den Typ $\alpha \rightarrow \gamma$ hat und Y den Typ γ.
2. Hat X den Typ γ und ist x_α eine Variable vom Typ α, so ist $(\lambda x_\alpha . X)$ vom Typ $\alpha \rightarrow \gamma$.

Man beachte, daß nun für jeden Typ α Variablen vom Typ α bereitstehen, und zwar abzählbar unendlich viele. Diese bezeichnen wir durch x_α, y_α und so weiter. Ist $\alpha \neq \beta$, so ist auch $\mathtt{x}_\alpha \neq \mathtt{x}_\beta$. Mit diesen Bedingungen ist die Bildung von λ–Termen erheblich eingeschränkt. Zum Beispiel ist `(λx.(xx))` kein Term, ganz gleich, welchen Typ `x` hat. Man kann zeigen, daß ein getypter Term immer eine Normalform hat. Wir wollen dies hier jedoch nicht tun. Man beachte im Übrigen, daß wenn der Term `x + y` den Typ α hat und `x` und `y` ebenfalls den Typ α, so hat die Funktion `(λx.λy.x+y))` den Typ $\alpha \rightarrow (\alpha \rightarrow \alpha)$. Der Typ des Ω–Terms entspricht nämlich dem seines Werts und ist deswegen α.

Wir wollen nun noch eine Interpretation des λ–Kalküls vorstellen, welche wir in diesem Kapitel benötigen. Im allgemeinen ist ein *Modell* des λ–Kalküls eine partielle Algebra $\langle F, \Lambda, \bullet \rangle$, wo Λ eine totale Funktion namens Abstraktion ist und $\bullet$ eine partielle Funktion von F^2 nach F, die *Applikation*. Obwohl man bei der Abstraktion von Termen durchaus wissen muß, welche Variable (= Argumentstelle) man abstrahiert, ist dies bei geschlossenen Termen nicht nötig. Man erhält stets die gleiche Funktion. Wir interessieren uns nun für Modelle, bei denen F eine Menge von Mengen ist. Dazu werden jedem Typ Mengen zugeordnet, welche die möglichen Interpretationen der Elemente dieses Typs bestimmen. Wird also dem Typ α die Menge M zugeordnet, so darf eine Variable vom Typ α jeden Wert in M annehmen. Es folgt also, daß wenn β die Menge N zugeordnet wird, und X den Typ β hat, so ist die Interpretation von $(\lambda \mathtt{x}_\alpha . X)$ eine Funktion von M nach N. Wir setzen also die Interpretation von $\alpha \rightarrow \beta$ fest als die Menge aller Funktionen von M nach N. Dies ist eine willkürliche Wahl, eine andere Menge (zum Beispiel eine geeignete Teilmenge) würde sicherlich auch genügen.

Es seien M und N Mengen. Dann ist eine Funktion von M nach N eine Teilmenge F des kartesischen Produkts $M \times N$, welche gewissen Bedingungen genügt (siehe Abschnitt 1.1). So muß zu jedem $x \in M$ ein $y \in N$ existieren mit $\langle x, y\rangle \in F$, und falls $\langle x, y\rangle \in F$ und $\langle x, y'\rangle \in F$, so muß $y = y'$ sein. (Für partielle Funktionen entfällt die erste Bedingung. Alles andere ist analog. Wir wollen der Einfachheit halber uns nur mit Funktionen befassen.) Normalerweise stellt man sich eine Funktion als etwas vor, das Werte für gewisse Eingaben liefert. Dies ist hier nicht so. F ist keine Funktion in diesem Sinne, sondern lediglich der **Graph** einer Funktion. In der Mengenlehre unterscheidet man üblicherweise nicht zwischen einer Funktion und ihrem Graphen. Wir kommen darauf noch zurück. Wie hat man sich nun F als Menge vorzustellen? Dazu muß man zunächst einmal $\langle x, y\rangle$ definieren. Eine auf Kuratowski und Wiener zurückgehende Definition ist die folgende.

$$\langle x, y\rangle := \{x, \{x, y\}\}$$

Man überzeugt sich leicht, daß mit dieser Definition $\langle x, y\rangle = \langle u, v\rangle$ genau dann, wenn $x = u$ und $y = v$. Denn zunächst einmal ist sicher $\langle x, y\rangle = \langle u, v\rangle$, wenn diese Bedingung erfüllt ist. Auf der anderen Seite ist, wenn $\langle x, y\rangle = \langle u, v\rangle$ sicher $x = u$ oder $x = \{u, v\}$ und es ist $\{x, y\} = u$ oder $\{x, y\} = \{u, v\}$. Nehmen wir an, daß $x = u$. Dann ist $u \neq \{x, y\}$, da sonst $x \in u$ folgt, im Widerspruch zu der Fundiertheit von Mengen. Also haben wir $\{x, y\} = \{u, v\}$. Wir wissen schon, daß $x = u$ gilt. Dann muß sicher $y = v$ sein. Nun nehmen wir an, daß $x = \{u, v\}$. Dann ist $\{x, y\} = \{\{u, v\}, v\}$. Es ist aber weder $\{\{u, v\}, v\} = u$ noch $\{\{u, v\}, v\} = \{u, v\}$, wiederum wegen der Fundiertheit. Also tritt der Fall $x = \{u, v\}$ nicht ein, und wir haben $x = u$ und $y = v$, wie versprochen.

Jetzt setzen wir einfach

$$M \times N := \{\langle x, y\rangle : x \in M, y \in N\}$$

Dies ist eine Menge, wenn M und N Mengen sind. Man beachte, daß $M \times (N \times O) \neq (M \times N) \times O$. Denn $M \times (N \times O)$ besteht aus allen Paaren der Form $\langle x, \langle y, z\rangle\rangle$, wo $x \in M$, $y \in N$ und $z \in O$, während $(M \times N) \times O$ aus allen Paaren der Form $\langle\langle x, y\rangle, z\rangle$ besteht. Ist nämlich $\langle x, \langle y, z\rangle\rangle = \langle\langle x', y'\rangle, z'\rangle$, so müssen wir haben $x = \langle x', y'\rangle$ und $\langle y, z\rangle = z'$. Daraus folgt $x' \in x$. M besitzt ein bezüglich $\in$ minimales Element x^* (Fundiertheit). Setzen wir dieses für x, so haben wir einen Widerspruch. Dies zeigt $M \times (N \times O) \neq (M \times N) \times O$. Allerdings ist die Abbildung

$$\rtimes\ : \langle x, \langle y, z\rangle\rangle \mapsto \langle\langle x, y\rangle, z\rangle : M \times (N \times O) \to (M \times N) \times O$$

eine Bijektion. Ihre Umkehrung ist die Abbildung

$$\ltimes\ : \langle\langle x, y\rangle, z\rangle \mapsto \langle x, \langle y, z\rangle\rangle : (M \times N) \times O \to M \times (N \times O)$$

Als letztes setzen wir

$$M \to N := \{F \subseteq M \times N : F \text{ Funktion } \}\ .$$

Dies ist eine etwas andere Notation, als wir sie bisher benutzt haben, aber sie ist in diesem Zusammenhang vorteilhafter. Nun sind also Funktionen Mengen, und ihre Argumente auch. Deswegen benötigen wir noch eine Abbildung, welche eine Funktion auf ein Argument anwendet. Da sie auf allen Mengen erklärt werden muß, ist sie notwendig partiell. Ist nun x eine Funktion und y ein Element, so definieren wir $\mathsf{app}(x, y)$ wie folgt:

$$\mathsf{app}(x,y) := \begin{cases} z & \text{falls } \langle y, z\rangle \in x\ , \\ \star & \text{falls kein } z \text{ existiert mit } \langle y, z\rangle \in x\ . \end{cases}$$

app ist eine wirkliche Funktion (allerdings ist sie eine partielle Funktion). Ihr Graph auf dem Mengenuniversum ist eine echte Klasse, keine Menge. Es ist eben die Klasse aller Paare $\langle\langle F, x\rangle, y\rangle$, wo F eine Funktion ist und $\langle x, y\rangle \in F$.

Ist nun $F \in M \to (N \to O)$, so ist $F \subseteq M \times (N \to O) \subseteq M \times (N \times O)$. Dann ist $\rtimes[F] \subseteq (M \times N) \times O$, und man rechnet leicht nach, daß sogar $\rtimes[F] \subseteq (M \times N) \to O$. Auf diese Weise wird aus einer einstelligen Funktion mit Werten in $N \to O$ eine zweistellige Funktion von $M \times N$ nach O. Umgekehrt kann man sehen, daß wenn $F \in (M \times N) \to O$, so ist $\ltimes[F] \in M \to (N \to O)$.

In der Kategorialgrammatik, um die es in diesem Kapitel geht, werden wir λ–Terme einsetzen, um Bedeutungen von Symbolen und Zeichenketten zu benennen. Wichtig ist aber, daß der λ–Term nur ein formales Gebilde ist (nämlich eine bestimmte Zeichenkette), und nicht die eigentliche Bedeutung. Zum Beispiel ist `(λx.(λy.x+y))` eine Zeichenkette, welche eine Funktion benennt. Diese Funktion ist in der Mengeninterpretation eine Teilmenge von, sagen wir, $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$. Deswegen muß man zwischen der Gleichheit $=$ und dem Symbol $\equiv$ strikt unterscheiden. $X = Y$ bedeutet, daß wir es mit denselben Zeichenketten (also denselben λ–Termen) zu tun haben, während $X \equiv Y$ bedeutet, daß X und Y stets dieselbe Funktion bezeichnen. In diesem Sinne ist also `(λx.(λy.x+y))(x)(z)` $\neq$ `x+z`, aber sie bezeichnen stets und immer denselben Wert. Trotzdem wollen wir im Folgenden zwischen dem λ–Term und seiner Interpretation außerhalb dieses Abschnitts *nicht* unterscheiden, um die Notation nicht zu sehr aufzublähen.

Kommen wir also zu dem getypten λ–Kalkül zurück. Wir hatten im ungetypten Fall auch noch unsere Signatur gehabt. Man muß im getypten Kalkül jetzt anstelle der üblichen Signatur eine andere Signatur wählen. Jetzt muß man zu jedem Funktionssymbol f nicht alleine die Stelligkeit, sondern auch den Typ der Argumente angeben. Anstelle dessen kann man allerdings auch f schlicht als eine primitive Funktion (= Konstante) des getypten λ–Kalküls aufnehmen. Damit wird die Signatur ganz in dem λ–Kalkül aufgelöst. Dies hat den Vorzug, daß wir sie nicht extra behandeln müssen. Eine *Interpretation* unseres Kalküls wollen wir jetzt beschreiben. Angenommen, wir haben eine Menge B von Basistypen. Dann sei zunächst eine Abbildung $\ulcorner - \urcorner$ gewählt derart, daß $\ulcorner \alpha \urcorner$ eine Menge ist. Diese dehnen wir jetzt auf die

Typen wir folgt aus: $\ulcorner \alpha \to \beta \urcorner := \ulcorner \alpha \urcorner \to \ulcorner \beta \urcorner$. [1] Ferner wählen wir für jede Basisfunktion f vom Typ α ein Element in $\ulcorner \alpha \urcorner$, welches wir auch mit $\ulcorner f \urcorner$ bezeichnen. Nun setzen wir diese Zuweisung auf alle geschlossenen λ–Terme fort. Dabei gelten folgende Regeln:

$$\begin{array}{lcl} \ulcorner MN \urcorner & := & \mathsf{app}(\ulcorner M \urcorner, \ulcorner N \urcorner) \\ \ulcorner (\lambda x.Nx) \urcorner(x) & := & \ulcorner N \urcorner \end{array}$$

Auf diese Weise wird tatsächlich jedem geschlossenen Term eine Funktion zugeordnet, sodaß gilt: ist $M \triangleright N$, so $\ulcorner M \urcorner = \ulcorner N \urcorner$.

Der λ–Kalkül hat noch einen schwerwiegenden Nachteil, nämlich den, daß die Verwaltung von Variablen einige Vorsicht erfordert und keineswegs banal ist. Die Benennung von Variablen läßt sich interessanterweise aber vermeiden, wenn man auf freie Variable überhaupt verzichtet. Dies bringt uns auf die folgende Definition.

Definition 3.7.10 *Ein **Kombinator** ist ein geschlossener, reiner λ–Term.*

Die folgenden Kombinatoren sind von fundamentaler Bedeutung:

$$\begin{array}{lcl} \mathbf{I} & := & (\lambda \mathtt{x.x}) \\ \mathbf{K} & := & (\lambda \mathtt{x.}(\lambda \mathtt{y.x})) \\ \mathbf{S} & := & (\lambda \mathtt{x.}(\lambda \mathtt{y.}(\lambda \mathtt{z.xz(yz)}))) \end{array}$$

Der Witz ist nun, daß man in der Kombinatorischen Logik nicht mehr λ–Terme notiert, sondern nur noch Kombinatoren. Man definiert nun wie folgt. Ein **kombinatorischer Term** ist ein Term aufgebaut aus Variablen und Kombinatoren. Ferner wird die Redexrelation $\triangleright$ axiomatisch wie folgt eingeführt.

$$\begin{array}{lcl} \mathbf{I}X & \triangleright & X \\ \mathbf{K}XY & \triangleright & X \\ \mathbf{S}XYZ & \triangleright & XY(XZ) \\ X & \triangleright & X \end{array}$$

Ferner gelten folgende Regeln:

1. Ist $X \triangleright Y$ und $Y \triangleright Z$, so $X \triangleright Z$.
2. Ist $X \triangleright Y$, so $XZ \triangleright YZ$.
3. Ist $X \triangleright Y$, so $ZX \triangleright ZY$.

[1] Dies wird gewisse Schwierigkeiten bergen, wenn man nicht garantieren kann, daß die Realisierungen von verschiedenen Typen disjunkt ist. Dies kann durchaus vorkommen, zum Beispiel, wenn $\ulcorner \alpha \urcorner := \{\varnothing, \{\varnothing, \{\varnothing, \varnothing\}\}\}$. Dies kann allerdings ein erwünschter Effekt sein, obwohl es für unsere Zwecke stört. Wir werden auf diese Schwierigkeiten jedoch nicht näher eingehen.

Nun kann man umgekehrt $\lambda x.N$ aus N definieren. Setze

1. $[x]x := \mathbf{I}$.
2. $[x]M := \mathbf{K}M$, falls x nicht frei in M auftritt.
3. $[x]Ux := U$, falls x nicht in U frei auftritt.
4. $[x]UV := \mathbf{S}([x]U)([x]V)$, sonst.

Zum Beispiel ist $[\mathtt{y}]\mathtt{yx} = \mathbf{S}([\mathtt{y}]\mathtt{y})([\mathtt{y}]\mathtt{x}) = \mathbf{SI(Kx)}$. In der Tat, wendet man dies auf $\mathtt{y}$ an, so erhält man

$$\mathbf{SI(Kx)}\mathtt{y} \triangleright \mathbf{I}\mathtt{y}\mathbf{(Kx}\mathtt{y}\mathbf{)} \triangleright \mathtt{y}\mathbf{(Kx}\mathtt{y}\mathbf{)} \triangleright \mathtt{yx}$$

Ferner ist dann

$$\mathbf{U} := [\mathtt{y}]\mathtt{(}[\mathtt{x}]\mathtt{yx)} = [\mathtt{y}]\mathbf{SI(Kx)} = \mathbf{S(K(SI))K}$$

Der Leser möge verifizieren, daß $\mathbf{U}\mathtt{xy} \triangleright \mathtt{yx}$. Auf diese Weise kann man also aus der Spezifikation eines Kombinators ganz leicht den Kombinator explizit berechnen. Es folgt dann

Theorem 3.7.11 *Es sei C ein Kombinator. Dann existiert ein Kombinator H, aufgebaut aus **I**, **K** und **S**, mit $C \equiv H$.*

Auch Kombinatoren kann man typen, und die Regeln für die Typzuweisung lassen sich leicht aus den Definitionen der Kombinatoren ablesen. Zum Beispiel hat **I** den Typ $\alpha \to \alpha$, sofern α der Typ von $\mathtt{x}$ ist. Wir gehen nun davon aus, daß ein ungetypter Kombinator alle Typen besitzt, welche mit den Typregeln konform sind. Also hat das Symbol **I** alle Typen der Form $\alpha \to \alpha$, denn wegen $N = \mathbf{I}N$ muß $\mathbf{I}\mathtt{x}$ den Typ α haben, wenn $\mathtt{x}$ den Typ α hat. Dies kann nur sein, wenn **I** den Typ $\alpha \to \alpha$ hat. Genauso hat **K** alle Typen der Form $\alpha \to (\beta \to \alpha)$, sowie **S** alle Typen der Form $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$. Im Rahmen des getypten Kalküls sind also **I**, **K** und **S** schematische Symbole. Man definiert nun für Kombinatoren Typen induktiv wie folgt: ist X vom Typ $\alpha \to \beta$ und Y vom Typ β, so ist (XY) wohlgeformt und hat den Typ β.

Übung 88. Es sei **B** definiert durch $\mathbf{B}\mathtt{xyz} \equiv \mathtt{x(yz)}$ und **C** durch $\mathbf{C}\mathtt{xyz} \equiv \mathtt{xzy}$. Stellen Sie **B** und **C** durch **S**, **K** und **I** dar.

Übung 89. Bestimmen Sie die Typen von **B** und **C** aus der vorigen Aufgabe.

Übung 90. Zeigen Sie, daß der zu **S** gehörende getypte λ–Term genau dann wohlgeformt ist, wenn **S** den Typ $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$ hat für gewisse α, β und γ.

Kapitel 4

PTIME Sprachen

4.1 Mild–Kontextsensitive Sprachen

Der Begriff 'mild kontextsensitiv' wurde in den 80er Jahren von Aravind Joshi eingeführt (siehe [28]). Diese sind wie folgt characterisiert:

1. Kontextfreie Sprachen sind mild kontextsensitiv. Es gibt mild kontextsensitive aber nicht kontextfreie Sprachen.
2. Mild kontextsensitive Sprachen können in polynomieller Zeit erkannt werden.
3. Es gibt nur eine beschränkte Möglichkeit, überkreuzende Abhängigkeiten zu erzeugen.
4. Mild kontextsensitive Sprachen haben die beschränkte Wachstumseigenschaft. (S hat die **beschränkte Wachstumseigenschaft**, falls $\{|\vec{x}| : x \in S\}$ eine lineare Menge enthält.)

Diese Bedingungen sind keinesfalls sehr stark, außer der dritten. Diese sorgt dafür, daß mild kontextsensitive Sprachen eine echte Teilklasse der kontextsensitiven Sprachen sind. Die erste Bedingung muß man nicht kommentieren. Die dritte Bedingung ist sehr schwach; sie sagt lediglich, daß es eine Zahl c_S gibt derart, daß für jedes $\vec{x} \in S$ ein $\vec{y} \in S$ existiert mit $|\vec{y}| \leq |\vec{x}| + c_S$. Man hat versucht, dies so einzuschränken, daß jeder Schnitt mit einem Alphabet, oder gar jeder Schnitt mit semilinearen Mengen auch diese Eigenschaft haben muß. Diese Bedingungen sind jedoch kaum anschaulich zu motivieren. Ähnlich problematisch ist die zweite Bedingung: was ist eine überkreuzende Abhängigkeit? Wir werden in diesem Kapitel Grammatiken studieren, bei welchen man einen Strukturbegriff wie bei kontextfreien Sprachen definieren kann.

Konstituenten sind dann gewisse Mengen disjunkter Teilwörter. Nimmt man dies als allgemeine Definition, so kann man die zweite Bedingung so deuten: es gibt eine Zahl n derart, daß eine Konstituente höchstens n Teile hat. Dies ist gewiß nicht die von Joshi intendierte Interpretation, aber es fällt schwer, den Begriff einer *Art* von Abhängigkeit zu definieren.

Die Bedingungen sind also bis auf die dritte relativ problematisch. Wir wollen deswegen auch in diesem Kapitel solche Sprachen studieren, welche einzig dieser Bedingung genügen. Um der Begriffsverwirrung keinen Vorschub zu leisten, wollen wir sie **PTIME** Sprachen nennen. Es sei M eine beliebige Menge und $S \in M$. Wir nennen $f : M \to \{0,1\}$ die **charakteristische Funktion** von S, falls $f(x) = 1$ genau dann, wenn $x \in S$. Wir bezeichnen f auch mit χ_S.

Definition 4.1.1 *Es sei A ein endliches Alphabet und $S \subseteq A^*$. Dann sei $S \in$ **PTIME** genau dann, wenn die charakteristische Funktion $\chi_S : A^* \to \{0,1\}$ durch eine deterministische Turingmaschine in polynomieller Zeit berechenbar ist.*

Im allgemeinen wollen wir auch sagen, eine Funktion $f : A^* \to B^*$ sei in **PTIME**, falls es eine deterministische Turingmaschine gibt, welche sie in polynomieller Zeit berechnet. Fast alle Sprachen, die wir bisher untersucht haben, sind **PTIME** Sprachen. Dies wird aus den folgenden Sätzen hervorgehen.

Proposition 4.1.2 *Jede kontextfreie Sprache ist in **PTIME**.*

Dies ist eine direkte Folge aus Satz 2.3.24. Jedoch bekommt man erheblich mehr.

Proposition 4.1.3 *Sei A ein endliches Alphabet, und L_1, L_2 Sprachen über A. Sind $L_1, L_2 \in$ **PTIME**, so auch $A^* - L_1$, $L_1 \cap L_2$ und $L_1 \cup L_2$.*

Der Beweis dieses Satzes ist einfach und als Übung überlassen. Wir bekommen also, daß der Schnitt von kontextfreien Sprachen, also zum Beispiel $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \in \omega\}$, eine **PTIME** Sprache ist. Ferner wollen wir zeigen, daß die vollen Urbilder von **PTIME** Sprachen unter der Parikh–Abbildung auch in **PTIME** sind. Dazu werden wir $M(A)$ mit der Menge aller Zeichenketten $\prod_{i<n} \mathtt{a}_i^{p_i}$ identifizieren. Die Parikh–Abbildung ist dann eine Funktion $\pi : A^* \to A^*$. Diese soll, gegeben eine Zeichenkette $\vec{x}$, diejenige Zeichenkette $\vec{\mathtt{a}}_0^{p_0} \cdot \vec{\mathtt{a}}_1^{p_1} \cdot \ldots \cdot \mathtt{a}_{n-1}^{p_{n-1}}$ ausgeben, für die p_j gerade die Anzahl der in $\vec{x}$ auftretenden $\mathtt{a}_j$ ist. Anschließend betrachten wir eine beliebige Funktion $g : A^* \to \{0,1\}$, welche in polynomieller Zeit deterministisch berechenbar ist und folgende Eigenschaft hat: das Urbild von 1 ist in dem Bild von π enthalten sein. $g^{-1}(1)$ kann dann in natürlicher Weise als Teilmenge von $M(A)$ aufgefaßt werden.

Theorem 4.1.4 *Es sei $S \subseteq M(A)$ in **PTIME**. Dann ist das volle Urbild $\pi^{-1}[S]$ auch in **PTIME**. Insbesondere sind alle vollen Urbilder semilinearer Mengen in **PTIME**.*

Der Leser sei an dieser Stelle gewarnt, daß nicht alle semilinearen Sprachen in **PTIME** sind. Das geht auch gar nicht. Denn **PTIME** ist abzählbar, während es überabzählbar viele semilineare Sprachen gibt.

Der Satz 4.1.4 ist die direkte Folge des folgenden, sehr einfach zu zeigenden Satzes.

Theorem 4.1.5 *Es sei $f : B^* \to A^*$ in **PTIME** und $S \subseteq A^*$ in **PTIME**. Dann ist $L := f^{-1}[S]$ auch in **PTIME**.*

Beweis. Nach Definition ist $\chi_S \in$ **PTIME**. Dann ist aber $\chi_S \circ f \in$ **PTIME**. Dies ist die charakterische Funktion von L. $\dashv$

Eine andere Forderung, welche mild kontextsensitive Sprachen erfüllen sollten, war, daß sie nur konstantes Wachstum haben sollten. Dies bedeutet anschaulich, daß die Menge $\{|\vec{x}| : \vec{x} \in S\} \subseteq \omega$ keine Lücken beliebiger Größe hat. Wir überlassen es dem Leser zu zeigen, daß falls S semilinear ist, S auch konstantes Wachstum hat, die Umkehrung jedoch nicht gilt.

Wir hatten in Anschnitt 1.2 die sogenannte Polnische Notation für Terme eingeführt. Hier nun wollen wir ein etwas exotisches Verfahren zur Notierung von Termen vorstellen, welches durch Betrachtung von gewissen australischen Sprachen motiviert ist. Sei dazu $\langle F, \Omega \rangle$ eine endliche Signatur. Ferner sei $\Omega := \max\{\Omega(f) : f \in \Omega\}$. Induktiv ordnen wir jedem Term t eine Menge $M(t) \subseteq (F \cup \{\mathtt{0}, \mathtt{1}, \ldots, \Omega - 1\})^*$ zu:

1. Ist $t = f$ mit $\Omega(f) = 0$, so setze $M(f) := \{f\}$.
2. Ist $t = f(s_0, \ldots, s_{\Omega(f)-1})$, so setze

$$M(t) := \{t\} \cup \bigcup_{i<\Omega(f)} \{\vec{x} \cdot i : \vec{x} \in M(s_i)\}$$

Ein Element von $M(t)$ ist ein Produkt $f \cdot \vec{y}$, wo $f \in F$ und $\vec{y} \in \Omega^*$. Wir nennen f das **Hauptsymbol** und $\vec{y}$ seine **Kennung**. Nun sagen wir, $\vec{y}$ sei eine **A–Form** von t, falls $\vec{y}$ das Produkt der Elemente von $M(t)$ in einer beliebigen Reihenfolge ist. Ist zum Beispiel $t = \mathtt{(((x + a) - y) + (z - c))}$, so ist

$$M(t) = \{\mathtt{+}, \mathtt{-0}, \mathtt{-1}, \mathtt{+00}, \mathtt{x000}, \mathtt{a001}, \mathtt{y01}, \mathtt{z10}, \mathtt{c11}\}$$

Entsprechend ist die folgende Zeichenkette eine A–Form von t:

```
c11z10+00-0-1y01x000a001+
```

Theorem 4.1.6 *Es sei $\langle F, \Omega\rangle$ eine endliche Signatur und S die Sprache der A–Formen der Terme dieser Signatur. Dann ist S in **PTIME**.*

Beweis. Zunächst einmal ist es sinnvoll, sich zu überlegen, daß man aus einer A–Form $\vec{x}$ die Menge M berechnen kann, aus der $\vec{x}$ gebildet wurde. Dazu muß man $\vec{x}$ nur an den richtigen Stellen auftrennen. Dies bedeutet schlicht, daß man die Elemente von M identifizieren kann. Nun beginnen wir also die Konstruktion von t. Wir gehen dabei von t in Polnischer Notation aus. Auf einem Rechenband listen wir die Zeichenreihen aus Ω^* in lexikographischer Reihenfolge auf. Diese zählen uns unsere Kennungen auf. Wir beginnen mit ε. Nun suchen wir das Element mit der Kennung ε auf und schreiben es auf die Ausgabe. Sei es f. Wir löschen die Kette $f \cdot \varepsilon$ auf der Eingabe. Ist $\Omega(f) = 0$, so muß die Eingabe jetzt leer sein. Wenen nicht, dann ist $\vec{x}$ keine A–Form. Falls ja, so gehen wir weiter zu der Kennung `0` und suchen das Symbol mit der Kennung `0`. Sei dies g`0`. Wir löschen diese Kette auf dem Eingabeband. Es gilt: ist $\Omega(g) = 0$, so darf es jetzt kein Symbol mit der Kennung `00` geben. Ansonsten ist die Eingabe keine A–Form. So machen wir weiter. Jedesmal, wenn wir eine Zeichenkette $f\vec{\alpha}$ identifiziert haben, so löschen wir $f\vec{\alpha}$ auf der Eingabe und fügen f auf der Ausgabe hinzu. Dabei muß allerdings darauf geachtet werden, daß f an der Stelle im Term tatsächlich die Kennung $\vec{\alpha}$ hat. Falls nicht, so ist die Eingabe keine A–Form. Sind wir fertig, prüfen wir ein letztes Mal, ob die Ausgabekette ein Term in Polnischer Notation ist, und dann sind wir fertig. Es fehlt noch der Nachweis, daß man in polynomieller Zeit die Kennung eines Symbols in einer Zeichenkette $\vec{x}g \in F^*$ berechnen kann. Dazu sein $\vec{x}g$ gegeben. Wir berechnen maximale Teilterme in $\vec{x}g$, die das letzte Zeichen nicht enthalten und ersetzen sie durch h, wo h ein nullstelliges Funktionssymbol ist. (Dies muß existieren, sonst ist $S = \varnothing$.) Dies tun wir so lange, wie es möglich ist. Wir erhalten die Zeichenkette $\vec{y}g$. Anschließend gehen wir diese von rechts nach links (!) durch. Für jeden maximalen Block der Form fh^p, schreiben wir p auf die Ausgabe, sofern $p < \Omega(f)$. Ansonsten ist $\vec{x}g$ ohnehin nicht das Präfix eines Terms und die gesamte Berechnung hält an. Erreichen wir den Anfang der Zeichenkette, so ist die Kennung berechnet. ⊣

Wir wollen nun die Charakterisierung von **PTIME** anpacken. Zentrales Hilfsmittel ist dazu ein Satz von Chandra, Kozen und Stockmeyer ([7]), welcher die Klasse **PTIME** auch durch eine Platzschranke charakterisiert. Dabei treten allerdings Maschinen auf, welche die Turingmaschine um eine besondere Form der Parallelität erweitern. Doch davon später. Zunächst führen wir noch eine Klasse von Funktionen ein. Wir sagen, eine Funktion $f : A^* \to B^*$ sei in **LOGSPACE**, falls sie durch

eine sogenannte deterministische logarithmisch beschränkte Turingmaschine berechnet werden kann. Eine Turingmaschine heißt **logarithmisch beschränkt**, falls sie $k+2$ Bänder hat, wobei $k=0$ sein darf. Ferner bekommt sie auf dem ersten Band die Eingabe, $\vec{x}$. Auf diesem Band darf sie nur lesen. Auf dem letzten Band, mit der Nummer $k+2$, darf sie nur schreiben, und dort muß am Ende des Prozesses das Ergebnis stehen, also $f(\vec{x})$. Alle anderen Bänder sind in ihrer Länge nach beschränkt, und zwar durch $\log_2 |\vec{x}|$. Das bedeutet, hat $\vec{x}$ die Länge 12, so haben die Bänder 2 bis $k+1$ jeweils die Länge 3, da $3 \leq \log_2 12 < 4$. Uns muß es an dieser Stelle nicht kümmern, was es mit dieser Beschränkung auf sich hat. Dafür gibt es einleuchtende Interpretationen, die wir in Abschnitt 4.2 noch liefern werden. Wir weisen darauf hin, daß $f(\vec{x})$ beliebig groß sein darf. Es ist nicht durch irgendeine Länge beschränkt. Wir werden jedoch nachweisen, daß $f(\vec{x})$ nicht allzu groß werden kann. Das letzte Band enthält die Ausgabe. Man kann sich leicht überlegen, daß man es so einrichten kann, daß die Maschine auf dem Ausgabeband nur nach rechts geht. Ferner läßt es sich stets so einrichten, daß die Rechenbänder nur jeweils eine Binärzahl enthalten.

Definition 4.1.7 *Es sei A^* ein endliches Alphabet und $S \subseteq A^*$. Wir sagen, S sei in **LOGSPACE**, falls χ_S **LOGSPACE**–berechenbar ist.*

Theorem 4.1.8 *Es sei $f : A^* \to B^*$ **LOGSPACE**–berechenbar. Dann ist f in **PTIME**.*

Beweis. Wir betrachten eine Konfiguration unserer Maschine. Dabei nehmen wir das Ausgabeband aus. Eine Konfiguration besteht aus einer Position des Lesekopfes des ersten Bandes, und einem k–Tupel von Binärzahlen der Länge $\leq \log_2 |\vec{x}|$. Eine Position auf der Zeichenkette entspricht ebenfalls einer solchen Binärzahl. Wir haben also $k+1$ Binärzahlen, und davon gibt es insgesamt

$$2^{(k+1)\cdot \log_2 |\vec{x}|} = |\vec{x}|^{k+1} \ .$$

Demnach kann die Maschine höchstens $|\vec{x}|^{k+1}$ Schritte ausführen. Werden es nämlich mehr, so ist die Maschine in einer unendlichen Schleife gefangen, und die Berechnung terminiert nicht. Da dies ausgeschlossen war, kann es also höchstens polynomiell viele Schritte geben. $\dashv$

Da also f nunmehr polynomiell berechenbar ist, ergibt sich leicht, daß $|f(\vec{x})|$ ebenfalls polynomiell beschränkt ist durch $|\vec{x}|$. Dies zeigt, daß die Platzschranke für die Rechenbänder auch eine Beschränkung für die Länge des Ergebnisses bewirken.

Wir haben also eine Teilklasse von **PTIME** gefunden, welche durch den Platzbedarf charakteristiert ist. Leider sind die beiden Klassen aber nicht gleich. Deshalb müssen wir noch ein beträchtliches Stück Arbeit verrichten. Zunächst aber beobachten wir Folgendes.

Theorem 4.1.9 *Es seien $f : A^* \to B^*$ und $g : B^* \to C^*$ in **LOGSPACE**. Dann ist auch $g \circ f$ in **LOGSPACE**.*

Beweis. Nach Voraussetzung einmal existiert eine logarithmisch beschränkte $k+2$–Band Maschine T, welche f berechnet, und eine logarithmisch beschränkte $\ell+2$–Band Maschine U, welche g berechnet. Wir kaskadieren nun diese Maschinen, indem wir das Band $k+2$ von T als Eingabeband von U benützen. Die entstehende Maschine ist deterministisch, aber nicht notwendig logarithmisch beschränkt, denn das Ausgabeband, Band Nummer $k+2$, von T ist nicht logarithmisch beschränkt. Wie wir uns überlegen wollen, benötigt man dieses Band aber gar nicht. Denn T kann auf dieses Band nicht zurückgreifen, sondern muß von links nach rechts durchgehend, Symbol für Symbol auf das Band schreiben. U hingegen würde es genügen, wenn es zu jeder Zahl i das Symbol an der $i$ten Stelle des Ergebnisses von T erfragen könnte. Da es nur um den Platzbedarf, nicht aber um den Zeitbedarf geht, so ist klar, daß auch U das Ausgabeband nicht benötigt. Wir tun also Folgendes. Wir ersetzen das Ausgabeband von T durch zwei Rechenbänder. Diese Bänder werden nichts weiter enthalten als den Binärkode der Stelle des Schreibkopfes der Maschine T auf ihrem Ausgabeband, sowie des Lesekopfes der Maschine U auf ihrem Eingabeband. Da dies auch das Ausgabeband von T ist, und $|f(\vec{x})| \leq p(|\vec{x}|)$ für ein Polynom p, so gilt $\log_2 |f(\vec{x})| \leq \lambda \log_2 |\vec{x}|$ für eine natürliche Zahl λ. Also sind diese Bänder tatsächlich auch logarithmisch beschränkt. Die Maschine arbeitet nun wie folgt. Anstelle des Zeigers auf das Leseband von U bedient sich U der Position ihres Lesekopfes, geschrieben in Binärdarstellung. Dieser wird nach Bedarf um eins erhöht oder erniedrigt. Immer wenn U das Symbol Nummer i auf dem Band benötigt, wird die Maschine T aktiviert. T bekommt nur den Index i. T berechnet nun seinerseits das $i$te Symbol der Ausgabe, ohne allerdings die Ausgabe irgendwo zu schreiben. Benötigt wird ja nur das Zeichen mit der Nummer i. Auf diese Weise umgehen wir, das Ergebnis $f(\vec{x})$ jemals in voller Länge hinschreiben zu müssen. $\dashv$

Dieser Beweis ist der Schlüssel für sämtliche noch folgenden Beweise. Wir werden jetzt zeigen, daß es eine bestimmte Klasse von Problemen gibt, welche, wie man sagt, vollständig in Bezug auf **PTIME** bezüglich **LOGSPACE**–Reduktion ist. Diese werden wir als Meßlatte für unsere Charakterisierung nehmen.

Eine n–stellige **boolesche Funktion** ist eine beliebige Funktion $f : \{0,1\}^n \to \{0,1\}$. Man kann jede Funktion durch Verkettung aus den Funktionen $\wedge$, $\vee$ und $\neg$ herstellen. Dabei ist

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

	$\neg$
0	1
1	0

Wir wollen nun annehmen, f sei irgendwie durch $\wedge$, $\vee$, $\neg$ zusammengesetzt. Etwa sei $f(x_0, x_1, x_2) := \neg(\neg x_2 \wedge (x_1 \vee x_0))$. Nun setzen wir für die Variablen x_0, x_1 und x_2

konkrete Werte ein (also entweder 0 oder 1). Welches ist dann der Wert von f? Dieses Problem läßt sich ohne großen Aufwand lösen. Es ist in **PTIME**. Dies ist recht einfach zu sehen, allerdings ist die Formulierung des Problems eine kunstreiche Sache. Wir wollen nämlich f nicht als Zeichenreihe in der üblichen Weise aufschreiben, sondern als Folge von sogenannten **Zellen**. Eine **Zelle** sei eine Zeichenkette von der Form $(\vec{\alpha},\vec{\varepsilon},\vec{\eta},\vee)$, $(\vec{\alpha},\vec{\varepsilon},\vec{\eta},\wedge)$, oder von der Form $(\vec{\alpha},\vec{\varepsilon},\neg)$, wobei $\vec{\alpha}$ eine Binärfolge ist — notiert über dem Alphabet $\{\mathtt{a},\mathtt{b}\}$ —, und $\vec{\varepsilon},\vec{\eta}$ entweder Binärfolgen (notiert über $\{\mathtt{a},\mathtt{b}\}$) sind oder Einzelzeichen von der Form $\mathtt{0}$ oder $\mathtt{1}$. $\vec{\alpha}$ heißt die **Nummer** der Zelle und $\vec{\varepsilon}$ und $\vec{\eta}$ die **Argumentkennungen**, falls sie nicht die Form $\mathtt{0}$ oder $\mathtt{1}$ haben. Ferner sei die durch $\vec{\varepsilon}$ und $\vec{\eta}$ dargestellte Zahl kleiner als die durch $\vec{\alpha}$ dargestellte Zahl. (Dies sichert, daß es keine Schleifen gibt.) Eine Folge von Zellen heißt ein **Netzwerk**, falls keine zwei Zellen die gleiche Nummer haben. Ferner verlangen wir, daß die Zellennummern die Zahlen von 1 bis zu einer gewissen Zahl ν sind und daß für jede Zelle mit Argumentkennung $\vec{\varepsilon}$ (oder $\vec{\eta}$) auch eine Zelle mit Nummer $\vec{\varepsilon}$ ($\vec{\eta}$) existiert sowie genau eine Zelle maximal ist in dem Sinne, daß ihre Nummer nicht die Argumentkennung einer anderen Zelle ist. Diese Zelle heißen wir das **Ziel** des Netzwerks. Anschaulich definiert das Netzwerk eine boolesche Funktion, in welche konstante Werte eingesetzt sind. Diese Funktion wollen wir auswerten. Wir ordnen nun jeder Zelle mit Nummer $\vec{\alpha}$ einen Wert $w(\vec{\alpha})$ wie folgt zu. Der Wert der Zelle $(\vec{\alpha},\vec{\varepsilon},\vec{\eta},\vee)$ ist $w(\vec{\varepsilon})\vee w(\vec{\eta})$, der Wert von $(\vec{\alpha},\vec{\varepsilon},\vec{\eta},\wedge)$ ist $w(\vec{\varepsilon})\wedge w(\vec{\eta})$, und der Wert von $(\vec{\alpha},\vec{\varepsilon},\neg)$ ist $\neg w(\vec{\varepsilon})$. Der Wert des Netzwerkes ist der Wert seines Ziels. Unser Problem lautet also: gegeben ein Netzwerk $\vec{x}$, berechne seinen Wert. Netzwerke sind Zeichenketten über dem Alphabet $W := \{(,),,,\mathtt{0},\mathtt{1},\mathtt{a},\mathtt{b},\wedge,\vee,\neg\}$. Wir geben ein Beispiel. Unsere Funktion $f(x_0,x_1,x_2) = \neg(\neg x_2 \wedge (x_1 \vee x_0))$ wollen wir auswerten für $x_0 := 0$, $x_1 := 1$ und $x_2 := 0$. Dann schreiben wir folgendes Netzwerk hin:

$$\mathtt{(a,0,\neg)(b,1,0,\vee)(ba,a,b,\wedge)(bb,ba,\neg)}$$

Jetzt ist $w(\mathtt{a}) = \neg 0 = 1$, $w(\mathtt{b}) = 0 \vee 1 = 1$, $w(\mathtt{ba}) = w(\mathtt{a}) \wedge w(\mathtt{b}) = 1 \wedge 1 = 1$, und es ist $w(\mathtt{bb}) = \neg w(\mathtt{ba}) = \neg 1 = 0$.

Theorem 4.1.10 *Die Menge aller Netzwerke ist in* ***LOGSPACE****. Ferner ist die Funktion, welche jedem Netzwerk seinen Wert zuordnet, in* ***PTIME****.*

Beweis. Die erste Behauptung ist etwas mühselig, aber nicht schwer zu zeigen. Dazu muß man die Zeichenkette mehrmals durchlaufen, und jeweils auf die verschiedenen Kriterien prüfen. Man prüft etwa die erste Bedingung so. Man beginnt mit der ersten Zelle und merkt sich deren Position. Dann läuft man zur nächsten Zelle, merkt sich deren Position und vergleicht dann die Zellennummern. Falls diese gleich sind, Ausgabe 0, und anhalten. Ansonsten zur nächsten Zelle. Und so weiter. Das Merken von Positionen verlangt nur logarithmisch viel Platz. Zum Vergleich dieser Zeichenketten muß man sich immer nur ein Zeichen merken. Nun also zur zweiten Behauptung. Wir haben ein Netzwerk gegeben. (Falls nicht, werfen wir den Wert

#, für undefiniert, aus.) Nun zählen wir die Zellennummern hoch und berechnen induktiv deren Wert, welche wir als Paar $(\vec{\alpha}, w(\vec{\alpha}))$ auf einem Arbeitsband ablegen. Haben wir die maximale Zellennummer erreicht, so sind wir fertig. $\dashv$

Es ist nicht bekannt, ob man den Wert eines Netzwerks in **LOGSPACE** berechnen kann. Man überlege sich, daß man möglicherweise zu viele Zwischenergebnisse speichern muß. Es gilt aber folgender Satz.

Theorem 4.1.11 *Sei $f : A^* \to \{0,1\}$ in **PTIME**. Dann existiert eine Funktion $N : A^* \to W^*$ in **LOGSPACE** derart, daß für jedes $\vec{x} \in A^*$ $N(\vec{x})$ ein Netzwerk ist und $f(\vec{x}) = (w \circ N)(\vec{x})$.*

Beweis. Wir geben zunächst eine Konstruktionsvorschrift an und deuten dann an, warum sie in **LOGSPACE** ist. Nach Voraussetzung existiert ein k und ein c derart, daß $f(\vec{x})$ in $\rho := c \cdot |\vec{x}|^k$ Zeit berechenbar ist. Ist $\vec{x}$ gegeben, so definieren wir eine Matrix $\mathcal{C} := (C(i,j))_{i,j}$, wo $0 \leq i,j \leq c \cdot |\vec{x}|^k$. Der Wert von $C(i,j)$ soll der Inhalt der $j$ten Zelle des Bandes zum Zeitpunkt i sein. Ist der Lesekopf auf dieser Zelle, so wird dies auch vermerkt, sowie der Zustand des Automaten zu diesem Zeitpunkt. Wir können dies schließlich durch eine einzige Binäzahl kodieren, welche stets eine wohldefinierte Länge, etwa λ, haben soll. Wir bezeichnen mit $C(i,j,k)$ das $k$te Binärzeichen von $C(i,j)$. Der Eintrag $C(i+1,j)$ hängt in vorhersagbarer Weise von den Einträgen $C(i,j-1)$, $C(i,j)$ und $C(i,j+1)$. (Die Maschine ist ja deterministisch.) Es gibt nun für jedes $k < \lambda$ boolesche Funktionen f_L^k, f_M^k und f_R^k derart, daß $f_L^k, f_R^k : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, $f_M^k : \{0,1\}^{3\lambda} \to \{0,1\}^\lambda$, und

$$\begin{array}{lcl} C(i+1,0,k) & = & f_L(C(i,0), C(i,1)) \\ C(i+1,j,k) & = & f_M(C(i,j-1), C(i,j), C(i,j+1)) \\ C(i+1,\rho,k) & = & f_R(C(i,\rho-1), C(i,\rho)) \end{array}$$

Das Netzwerk entsteht nun wie folgt. Wir ordnen jedem einzelnen Binärzeichen, welches nicht auf dem ersten oder letzten Zeitschritt liegt, ein Netzwerk aus Zellen zu (also haben wir höchstens $\gamma\lambda\rho(\rho-2)$ Zellen). Das Binärzeichen in der Zelle wird gerade der Wert dieser Zelle in dem Netzwerk sein. Der Zeitschritt ρ wird nicht benötigt, weil dann das Ergebnis vorliegen muß, und der Zeitschritt 0 wird nicht benötigt, weil dieser der Eingabe benötigt. Ebenso verdient der Zeitschritt ρ getrennte Behandlung. Hier wollen wir letztlich nur ein einziges Ergebnis haben, nämlich 0 oder 1. Was nun ist das Netzwerk einer Zelle? Dazu betrachten wir die Zelle zu dem $k$ten Binärzeichen von $C(i,j)$. Dies ist (zum Beispiel) von der Form $f_M^k(C(i-1,j-1), C(i-1,j), C(i-1,j+1))$. Also setzen wir das zu f_M^k gehörende Netzwerk an die Stelle. Falls $i = 1$, so finden wir $C(0,j)$ auf dem Leseband. Die Vorschrift ist nun denkbar einfach. Die Maschine bekommt $\vec{x}$ geliefert. Für i von 1 bis $\rho - 1$ und j von 0 bis ρ sowie k von 0 bis $\lambda - 1$ wird sie jeweils ein lokales Netzwerk hinschreiben, welches dem Tupel (i,j,k) enspricht und dessen Wert gerade

$C(i,j,k)$ ist. Daß dies in **LOGSPACE** ist, ist nicht schwer zu sehen. Wichtig ist, daß man nur einmal wissen muß, welches Netzwerk man hinschreiben muß, alle anderen analogen Netzwerke ergeben sich durch Verschieben der Indizes (Addieren einer konstanten Zahl zu allen Zellennummern und Argumentkennungen). Man muß also lediglich einmal alle lokalen Netzwerke ausrechnen, und kann sie dann durch einfaches Manipulieren der Binärfolgen in das Netzwerk einhängen. Gewiß ist dies **LOGSPACE** berechenbar. ⊣

Man sagt auch, daß die Berechnung von Werten eines Netzwerks vollständig bezüglich der Klasse **PTIME** ist. Ein Netzwerk heißt **monoton**, falls es nicht das Symbol $\neg$ enthält.

Theorem 4.1.12 *Es existiert eine **LOGSPACE**–berechenbare Funktion M, welche ein beliebiges Netzwerk in ein monotones Netzwerk gleichen Werts umformt.*

Der Beweis ist einigermaßen langwierig, wir wollen ihn daher nur skizzieren. Wir nehmen an, unser Netzwerk berechne die Funktion $f(\vec{x})$. Dann ändern wir $\vec{x}$ so ab, daß wir nur mit $\wedge$ und $\vee$ hinkommen. Ist $\vec{x} = \prod_{i<n} x_i$, so definiere $y_i := \mathtt{01}$, falls $x_i = \mathtt{0}$, und $y_i := \mathtt{10}$ sonst. Schließlich setze $\vec{y} := \prod_{i<n} y_i$. Dies bedeutet: haben wir an der Stelle $2i$ eine $\mathtt{1}$, so haben wir an der Stelle $2i+1$ eine $\mathtt{0}$, und haben wir an der Stelle $2i$ eine $\mathtt{0}$, so haben wir an der Stelle $2i+1$ eine $\mathtt{1}$. Ist zum Beispiel $\vec{x} = \mathtt{010}$, so ist $\vec{y} = \mathtt{011001}$. Wir behaupten: es existiert eine monotone Funktion g derart, daß $g(\vec{y}) = f(\vec{x})$. Dies ist leicht zu zeigen, und wird als Übung überlassen. Der Input wird also durch seine Negation verdoppelt. Dies übertragen wir auf das Netzwerk. Jede Zelle der Form $(\vec{\alpha}, \vec{\varepsilon}, \vec{\eta}, \wedge)$ wird nun ebenfalls verdoppelt, nämlich durch eine Zelle der Form $(\vec{\alpha}_1, \vec{\varepsilon}_1, \vec{\eta}_1, \vee)$ und dual dazu die Zelle der Form $(\vec{\alpha}, \vec{\varepsilon}, \vec{\eta}, \vee)$ durch eine Zelle der Form $(\vec{\alpha}_1, \vec{\varepsilon}_1, \vec{\eta}_1, \wedge)$. Grundlage sind die sogenannten de Morgan'schen Gleichungen $\neg(x \wedge y) = (\neg x) \vee (\neg y)$ und $\neg(x \vee y) = (\neg x) \vee (\neg y)$. Mit jeder Zelle Z, welche den Wert x hat, existiert eine Zelle Z_1 mit Wert $\neg x$. Praktischerweise wollen wir es so halten, daß die Nummern von Z und Z_1 sich nur um eins unterscheiden. Nun können wir allerdings $\neg$ völlig eliminieren. Denn falls Zelle Z den Wert von der Zelle U (im alten Netzwerk) mit der Nummer k negiert, so können wir den Wert von U an der Stelle k' finden, und den von Z daher an der Stelle $k'+1$ bzw. $k'-1$, je nachdem ob k' gerade oder ungerade ist. Am Schluß muß man noch etwas Kosmetik betreiben, indem man nur die Zellen behält, die man wirklich zur Berechnung des Ziels benötigt.

Wir haben nun die folgende Situation: ist g eine beliebige **PTIME**–berechenbare Funktion von A^* nach $\{0,1\}$, so existiert eine **LOGSPACE**–berechenbare Funktion N^+, welche zu gegebenem $\vec{x}$ ein mononotones Netzwerk gleichen Werts konstruiert.

Nun wenden wir uns den versprochenen neuartigen Turingmaschinen zu.

Definition 4.1.13 *Eine **alternierende Turingmaschine** ein Septupel*

$$\langle A, \mathtt{L}, Q, q_0, F, f, \gamma \rangle \ ,$$

*wo $\langle A, \mathtt{L}, Q, q_0, f \rangle$ eine Turingmaschine ist und $\gamma : Q \to \{\wedge, \vee\}$ eine beliebige Funktion. Ein Zustand q heißt **universell**, falls $\gamma(q) = \wedge$, und andernfalls **existentiell**.*

Wir verallgemeinern stillschweigend auch die anderen Konzepte der Turingmaschine; wir setzen also voraus, daß der Leser in der Lage ist, eine alternierende Mehrbandmaschine zu defineren, sowie eine alternierende logarithmisch beschränkte Mehrbandmaschine. Dazu muß man stets einfach noch die Funktion γ hinzufügen. Wann akzeptiert nun eine alternierende Turingmaschine eine Zeichenkette $\vec{x}$? Dies definiert man am besten über Konfigurationen. Eine Konfiguration heißt **akzeptiert** durch T, falls einer der folgenden Fälle eintritt:

* T ist in einem existientiellen Zustand und eine der nachfolgende Konfigurationen wird von T akzeptiert.
* T ist in einem universellen Zustand und alle nachfolgenden Konfigurationen werden von T akzeptiert.

Man beachte, die Maschine kann eine Konfiguration akzeptieren, wenn es keine nachfolgenden Konfigurationen gibt (und die Berechnung also terminiert), vorausgesetzt, sie ist in einem universellen Zustand. Der Unterschied zwischen universellen und existentiellen Zuständen greift aber eigentlich erst, wenn die Maschine nicht deterministisch ist. Denn dann kann es mehrere nachfolgende Konfigurationen geben. Die typische Definition einer Turingmaschine definiert hier Akzeptanz als wie für einen existentiellen Zustand. Universelle Zustände sind aber anderer Natur. In diesem Fall muß die Maschine sich in mehrere parallel arbeitende Maschinen teilen, welche jeweils eine nachfolgende Konfiguration bearbeiten. Wir definieren nun $f : A^* \to B^*$ als **ALOGSPACE**, falls es eine logarithmisch beschränkte alterminerende Mehrbandmaschine gibt, welche f berechnet.

Definition 4.1.14 *Eine Sprache $S \subseteq A^*$ ist in **ALOGSPACE**, wenn ihre charakteristische Funktion **ALOGSPACE**–berechenbar ist.*

Theorem 4.1.15 (Chandra & Kozen & Stockmeyer) ***PTIME*** $=$ ***ALOGSPACE***.

Zum Beweis benötigen wir nicht mehr viel. Zunächst einmal ist Folgendes klar.

Lemma 4.1.16 ***LOGSPACE*** $\subseteq$ ***ALOGSPACE***.

Denn jede deterministische logarithmisch beschränkte Turingmaschine ist auch eine alternierende Maschine, indem man einfach jeden Zustand als universell definiert. Ebenso ist folgende Behauptung leicht gezeigt, wenn man an die Erläuterungen zu **LOGSPACE**–berechenbaren Funktionen anknüpft.

Lemma 4.1.17 *Sind $f : A^* \to B^*$ und $g : B^* \to C^*$ in **ALOGSPACE**, so auch $g \circ f$.*

Lemma 4.1.18 ***ALOGSPACE*** $\subseteq$ ***PTIME***.

Auch hier ist der Beweis nicht schwer. Wir wissen schon, daß es höchstens polynomiell viele Konfigurationen gibt. Die Abhängigkeiten zwischen diesen Konfigurationen lassen sich in polynomieller Zeit verfolgen. (Jede Konfiguration hat eine beschränkte Anzahl Nachfolger. Die Schranke hängt nur von T ab.) Diese gibt einen Berechnungsbaum, den man also in polynomieller Zeit bestimmen kann. Nun müssen wir in einem letzten Schritt bestimmen, ob die Maschine die Anfangskonfiguration akzeptiert. Dazu müssen wir durch Induktion über die Tiefe in unserem Berechnungsbaum bestimmen, ob die jeweiligen Konfigurationen akzeptiert werden. Dies ist ebenfalls in polynomieller Zeit möglich. Dies beendet den Beweis.

Es bleibt uns als Letztes die umgekehrte Inklusion. Dazu folgende Idee. Es sei f in **PTIME**. Wir können f schreiben als $w \circ N^+$, wo N^+ eine monotones Netzwerk berechnet. Wie oben angemerkt, kann man N^+ in **LOGSPACE** konstruieren, also insbesondere wegen Lemma 4.1.16 in **ALOGSPACE**. Falls wir nun zeigen können, daß w in **ALOGSPACE** ist, so greift Lemma 4.1.17 und liefert uns, daß $f = w \circ N^+ \in$ **ALOGSPACE**.

Lemma 4.1.19 $w \in$ ***ALOGSPACE***.

Beweis. Wir konstruieren eine logarithmische beschränkte alternierende Maschine, welche für ein beliebiges mononotones Netzwerk $\vec{x}$ seinen Wert $w(\vec{x})$ bestimmt. Sei ein monotones Netzwerk gegeben. Wir gehen zunächst zu seinem Ziel. Absteigend vom Ziel führen wir folgende Schritte aus.

1. Enthält die Zelle $\wedge$, so gehe in den universellen Zustand q_1.
2. Enthält die Zelle $\vee$, so gehe in den existentiellen Zustand q_2.
3. Wähle eine Argumentkennung $\vec{\alpha}$ der aktuellen Zelle und gehe zur Zelle Nummer $\vec{\alpha}$.

4. Falls $\vec{\alpha}$ keine Argumentkennung ist, so gehe in den Zustand q_f über, falls $\vec{\alpha} = \mathtt{1}$, und in q_g, falls $\vec{\alpha} = \mathtt{0}$ ist. Hierbei ist q_f universell und q_g existentiell, und es sind keine Übergänge von q_f und q_g definiert.

Alle anderen Zustände sind übrigens universell, aber die Maschine arbeitet ohnehin deterministisch. Nur in einem Fall tut sie es nicht, nämlich wenn sie sich die Werte ihrer Argumente holt. Dann wählt sie nichtdeterministisch ein Argument aus. Ist die Zelle eine $\vee$–Zelle, so wird sie an dieser Stelle die Konfiguration akzeptieren, wenn ein Argument den Wert 1 ergibt, da der Zustand existientiell ist. Ist die Zelle aber eine $\wedge$–Zelle, so wird sie die Konfiguration akzeptieren, wenn beide Argumente den Wert 1 haben, denn jetzt ist der Zustand universell. Die letzte Bedingung ist die Terminierungsbedingung. Falls die Zeichenkette keine Argumentkennung ist, so ist sie `0` oder `1`, und ihr Wert kann ohne Rückgriff auf weitere Zellen ausgerechnet werden. Ist dieser `1`, so wechselt der Automat in einen finalen Zustand, der universell ist, und somit ist die Konfiguration akzeptiert. Ist der Wert `0`, so wechselt der Automat in einen finalen Zustand, der existentiell ist, und die Konfiguration wird abgelehnt. $\dashv$

Übung 91. Zeigen Sie Proposition 4.1.3: Mit S_1 und S_2 sind auch $S_1 \cup S_2$, $S_1 \cap S_2$ sowie $A^* - S_1$ in **PTIME**.

Übung 92. Zeigen Sie, daß S konstantes Wachstum besitzt, falls S semilinear ist. Geben Sie ein Beispiel einer Sprache, welche konstantes Wachstum hat, aber nicht semilinear ist.

Übung 93. Zeigen Sie, daß zu jeder n–stelligen booleschen Funktion f eine monotone boolesche $2n$–stelligen Funktion g existiert, derart, daß $f(x_0, \ldots, x_{n-1}) = g(x_0, \neg x_0, x_1, \neg x_1, \ldots, x_{n-1}, \neg x_{n-1})$.

4.2 Literalbewegungsgrammatiken

Das Konzept der Literalbewegungsgrammtik — kurz **LBG** — wurde vor wenigen Jahren von Annius Groenink entwickelt (siehe [21]). Mit Hilfe dieser Grammatiken kann man die PTIME Sprachen sehr einfach charakterisieren. Die Idee zu dieser Charakterisierung geht auf ein Resultat von Bill Rounds zurück ([43]). Außerdem lassen sich viele bekannte Grammatiktypen, die wir im Einzelnen noch vorstellen werden, bequem im Rahmen der LBGn darstellen. Die wesentliche Idee bei einer LBG ist die, daß die Regeln ein kontextfreies Skelett enthalten, welches gewissermaßen die abstrakte Struktur der Zeichenkette beschreibt, und eine explizite Beschreibung der Art und Weise, wie die Zeichenkette aus ihren Teilen zusammengesetzt wird. Die Notation wird dazu wie folgt verändert. Anstelle eines Nichtterminalsymbols, sagen wir `Q`, schreiben wir jetzt $\mathtt{Q}(\vec{x})$, wo $\vec{x}$ eine Zeichenkette ist, und sagen, die Zeichen-

kette $\vec{x}$ habe die Eigenschaft $\mathtt{Q}$ oder $\vec{x}$ sei eine $\mathtt{Q}$–Zeichenkette. Die Eigenschaften spielen nun die Rolle der Nichtterminalsymbole in den kontextfreien Grammatiken, aber technisch gesehen werden sie anders gehandhabt. Denn eine LBG behandelt sie nicht als *Symbol*, und darum schreiben wir auch nicht $\mathtt{Q}$, sondern $\mathtt{Q}(\vec{x})$. In einer LBG werden Einheiten von der Form $\mathtt{Q}(\vec{x})$ manipuliert. Da $\vec{x}$ bisher eine konkrete Zeichenkette war, müssen wir zur Formulierung von LBGn nun noch Variablen für Zeichenketten einführen. Dies seien einfach die üblichen Symbole x, y, z usf. Abgesehen von diesen Variablen gibt es auch Konstanten $\mathtt{a}$, $\mathtt{b}$, für die Symbole aus unserem Alphabet A. Wir geben nun ein einfaches Beispiel einer LBG an. Sie besteht aus zwei Regeln:

$$\mathtt{S}(x \cdot x) \leftarrow \mathtt{S}(x), \qquad \mathtt{S}(\mathtt{a}) \leftarrow .$$

Diese Regeln sind wie bei `Prolog` im Hornklauselformat niedergeschrieben. Wir können sie in der Tat auch als Hornklauseln auffassen. Die erste Regel sagt: ist x eine Zeichenkette vom Typ $\mathtt{S}$, so ist auch $x \cdot x$ eine Zeichenkette vom Typ $\mathtt{S}$. Die zweite Regel sagt: $\mathtt{a}$ ist eine Zeichenkette vom Typ $\mathtt{S}$. Wir vereinbaren daß $\mathtt{S}$ — wie üblich — das ausgezeichnete Symbol sei. Die von dieser Menge erzeugte Sprache sei die kleinste Menge, die den beiden Regeln genügt. Wir bezeichnen sie mit $L(G)$. Wir können die Definitionen wie folgt in Prädikatenlogik niederschreiben:

$$\varphi(S) := (\forall x)(S(x) \to S(x \cdot x)) \wedge S(\mathtt{a})$$

$$L(G) = \{x : (\forall S)(\varphi(S) \to S(x))\}$$

$L(G)$ läßt sich also in sogenannter monadischer Logik zweiter Stufe beschreiben, falls man Variablen für Zeichenketten nimmt und als Symbole die Gleichheit, Konstanten für ε, jeden Buchstaben und die Verkettungsoperation $\cdot$. Vergleiche dazu die Ausführungen im Kapitel 5!

Proposition 4.2.1 $L(G) = \{\mathtt{a}^{2^n} : 0 \leq n\}$.

Beweis. Zunächst gilt: $\mathtt{a} \in L(G)$. Dies zeigt den Fall $n = 0$. Induktiv zeigt man nun $\mathtt{a}^{2^n} \in L(G)$ für jedes $n > 0$. Ist nämlich $\mathtt{a}^{2^n}$ eine Zeichenkette vom Typ $\mathtt{S}$, so auch $\mathtt{a}^{2^{n+1}} = \mathtt{a}^{2^n} \cdot \mathtt{a}^{2^n}$. Dies zeigt, daß $L(G) \supseteq \{\mathtt{a}^{2^n} : n \geq 0\}$. Andererseits erfüllt $L(G)$ die Formel φ. Denn es ist $\mathtt{a} \in L(G)$ und mit $\vec{x} \in L(G)$ ist auch $\vec{x} \cdot \vec{x} \in L(G)$. Denn ist $\vec{x} = \mathtt{a}^{2^n}$ für ein gewisses $n \geq 0$, so ist $\vec{x} \cdot \vec{x} = \mathtt{a}^{2^n \cdot 2} = \mathtt{a}^{2^{n+1}} \in L(G)$. $\dashv$

Es gibt alternativ zu der Definition von $L(G)$ eine direkte Definition über die Erzeugung. Wir sagen, $G \vdash \mathtt{S}(\vec{x})$ (Vektorpfeil!), falls entweder $S(\vec{x}) \leftarrow .$ eine Regel ist, oder $\vec{x} = \vec{y} \cdot \vec{y}$ und $G \vdash \mathtt{S}(\vec{y})$. Beide Definitionen laufen aber letztlich auf dasselbe hinaus. Wir definieren nun zunächst sogenannte 1–LBGn.

Definition 4.2.2 *Es sei A ein Alphabet. Eine **1–LBG** über A ist ein Quadrupel $\langle \mathtt{S}, N, A, R\rangle$, wo N eine endliche Menge disjunkt zu A, die Menge der **Prädikate**,*

$\mathtt{S} \in N$, *und* R *eine endliche Menge von* ***Regeln****. Eine (****n–stellige****) Regel hat die Form*

$$T(t) \leftarrow U_0(s_0)\ U_1(s_1)\ \ldots U_{n-1}(s_{n-1})$$

wobei t, s_i $(i < n)$ *Polynome sind, welche aus den Terminalsymbolen, Variablen und* ε *mit Hilfe von* $\cdot$ *(Konkatenation) gebildet sind.*

Man beachte den Fall $n = 0$. Im obigen Fall haben wir $n = 1$ und $T = U_0 = \mathtt{S}$, $t = x \cdot x$ und $s_0 = x$, bzw. $n = 0$ und $T = \mathtt{S}$, $t = \mathtt{a}$. Diese Definition ist nicht die endgültige, obwohl auch sie schon abschreckend genug aussehen mag. Wir definieren nun, was es heißt, eine 1–LBG erzeuge oder akzeptiere eine Zeichenkette. Dazu nur noch folgende Vereinbahrung. Es heiße eine Funktion α, welche jeder Variablen eine Zeichenkette aus A^* zuordnet, eine **Belegung**. Gegeben α, so definieren wir s^α für ein Polynom durch homomorphe Fortsetzung. Ist etwa $s = \mathtt{a} \cdot x^2 \cdot \mathtt{b} \cdot y$ und ist $\alpha(x) = \mathtt{ac}$ und $\alpha(y) = \mathtt{bba}$, so ist $s^\alpha : \mathtt{aacacbbba}$, wie man leicht nachrechnet.

Definition 4.2.3 *Es sei* $G = \langle \mathtt{S}, N, A, R\rangle$ *eine 1–LBG über* A*. Wir definieren* $G \vdash^0 Q(\vec{x})$*, falls es ein Polynom* s *und eine Belegung* α *gibt mit (a)* $\vec{x} = s^\alpha$ *und (b)* $Q(s) \leftarrow . \in R$*. Ferner gelte* $G \vdash^{n+1} Q(\vec{x})$*, falls* $G \vdash^n Q(\vec{x})$ *oder falls es ein* $n \in \omega$ *gibt, Polynome* t, s_i, $i < n$, *Prädikate* R_i, $i < n$, *Zeichenketten* $\vec{y}_i$, $i < n$, *sowie eine Belegung* α *derart, daß gilt:*

- $\vec{x} = t^\alpha$,
- $\vec{y}_i = s_i^\alpha$ *für alle* $i < n$,
- $G \vdash^n R_i(\vec{y}_i)$ *für alle* $i < n$, *und*
- $Q(t) \leftarrow R_0(s_0)\ \ldots\ R_{n-1}(s_{n-1}) \in R$.

Schließlich sei $G \vdash Q(\vec{x})$ *genau dann, wenn ein* $n \in \omega$ *existiert mit* $G \vdash^n Q(\vec{x})$*. Und es sei*

$$L(G) := \{\vec{x} : G \vdash \mathtt{S}(\vec{x})\}$$

Wir wollen dies an einigen Beispielen erläutern. Es sei K die folgende Grammatik.

$$\mathtt{S}(v \cdot x \cdot y \cdot z) \leftarrow \mathtt{S}(v \cdot y \cdot x \cdot z), \qquad \mathtt{S}(\mathtt{a} \cdot \mathtt{b} \cdot \mathtt{c} \cdot x) \leftarrow \mathtt{S}(x), \quad \mathtt{S}(\varepsilon) \leftarrow .$$

Dann ist $L(K)$ diejenige Sprache, welche alle Zeichenketten erzeugt, in denen die gleiche Anzahl $\mathtt{a}$, $\mathtt{b}$ und $\mathtt{c}$ vorkommen. Dazu zeigt man zunächst, daß $(\mathtt{abc})^* \in L(K)$ ist, und daß (aufgrund der ersten Regel) $L(K)$ abgeschlossen ist unter Permutation der Zeichenketten. Hierbei ist $\vec{y}$ eine Permutation von $\vec{x}$, wenn $\vec{y}$ und $\vec{x}$ das gleiche

Parikh–Bild haben. Dazu ein Beispiel (der allgemein Fall ist als Übung überlassen). Es sei $\mathtt{S}(\mathtt{aabbcc})$. Setze $\alpha(v) := \mathtt{a}$, $\alpha(x) := \mathtt{ab}$, $\alpha(y) := \mathtt{bc}$ und $\alpha(z) := \mathtt{c}$. Dann ist

$$(v \cdot x \cdot y \cdot z)^\alpha = \mathtt{aabbcc}, \quad (v \cdot y \cdot x \cdot z)^\alpha = \mathtt{abcabc}$$

Es gibt noch eine andere Art, diese Definitionen aufzufassen. Wir sagen, es sei $T(\vec{x}) \leftarrow U_0(\vec{y}_0)\ U_1(\vec{y}_1)\ \ldots\ U_{n-1}(\vec{y}_{n-1})$ sei eine **Instanz** der Regel

$$T(t) \leftarrow U_0(s_0)\ U_1(s_1)\ \ldots\ U_{n-1}(s_{n-1})$$

falls es eine Belegegung β gibt derart, daß $\vec{x} = t^\beta$ und $\vec{y}_i = s_i^\beta$ für alle $i < n$. Nun können wir sagen: $G \vdash^0 Q(\vec{x})$, falls $Q(\vec{x}) \leftarrow$. Instanz einer Regel von G ist, und $G \vdash^{n+1} Q(\vec{x})$, falls $G \vdash^n Q(\vec{x})$ oder es eine Zahl n gibt, Prädikate R_i und Zeichenketten $\vec{y}_i$, $i < n$, derart, daß

* $G \vdash^n R_i(\vec{y}_i)$, und
* $Q(\vec{x}) \leftarrow R_0(\vec{y}_0)\ \ldots\ R_{n-1}(\vec{y}_{n-1})$ Instanz einer Regel von G ist.

Dies macht das Verständnis des Erzeugungsbegriffs etwas leichter. Es geht also darum, daß die Regeln als Schemata aufgefaßt werden in denen man konkrete Zeichenketten durch Variablen ersetzt.

Es sei H eine kontextfreie Grammatik. Dann formulieren wir eine LBG $H^\spadesuit$ wie folgt. (Wir nehmen für die Präsentation an, daß H in Chomsky–Normalform ist.) Für jedes Nichtterminalsymbol $\mathtt{A}$ führen wir eine Variable $\underline{\mathtt{A}}$ ein. Das Startsymbol ist also $\underline{\mathtt{S}}$. Ist $\mathtt{A} \to \mathtt{B}\ \mathtt{C}$ eine Regel aus H, so enthält $H^\spadesuit$ die Regel

$$\underline{\mathtt{A}}(x \cdot y) \leftarrow \underline{\mathtt{B}}(x)\ \underline{\mathtt{C}}(y)$$

Ist $\mathtt{A} \to \mathtt{a}$ eine terminale Regel, so führen wie die folgende Regel in $H^\spadesuit$ ein:

$$\underline{\mathtt{A}}(\mathtt{a}) \leftarrow .$$

Man kann relativ leicht zeigen, daß $L(H) = L(H^\spadesuit)$.

LBGs erlauben also, alle kontextfreien Sprachen zu erzeugen wie auch Sprachen mit nichtkonstantem Wachstum. Halten wir aber zunächst fest:

Theorem 4.2.4 *Es seien S_1 und S_2 Sprachen über A, die durch 1–LBGn erzeugbar sind. Dann existieren 1–LBGn, welche die Sprachen $S_1 \cap S_2$ und $S_1 \cup S_2$ erzeugen.*

Beweis. Es seien G_1 und G_2 1–LBGn, welche S_1 bzw. S_2 erzeugen. Wir nehmen an, die Mengen der Nichtterminalsymbole von G_1 und von G_2 sind disjunkt. Es sei $\mathtt{S}_i$ das Startprädikat von G_i, $i \in \{1,2\}$. Es sei $H_\cup$ wie folgt konstruiert. Wir bilden die Vereinigung der Nichtterminalsymbole und Regeln aus G_1 und G_2. Ferner sei $\underline{\mathtt{S}}^\heartsuit$ ein neues Prädikat, welches das Startprädikat von H wird. Am Schluß fügen wir noch folgende Regeln hinzu: $\mathtt{S}^\heartsuit(x) \leftarrow \mathtt{S}_1(x)$, $\mathtt{S}^\heartsuit(x) \leftarrow \mathtt{S}_2(x)$. Dies definiert $H_\cup$. $H_\cap$ ist ähnlich definiert, nur daß wir anstelle der letzten beiden Regeln eine einzige Regel hinzufügen, und zwar $\mathtt{S}^\heartsuit(x) \leftarrow \mathtt{S}_1(x)\ \mathtt{S}_2(x)$. Man überlegt sich nun leicht, daß $L(H_\cup) = S_1 \cup S_2$ ist und $H_\cap = S_1 \cap S_2$. Wir führen dies für $H_\cap$ vor. Es gilt $\vec{x} \in L(H_\cap)$, falls ein n existiert mit $H_\cap \vdash^n \mathtt{S}^\heartsuit(\vec{x})$. Dies wiederum ist genau dann der Fall, wenn $H_\cap \vdash^{n-1} \mathtt{S}_1(\vec{x})$ und $H_\cap \vdash^{n-1} \mathtt{S}_2(\vec{x})$. Dies ist nichts anderes als $G_1 \vdash^{n-1} \mathtt{S}_1(\vec{x})$ und $G_2 \vdash^{n-1} \mathtt{S}_1(\vec{x})$. Da n beliebig war, ist also $\vec{x} \in H_\cap$ genau dann, wenn $\vec{x} \in L(G_1) = S_1$ und $\vec{x} \in L(G_2) = S_2$, wie versprochen. $\dashv$

Die 1–LBGn sind sehr stark, wie folgender Satz zeigt.

Theorem 4.2.5 *Es sei A ein endliches Alphabet und $S \subseteq A^*$. Genau dann ist S durch eine 1–LBG erzeugbar, wenn S aufzählbar ist.*

Der Beweis ist dem Leser als Übung überlassen. Da die Menge der aufzählbaren Sprachen über A abgeschlossen ist unter Vereinigung und Schnitt, folgt Satz 4.2.4 bereits aus Satz 4.2.5. Es folgt auch, daß das Komplement einer durch eine 1–LBG erzeugbaren Sprache nicht notwendig durch eine 1–LBG erzeugbar sein muß. Denn das Komplement einer aufzählbaren Sprache ist nicht notwendig aufzählbar.

Um zu interessanten Klassen von Sprachen zu kommen, muß man also das Format von Regeln irgendwie einschränken. Dazu einige Definitionen. Es sei folgende Regel gegeben.

$$\rho := T(t) \leftarrow U_0(s_0)\ U_1(s_1)\ \ldots U_{n-1}(s_{n-1})$$

- ρ heißt **aufwärts nichtlöschend**, falls jede Variable, welche in einem s_i, $i < n$, vorkommt, auch in t vorkommt.
- ρ heißt **aufwärts linear**, falls keine Variable mehr als einmal in t auftritt.
- ρ heißt **abwärts nichtlöschend**, falls jede Variable, die in t vorkommt, auch in einem der s_i vorkommt.
- ρ heißt **abwärts linear**, falls keine der Variablen mehrfach in den s_i auftritt. (Dies heißt: die s_i sind paarweise disjunkt in ihren Variablen, und keine Variable tritt doppelt in einem der s_i auf.)
- ρ heißt **nichtkombinatorisch**, falls die s_i Variablen sind.

* ρ heißt **einfach**, falls es nichtkombinatorisch, aufwärts nichtlöschend und aufwärts linear ist.

G hat dann die Eigenschaft $\mathcal{P}$, falls alle Regeln die Eigenschaft $\mathcal{P}$ haben. Besonders der Typ der einfachen Grammatiken wird uns noch beschäftigen. Die Definitionen entsprechen nicht immer dem, was man intuitiv erwarten würde. Zum Beispiel ist die folgende Regel aufwärts nichtlöschend, obwohl das Ergebnis der Regelanwendung ein Symbol tilgt: $\mathtt{U}(x) \leftarrow \mathtt{U}(x \cdot \mathtt{a})$. Dies ist so, weil in der Definition nur verlangt wird, daß keine Variable verlorengeht. Ferner könnte man alternativ aufwärts linear so definieren, daß man verlangt, daß jedes Symbol mindestens ebenso oft in t auftritt wie in allen s_i zusammen. Dies ist jedoch zu stark. Man möchte sich erlauben, daß eine Variable rechts doppelt auftreten kann, auch wenn sie links nur einmal auftritt.

Lemma 4.2.6 *Es sei ρ einfach und $Q(\vec{y}) \leftarrow R_0(\vec{x}_0)\ R_1(\vec{x}_1)\ \ldots\ R_{n-1}(\vec{x}_{n-1})$ eine Instanz von ρ. Dann gilt $|\vec{y}| \geq \max_{i<n} |\vec{x}_i|$. Ferner ist $\vec{x}_i$ Teilwort von $\vec{y}$ für jedes $i < n$.*

Theorem 4.2.7 *Es sei $S \subseteq A^*$ erzeugbar durch eine einfache 1–LBG. Dann ist S in **PTIME**.*

Beweis. Sei $\vec{x}$ eine beliebige Zeichenkette und $n := \sharp N \cdot |\vec{x}|$. Aufgrund von Lemma 4.2.6 gilt für jedes Prädikat Q: $G \vdash Q(\vec{x})$ genau dann, wenn $G \vdash^n Q(\vec{x})$. Daraus folgt, daß eine Ableitung von $\mathtt{S}(\vec{x})$ höchstens die Länge n hat. Ferner entstehen bei der Ableitung nur Prädikate der From $Q(\vec{y})$, wo $\vec{y}$ ein Teilwort von $\vec{x}$ ist. Folgender Algorithmus (welcher eine Modifikation des Chart–Algorithmus ist) benötigt polynomiell viel Zeit:

* Für $i = 0, \ldots, n$: Für jedes $\vec{y}$ der Länge i und jedes Prädikat Q prüfe, ob es Teilworte $\vec{z}_j$, $j < p$, der Länge $< i$ gibt, und Prädikate R_j, $j < p$, sodaß $Q(\vec{y}) \leftarrow R_0(\vec{z}_0)\ R_1(\vec{z}_1)\ \ldots\ R_{p-1}(\vec{z}_{p-1})$ eine Instanz einer Regel von G ist.

Der Grad des Polynoms ist $2p + 3$. $\dashv$

Die Umkehrung gilt aller Wahrscheinlichkeit nach nicht. Nun wollen wir das Konzept der Literalbewegungsgrammtik in voller Allgemeinheit definieren. Im Grunde genommen fehlt uns zur allgemeinen Definition nur noch die Vektorisierung. Wir lassen zu, daß unsere Prädikate nicht nur einstellige Prädikate sind, sondern beliebige k–stellige Prädikate. Das k kann man dabei uniform für alle Prädikate der Grammatik wählen. Es ist jedoch zweckmäßig, die Stelligkeit der Prädikate variabel zu halten.

Definition 4.2.8 *Es sei A ein Alphabet. Eine **LBG** über A ist ein Tripel $\langle \mathtt{S}, N, \Omega, A, R\rangle$, wo N eine endliche Menge disjunkt zu A ist, die Menge der **Prädikate**, $\Omega : N \to \omega$ eine Funktion, die jedem Prädikat seine Stelligkeit zuweist, $\mathtt{S} \in N$ mit $\Omega(\mathtt{S}) = 1$, und R eine endliche Menge von **Regeln**. Eine (**n–stellige**) Regel hat die Form*

$$T(t^0, t^1, \ldots, t^{p-1}) \leftarrow \quad U_0(s_0^0, s_0^1, \ldots, s_0^{q_0-1})\ U_1(s_1^0, s_1^1, \ldots, s_1^{q_1-1})\ \ldots \\ \ldots U_{n-1}(s_{n-1}^0, s_{n-1}^1, \ldots, s_{n-1}^{q_n-1})$$

*wobei $q := \Omega(T)$, $q_i := \Omega(U_i)$, sowie die t^j und die s_i^j $(i < n, j < q_i)$ Polynome sind, welche aus den Terminalsymbolen, Variablen und ε mit Hilfe von $\cdot$ (Konkatenation) gebildet sind. Ist k mindestens so groß wie das Maximum aller $\Omega(U)$, $U \in N$, so sagen wir auch, G sei eine k–**LBG**.*

Für die Zukunft wollen wir noch folgende Terminologie vereinbahren.

Definition 4.2.9 *Es sei P ein n–stelliges Prädikate und t_i, $i < n$, Terme. Dann heißt $L := P(\langle t_i : i < n\rangle)$ ein **Literal**. Ist σ eine Substitution von Termen für Variable , so heißt $L^\sigma := P(\langle t_i^\sigma : i < n\rangle)$ eine **Instanz** von L. Ist σ eine Einsetzung, das heißt, ist $\sigma(x) \in A^*$ für alle x, so heißt σ **Grundsubstitution** und L^σ eine **Grundinstanz** von L.*

Es gelten nun die gleichen Begriffe von Instanz und Erzeugung wie vorher. Ebenso werden die Begriffe auf den allgemeinen Fall übertragen. Es tritt des öfteren der Fall ein, daß man für alle Prädikate die gleiche Stelligkeit haben möchte. Dies kann man ohne Schwierigkeiten tun, indem man ein i–stelliges Prädikat A (wo $i < k$) durch ein k–stelliges Prädikat A^* ersetzt mit

$$A^*(x_0, \ldots, x_{k-1}) \leftrightarrow A(x_0, \ldots, x_{i-1}) \wedge \bigwedge_{j=i}^{k-1} x_j \doteq \varepsilon$$

Eine kleine Schwierigkeit besteht allerdings darin, daß das Startprädikat 1–stellig sein muß. Wir erlauben also, daß auch das Startprädikat k–stellig ist und setzen in diesem Fall

$$L(G) := \{\prod_{i<k} \vec{x}_i : G \vdash \mathtt{S}(x_0, \ldots, x_{k-1})\}$$

Eine wichtige Klasse, welche wir im Folgenden studieren wollen, ist die Klasse der einfachen LBGn. Man beachte, daß aufgrund der Definitionen eine Variable auf der rechten mehrmals, auf der linken Seite aber nur einmal auftreten darf. Diese Einschränkung stellt sich als nicht wirksam heraus. Betrachte folgende Grammatik.

$$\begin{array}{lcll} E(\varepsilon, \varepsilon) & \leftarrow & . & \\ E(a, a) & \leftarrow & . & (a \in A), \\ E(xa, ya) & \leftarrow & E(x, y). & (a \in A). \end{array}$$

Diese Grammatik erzeugt alle Paare $\langle \vec{x}, \vec{x} \rangle$ mit $\vec{x} \in A^*$. Sie ist einfach. Nehmen wir nun eine Regel, in der auf der linken Seite eine Variable mehrfach auftritt.

$$S(x \cdot x) \leftarrow S(x).$$

Diese ersetzen wir jetzt durch die folgenden Regel:

$$S(x \cdot y) \leftarrow S(x), \quad E(x, y).$$

Diese ist jetzt einfach. Zusammen mit den Regeln für E bekommmen wir eine Grammatik, welche einfach ist und dieselben Zeichenketten erzeugt. Ferner kann man es so einrichten, daß keine Variable auf der rechten Seite mehr als dreimal auftritt, und $s_i^j \neq s_k^j$ für $i \neq k$. Man ersetze nämlich alle s_i^j durch verschiedene Variable, etwa x_i^j, und füge die Klauseln $E(x_i^j, x_{i'}^{j'})$ hinzu, falls $s_i^j = s_{i'}^{j'}$. Von den Klauseln muß man allerdings auch nicht alle hinzufügen. Man kann sich leicht überlegen, daß für jede Variable x_i^j höchstens 2 Klauseln benötigt werden.

Man kann mit etwas Aufwand den Satz 4.2.7 verallgemeinern.

Theorem 4.2.10 *Es sei $S \subseteq A^*$ durch eine einfache LBG erzeugbar. Dann ist S in **PTIME**.*

Das Hauptergebnis dieses Anschnitts wird sein, daß auch die Umkehrung dieses Satzes gilt. Wir wollen dazu eine Vorüberlegung anstellen. Wir hatten bereits gesehen, daß die Klasse **PTIME** gleich der Klasse **ALOGSPACE** ist. Nun wollen wir das Resultat noch weiter verbessern. Es sei T eine Turingmaschine. T heiße **nur lesend**, falls keiner der Köpfe ein Symbol schreiben kann. Hat eine solche Maschine mehrere Bänder, so erlauben wir dieser Maschine, daß sie die Eingabe auf jedes Band gegeben bekommt. (Ein leeres Band ist nämlich nutzlos, wenn die Maschine nicht schreiben darf.) Alternativ dazu kann man sich vorstellen, daß die Maschine nur ein Band hat, aber beliebig viele Leseköpfe.

Definition 4.2.11 *Es sei $S \subseteq A^*$. Wir sagen, daß S in **ARO** ist, falls es eine alternierende, nur lesende Turingmaschine gibt, welche S akzeptiert.*

Theorem 4.2.12 ***ARO** = **ALOGSPACE**.*

Beweis. Es sei $S \in$ **ARO**. Dann existiert eine alternierende nur lesende Turingmaschine T, welche S akzeptiert. Wir müssen eine logarithmisch beschränkte Turingmaschine bauen, welche S akzeptiert. Wir belassen ein Band als Leseband, die anderen Bänder durch je durch Rechenbänder ersetzt. Auf das Rechenband mit der

Nummer i notieren wir dern Binärkode der Position des $i$ten Lesekopfes von T. Offensichtlich genügt $\log_2 |\vec{x}| + 1$ viel Platz. Die neue Maschine, U, simuliert T wie folgt. Falls T den Kopf Nummer i einen Schritt nach links bewegt, wird der Zähler auf dem Band i von U um eines erniedrigt. Wir der Kopf nach rechts bewegt, so erhöht U den Zähler um eins. Dies erfordert möglicherweise mehr als einen Rechenschritt, aber ist mit endlicher Kontrolle, ohne weiteren Platzbedarf ausführbar. Aller anderen Operationen sind bei T und U analog, insbesondere die Bewegung des 1. Kopfes. Sei nun umgekehrt eine alternierende, logarithmisch beschränkte Turingmaschine U gegeben. Wir wollen eine nur lesende, alternierende Turingmaschine konstruieren, die dieselbe Sprache akzeptiert. Dazu ersetzen wir die Rechenbänder je durch zwei (!) Lesebänder. Betrachten wir der Einfachheit halber das 2. Band von U. Wir wollen es so einrichten, daß auf dem ersten der vier Bänder von T — nennen wir es B — der Lesekopf an der Stelle i steht, falls das 2. Band auf U den Binärkode von i enthält. Dazu notieren wir auf dem zweiten der Bänder von T — nennen wir es C — die Position des Lese/Schreibkopfes von T auf dem 2. Band. Es ist nicht schwer, es so einzurichten, daß C korrekt arbeitet. Jedem Schritt des 2. Kopfes von T entspricht ein Schritt des Kopfes von C. Nun kann aber die Maschine auch auf das Rechenband schreiben. Dies macht die ganze Schwierigkeit aus. Wir müssen zeigen, daß folgende Schritte auf der Maschine berechnet werden können: das Schreiben einer Zahl (0 oder 1) auf das Rechenband an der Stelle i. Arithmetisch gesehen enstpricht das Schreiben einer 1 der folgenden Operation: (a) teste, ob die i Stelle in der Binärdarstellung der Position vom Band $B = 0$ ist, (b) falls ja, so schiebe den Kopf von B um 2^i nach rechts, (c) falls nein, so lasse den Kopf von B, wo er ist. Analog das Schreiben einer 0. Man überlegt sich nun, daß die Maschine mit Hilfe von drei Bändern die Zahl 2^i ausrechnen kann. Zweitens muß man sich überlegen, daß sie in der Lage ist, die $i$te Ziffer in der Binärdarstellung der Position von einem Kopf berechnen kann. Dies ist die Hauptschwierigkeit. Wir skizzieren, wie das geht. Sei p die Position des Kopfes. Dann muß sie 1 so lange verdoppeln (Exponent q mitzählen!), bis sie die größte Zahl $\geq p$ der Form 2^q errechnet hat. Ist $q < i$, so ist die $i$te Ziffer gleich 0. Andernfalls muß sie 2^q abziehen und diese Prozedur wiederholen. $\dashv$

Jetzt zu dem angekündigten Beweis. Wir nehmen an, S sei in **PTIME**. Dann wissen wir jetzt, daß es eine alternierende, nur lesende Turingmaschine T gibt, welche S akzeptiert. Diese Maschine arbeite mit k Bändern. Wir nehmen der Einfachheit an, daß die Maschine in jedem Schritt nur einen Kopf bewegen kann. Wir werden eine $2k + 2$–LBG G konstruieren mit $L(G) = S$. Dazu einige Vorüberlegungen. Ist $\vec{w}$ die Eingabe, so können wir die Position eines beliebigen Lesekopfes durch ein Paar $\langle \vec{x}, \vec{y} \rangle$ kodieren, für das gilt $\vec{x} \cdot \vec{y} = \vec{w}$. Eine Konfiguration ist dann schlicht durch Angabe des Zustands der Maschine sowie durch Angabe von k Paaren $\langle \vec{x}_i, \vec{y}_i \rangle$ mit $\vec{x}_i \cdot \vec{y}_i = \vec{w}$ bestimmt. Unsere Grammatik wird der Maschine Schritt für Schritt folgen. Jedem Zustand q ordnen wir ein Prädikat $q^\star$ zu. Ist q existentiell, so ist dieses Prädikat $2k + 2$–stellig und wir ordnen q die folgenden Regeln zu. Geht q in r über

bei Lesen des Zeichens a und geht die Maschine auf dem Band j nach links, so bekommt G die folgende Regel.

$$q^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad y'_j = a \cdot y_j, \quad x_j = x'_j \cdot a.$$

Geht die Maschine nach rechts, so fügen wir stattdessen die folgende Regel hinzu.

$$q^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad x'_j = x_j \cdot a, \quad y_j = a \cdot y'_j.$$

Falls die Maschine sich nicht bewegt, so fügen wir schließlich diese Regel hinzu.

$$q^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad E(x'_j, x_j), \quad E(y'_j, y_j).$$

Diese Regeln enthalten Prädikate der Form $x'_j = a \cdot x_j$. Diese sind nicht in dem richtigen Format. Wir können diese jedoch für jedes $a \in A$ ein 2–stelliges Prädikat $L^a(x'_j, x_j)$ ersetzen, wobei wir noch folgende Regeln hinzufügen.

$$L^a(\varepsilon, a) \leftarrow . \qquad L^a(xc, yc) \leftarrow L^a(x, y).$$

Dann gilt $G \vdash L^a(\vec{x}, \vec{y})$ immer dann, wenn $\vec{y} = a \cdot \vec{x}$. Ebenso definieren wir $R^a(x, y)$ durch

$$L^a(\varepsilon, a) \leftarrow . \qquad L^a(cx, cy) \leftarrow L^a(x, y).$$

Hier gilt $G \vdash R^a(\vec{x}, \vec{y})$ genau dann, wenn $\vec{y} = \vec{x} \cdot a$. Die Regeln sind einfach.

Man beachte, wie die ersten beiden Stelle in dem Prädikat benutzt wird, um überflüssige Variable zu 'entsorgen'. Ist nun der Zustand q universell, und gibt es genau p Übergänge mit Nachfolgezuständen r_i, $i < p$, (die nicht alle verschieden sein müssen) so wird $q^\star$ ebenfalls $2k + 2$–stellig, und wir führen noch p Symbole $q(i)^\star$ ein, welche jeweils $2k + 2$–stellig sein sollen. Nun wird als erstes die folgende Regel eingeführt:

$$\begin{aligned} q^\star(w, w', x_0, y_0, \ldots, x_{k-1}, y_{k-1}) \leftarrow \quad & q(0)^\star(w, w', x_0, y_0, \ldots, x_{k-1}, y_{k-1}) \\ & q(1)^\star(w, w', x_0, y_0, \ldots, x_{k-1}, y_{k-1}) \\ & \ldots \\ & q(p-1)^\star(w, w', x_0, y_0, \ldots, x_{k-1}, y_{k-1}) \end{aligned}$$

Anschließend verfahren wir wie oben. Besteht der Übergang i darin, daß q in r_i übergeht bei Lesen des Zeichens a und geht die Maschine auf dem Band j nach links, so bekommt G die folgende Regel.

$$q(i)^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r_i^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad L^a(y'_j, y_j), \quad R^a(x_j, x'_j).$$

Geht die Maschine nach rechts, so fügen wir stattdessen die folgende Regel hinzu.

$$q(i)^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad R^a(x'_j, x_j), \quad L^a(y_j, y'_j).$$

Bewegt die Maschine nicht, so nehmen wir

$$q^\star(w, x_j \cdot y_j, x_0, y_0, \ldots, x_{j-1}, y_{j-1}, x'_j, y'_j, x_{j+1}, y_{j+1}, \ldots, x_{k-1}, y_{k-1}) \leftarrow \\ r^\star(w, w, x_0, y_0, \ldots, x_{k-1}, y_{k-1}), \quad E(x'_j, x_j), \quad E(y'_j, y_j).$$

Wiederum haben wir einfache Regeln. Ist q ein akzeptierender Zustand, so nehmen wir die folgende Regel auf.

$$q^\star(w, w', x_0, y_0, \ldots, x_{k-1}, y_{k-1}) \leftarrow .$$

Die letzte Regel, die wir brauchen, ist

$$\mathtt{S}(w) \leftarrow q_0^\star(w, w, \varepsilon, w, \varepsilon, w, \ldots, \varepsilon, w)$$

Dies ist ebenfalls einfach. Denn die Variable w tritt ja auf der linken Seite nur einmal auf. Als letztes müssen wir zeigen, daß $L(G) = S$ ist. Da $S = L(T)$, genügt der Nachweis, daß $L(G) = L(T)$. Es ist $\vec{w} \in L(T)$, falls ein $n \in \omega$ existiert derart, daß T aus der Anfangskonfiguration für $\vec{w}$ in eine akzeptierte Endkonfiguration gerät. Hierbei ist die Anfangskonfiguration wie folgt. Auf allen Bändern steht $\vec{w}$ und die Leseköpfe befinden sich links von der Eingabe. Eine Endkonfiguration ist eine Konfiguration, von welcher aus keine Übergänge mehr existieren. Sie ist akzeptiert, wenn die Maschine in einem universellen Zustand ist.

Wir betrachten folgende Abbildung: dem Tupel $\zeta := q^\star(\vec{w}, \vec{w}, \vec{x}_0, \vec{y}_0, \ldots, \vec{x}_{k-1}, \vec{y}_{k-1})$ ordnen wir die Konfiguration ζ^K zu, in welcher T im Zustand q ist und der Kopf des $i$ten Bandes auf dem nächsten Symbol nach $\vec{x}_i$ positioniert ist. Jetzt gilt:

1. $G \vdash^0 \zeta$ genau dann, wenn ζ^K eine akzeptierte Endkonfiguration ist.
2. Ist q existentiell, so ist genau dann $\zeta \leftarrow \eta$ Instanz einer Regel von G, wenn T η^K aus ζ^K in einem Schritt berechnet.
3. Ist q universell, so ist genau dann ζ aus η_i, $i < p$, in zwei Regelschritten ableitbar wenn T genau die Übergänge $\zeta^K \to \eta_i^K$ berechnet $(i < p)$.

Sei dazu $\vec{w} \in L(G)$. Dies bedeutet, daß $G \vdash^n \mathtt{S}(\vec{w})$, und so

$$G \vdash^{n-1} \zeta := q_0^\star(\vec{w}, \vec{w}, \varepsilon, \vec{w}, \varepsilon, \vec{w}, \ldots, \varepsilon, \vec{w})$$

Dies entspricht der Anfangskonfiguration von T für die Eingabe $\vec{w}$. Wir leiten aus dem oben Gesagten ab, daß, falls $G \vdash^{n-1} \zeta$, so existiert ein $k \leq n$ derart, daß T in k Schritten ζ^K akzeptiert. Ferner gilt: falls T in k Schritten ζ^K akzeptiert, so gilt $G \vdash^{2k} \zeta$. Also haben wir $L(G) = L(T)$.

Theorem 4.2.13 (Groenink) *Genau dann ist S erzeugt durch eine einfache **LBG**, wenn $S \in$ **PTIME**.*

Übung 94. Beweisen Sie Satz 4.2.5. *Hinweis.* Sie müssen die Aktionen der Turingmaschine durch die Grammatik simulieren. Dabei kodieren wir durch die Zeichenkette die Konfiguration, durch die Nichtterminalsymbole die Zustände der Maschine.

Übung 95. Beweisen Sie Satz 4.2.10.

Übung 96. Zeigen Sie, daß $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \in \omega\}$ durch eine einfache 1–LBG erzeugt werden kann.

Übung 97. Es sei $G = \mathtt{S}, N, A, R\rangle$ eine LBG, welche eine Sprache S erzeugt. Es sei ferner U die Sprache aller $\vec{x}$, für die ein $\vec{y}$ mit gleichem Parikh–Bild existiert. (U ist also die Permutationshülle von S.) Sei $G^p := \mathtt{S}^\heartsuit, N \cup \{\mathtt{S}^\heartsuit, A, R^p\rangle$, wo $\mathtt{S}^\heartsuit$ ein Symbol $\notin A \cup N$ ist, und es sei

$$R^p := R \cup \{\mathtt{S}^\heartsuit(x) \leftarrow \mathtt{S}(x), \mathtt{S}^\heartsuit(v \cdot y \cdot x \cdot z) \leftarrow \mathtt{S}^\heartsuit(v \cdot x \cdot y \cdot z)\}$$

Zeigen Sie, daß $L(G^p) = U$.

4.3 Interpretierte Literalbewegungsgrammatiken

In diesem Abschnitt wollen wir uns mit interpretierten LBGs befassen. Die Grundidee hinter einer interpretierten LBG ist denkbar einfach. Jede Regel wird gekoppelt mit einer Vorschrift, wie die Bedeutungen der Elemente der rechten Seite zu der Bedeutung des Element auf der linken Seite zusammengefaßt werden. Wir bringen ein Beispiel. Die folgende Grammatik erzeugt — wie schon erwähnt — die Sprache $\{\mathtt{a}^{2^n} : n \geq 0\}$.

$$S(x \cdot x) \leftarrow S(x). \qquad S(\mathtt{a}) \leftarrow .$$

Wir schreiben nun eine Grammatik, welche alle Paare $\langle \mathtt{a}^{2^n}, n\rangle$ berechnet. Wir nehmen also n als die Bedeutung der Zeichenkette $\mathtt{a}^{2^n}$. Dazu wählen wir für die erste Regel die Funktion $\lambda n.n + 1$ und für die zweite die Konstante 0. Wir werden jetzt die Notation an die bisher übliche anpassen und deshalb wie folgt schreiben:

$$\mathtt{aaaa} : S : 2 \qquad \text{oder auch} \qquad \langle \mathtt{aaaa}, S, 2\rangle$$

Beide Notationen werden verwendet. Sie benennen ein Zeichen mit Exponenten aaaa, mit Typ S und mit Bedeutung 2. Die Regeln schreiben wir jetzt wie folgt:

$$\langle x \cdot x, S, n + 1\rangle \leftarrow \langle x, S, n\rangle. \qquad \langle \mathtt{a}, S, 0\rangle \leftarrow .$$

Diese Grammatik ist nun ganz leicht in eine Zeichengrammatik umformulierbar. Wir definieren einen nullstelligen Modus $\mathtt{A}_0$ und einen einstelligen Modus $\mathtt{A}_1$.

$$\begin{array}{lcl} \mathtt{A}_0 & := & \langle \mathtt{a}, S, 0\rangle \\ \mathtt{A}_1(\langle x, S, n\rangle) & := & \langle x \cdot x, S, n+1\rangle \end{array}$$

Der Strukturterm $\mathtt{A}_1\mathtt{A}_1\mathtt{A}_1\mathtt{A}_0$ definiert zum Beispiel das Zeichen $\langle \mathtt{a}^8, S, 3\rangle$.

Literalbewegungsgrammatiken scheinen sich in dieser Weise als Zeichengrammatiken definieren zu lassen. Doch der Schein trügt. Betrachten wir noch folgende Regel, die wir unserer Grammatik hinzufügen.

$$\langle x\mathtt{ab}y, S, 3n\rangle \leftarrow \langle x\mathtt{aa}y, S, n\rangle$$

Das Problem mit dieser Regel ist, daß die linke Seite nicht eindeutig durch die rechte Seite bestimmt ist. Zum Beispiel können wir aus $\langle \mathtt{aaaa}, S, 2\rangle$ in einem Schritt sowohl $\langle \mathtt{abaa}, S, 6\rangle$ wie auch $\langle \mathtt{aaba}, S, 6\rangle$ und $\langle \mathtt{aaab}, S, 6\rangle$ herleiten. Wir wollen deswegen folgende Vereinbahrung treffen.

Definition 4.3.1 *Es sei*

$$\begin{array}{rl} \rho : T(t^0, t^1, \ldots, t^{p-1}) \leftarrow & U_0(s_0^0, s_0^1, \ldots, s_0^{q_0-1})\ U_1(s_1^0, s_1^1, \ldots, s_1^{q_1-1})\ \ldots \\ & \ldots U_{n-1}(s_{n-1}^0, s_{n-1}^1, \ldots, s_{n-1}^{q_n-1}) \end{array}$$

eine Regel. ρ heißt ***definit****, falls für alle Instanzen dieser Regel gilt: sind die s_i^j gegeben, so sind die t^j eindeutig bestimmt. Eine Literalbewegungsgrammatik heißt* ***definit****, falls jede Regel definit ist.*

Offensichtlich benötigen wir für die Umformung in Zeichengrammatiken, daß wir es mit einer definiten Grammatik zu tun haben. Allerdings ist auch dies noch relativ allgemein. Wir wollen uns daher auf den Fall von einfachen LBGs beschränken. Diese haben den Vorzug, daß die s_i^j Variable über Zeichenketten und die t^j Polynome sind. Diese können wir dann in λ–Notation schreiben. Unsere Grammatik kann jetzt wie folgt spezifiziert werden.

$$\begin{array}{lcl} \mathtt{A}_0 & := & \langle \mathtt{a}, S, 0\rangle \\ \mathtt{A}_1 & := & \langle \lambda x.x \cdot x, S, \lambda n.n+1\rangle \end{array}$$

In manchen Fällen ist die Lage allerdings nicht so einfach. Denn diese Spezifikation funktioniert nur, wenn eine Variable auf der rechten Seite nur einmal auftaucht. Falls eine Variable auf der rechten Seite mehrfach auftritt, können wir die t^j nicht einfach als Polynome auffassen. Sie sind nämlich, wie sich herausstellt, partiell. Ein einfaches Beispiel ist die folgende Regel.

$$C(x) \leftarrow A(x), \quad B(x).$$

Intuitiv würde man hier einfach $\lambda x.x$ ansetzen; allerdings kann man damit nicht erreichen, daß die Zeichenketten auf der rechten Seite gleich sind. Denn gesetzt den Fall, wir wollten einen zweistelligen Modus $\mathtt{C}$ einführen.

$$\mathtt{C}(\langle \vec{x}, \alpha, X\rangle, \langle \vec{y}, \beta, Y\rangle) := \langle \mathtt{C}^{\varepsilon}(\vec{x}, \vec{y}), \mathtt{C}^{\tau}(\alpha, \beta), \mathtt{C}^{\mu}(X, Y)\rangle$$

Dann müssen wir sicherstellen, daß $\mathtt{C}^{\varepsilon}(\vec{x}, \vec{y})$ nur dann definiert ist, wenn $\vec{x} = \vec{y}$. Wir vereinbahren also, daß zusätzlich zur Konkatenation auf A^* auch noch eine 2–stellige Operation I zur Verfügung steht, welche wie folgt definiert ist:

$$I(\vec{x}, \vec{y}) := \begin{cases} \vec{x}, & \text{falls } \vec{x} = \vec{y}, \\ \star, & \text{sonst.} \end{cases}$$

Mit ihrer Hilfe können wir die Regel in einen zweistelligen Modus gießen. Dann setzen wir jetzt einfach $\mathtt{C}^{\varepsilon} := \lambda x.\lambda y.I(x, y)$.

Wir wollen unsere Konzepte erproben, indem wir ein paar Beispiele bringen. Es sei $\vec{x} \in \{\mathtt{L}, \mathtt{O}\}^*$ eine Binärfolge. Diese ist der Binärcode $n^\flat$ einer natürlichen Zahl n. Als Bedeutung dieser Binärfolge wollen wir den Turingcode von n nehmen. Für jede Zahl n ist der Turingcode $n^\sharp$ die Folge $\mathtt{a}^{n+1}$. Hier ist eine Grammatik für die Sprache $\{\langle n^\flat, S, n^\sharp\rangle : n \in \omega\}$. erzeugt.

$$\begin{array}{lcl} \langle x \cdot \mathtt{a}, S, n\rangle & \leftarrow & \langle x, T, n\rangle \\ \langle x \cdot x \cdot \mathtt{a}, T, n \cdot \mathtt{L}\rangle & \leftarrow & \langle x, T, n\rangle \\ \langle x \cdot x, T, n \cdot \mathtt{O}\rangle & \leftarrow & \langle x, T, n\rangle \\ \langle \varepsilon, T, \varepsilon\rangle & \leftarrow & . \end{array}$$

Man beachte, daß die Bedeutungen ebenfalls mit Hilfe von Konkatenation berechnet werden. Anstelle von $\lambda n.2n$ oder $\lambda n.2n + 1$ haben wir also $\lambda x.x \cdot \mathtt{O}$ und $\lambda x.x \cdot \mathtt{L}$.

Wir können auch eine Grammatik schreiben, welche Binärcodes in Turingcodes umrechnet. Dazu vertauschen wir einfach die Rollen von Exponent und Bedeutung; das bedeutet, wir drehen die Tripel einfach um.

Ein etwas schwierigeres Beispiel ist eine Grammatik, welche Tripel $\langle \vec{x} \otimes \vec{y}, S, \vec{z}\rangle$

von Binärzahlen herleitet, bei denen $\vec{z} = \vec{x} + \vec{y}$, wobei $|\vec{x}| \geq |\vec{y}|$.

$$\begin{array}{lcl}
\langle \mathtt{O}x \otimes y, S, z\rangle & \leftarrow & \langle x \otimes y, S, z\rangle \\
\langle \mathtt{L}x \otimes y, S, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, S, z\rangle \\
\langle x \otimes y, S, z\rangle & \leftarrow & \langle x \otimes y, A, z\rangle \\
\langle x \otimes y, S, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, U, z\rangle \\
 & & \\
\langle \mathtt{O}x \otimes \mathtt{O}y, A, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, U, z\rangle \\
\langle \mathtt{O}x \otimes \mathtt{L}y, U, \mathtt{O}z\rangle & \leftarrow & \langle x \otimes y, U, z\rangle \\
\langle \mathtt{L}x \otimes \mathtt{O}y, U, \mathtt{O}z\rangle & \leftarrow & \langle x \otimes y, U, z\rangle \\
\langle \mathtt{L}x \otimes \mathtt{L}y, U, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, U, z\rangle \\
 & & \\
\langle \mathtt{O}x \otimes \mathtt{O}y, A, \mathtt{O}z\rangle & \leftarrow & \langle x \otimes y, A, z\rangle \\
\langle \mathtt{O}x \otimes \mathtt{L}y, A, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, A, z\rangle \\
\langle \mathtt{L}x \otimes \mathtt{O}y, A, \mathtt{L}z\rangle & \leftarrow & \langle x \otimes y, A, z\rangle \\
\langle \mathtt{L}x \otimes \mathtt{L}y, U, \mathtt{O}z\rangle & \leftarrow & \langle x \otimes y, A, z\rangle \\
\langle \varepsilon \otimes \varepsilon, A, \varepsilon\rangle & \leftarrow & .
\end{array}$$

Will man die Beschränkung $|\vec{x}| \geq |\vec{y}|$ aufheben, so muß man anstelle des Startsymbols S ein Startsymbol S' und die folgenden Regeln einführen.

$$\langle x \otimes y, S', z\rangle \leftarrow \langle x \otimes y, S, z\rangle, \qquad \langle x \otimes y, S', z\rangle \leftarrow \langle y \otimes x, S, z\rangle$$

Man überlegt sich, daß $\langle \vec{x} \otimes \vec{y}, A, \vec{z}\rangle$ genau dann herleitbar ist, wenn $|\vec{x}| = |\vec{y}| = |\vec{z}|$, sowie $\vec{x}+\vec{y} = \vec{z}$. Und daß $\langle \vec{x} \otimes \vec{y}, U, \vec{z}\rangle$ genau dann herleitbar ist, wenn $|\vec{x}| = |\vec{y}| = |\vec{z}|$, sowie $\vec{x} + \vec{y} = \mathtt{L}\vec{z}$.

Kommen wir nun auf die Spezifikation von interpretierten LBGn zurück. Zunächst einmal wollen wir betrachten, wie man eine interpretierte LBG als Zeichengrammatik interpretieren kann. Dazu müssen wir lediglich unser Verständnis von den Exponenten zurechtrücken. Bisher hatten wir angenommen, Exponenten sind Zeichenketten. Jetzt müssen wir annehmen, daß sie Vektoren von Zeichenketten sind. Dazu folgende Definition.

Definition 4.3.2 *Es sei A eine endliche Menge, n eine natürliche Zahl. Wir bezeichnen mit $V(A) := \bigcup_{k<n+1}(A^*)^k$ die Menge der Vektoren von Zeichenketten über A. Ferner sei $F^V := \{\varepsilon, 0, \cdot, \otimes, \triangleright, \triangleleft, z, I\}$, $\Omega(\cdot) = \Omega(\otimes) = \Omega(I) = 2$, $\Omega(\triangleright) = \Omega(\triangleleft) = \Omega(z) = 1$; $\Omega(\varepsilon) = \Omega(0) = 0$. Dabei gilt Folgendes:* [1]

- $\langle \vec{x}_i : i < m\rangle \otimes \langle \vec{y}_j : j < n\rangle = \langle \langle \vec{x}_i : i < m\rangle, \langle \vec{y}_j : j < n\rangle\rangle$
- $0 = \langle\,\rangle$ *ist die leere Folge.*

[1] Die Klammern $\langle a, b\rangle$ sind hier rein metalinguistisch. Sie bezeichnen *Folgen* von Objekten. Daher ist zum Beispiel $\langle\langle x, y\rangle, z\rangle = \langle x, y, z\rangle$.

* $\cdot$ *ist die übliche Verkettung von Zeichenketten, mithin nicht definiert auf Vektoren der Länge* $\neq 1$.
* ε *ist die leere Zeichenkette.*
* $\triangleright\langle \vec{x}_i : i < m\rangle = \langle \vec{x}_i : i < m-1\rangle$, $\triangleleft\langle \vec{x}_i : i < m\rangle = \langle \vec{x}_i : 0 < i < m\rangle$.
* $z(\mathfrak{x}) = \star$, *falls* $\mathfrak{x}$ *keine Zeichenkette ist;* $z(\vec{x}) = \vec{x}$ *sonst.*
* $I(\mathfrak{x}, \mathfrak{y}) = \mathfrak{x}$, *falls* $\mathfrak{x} = \mathfrak{y}$, *und* $I(\mathfrak{x}, \mathfrak{y}) = \star$ *sonst.*

Die entstehende (partielle) Algebra heißt die ***Vektoralgebra*** *über* A *und wird mit* $\mathfrak{V}(A)$ *bezeichnet.*

In dieser Algebra gelten unter anderem folgende Gleichungen.

$$\begin{array}{lcl} \mathfrak{x} \otimes 0 & = & \mathfrak{x} \\ 0 \otimes \mathfrak{x} & = & \mathfrak{x} \\ \mathfrak{x} \otimes (\mathfrak{y} \otimes \mathfrak{z}) & = & (\mathfrak{x} \otimes \mathfrak{y}) \otimes \mathfrak{z} \\ \triangleright(\mathfrak{x} \otimes z(\mathfrak{y})) & = & \mathfrak{x} \\ \triangleleft(z(\mathfrak{y}) \otimes \mathfrak{x}) & = & \mathfrak{x} \end{array}$$

Die vierte und fünfte Gleichung gelten jeweils unter dem Vorbehalt, daß $z(\mathfrak{y})$ definiert ist. Ein Vektor $\mathfrak{x}$ hat genau dann die Länge m, wenn $z(\triangleright^{m-1}\mathfrak{x})$ definiert ist. In diesem Fall sind auch $z(\triangleright^{m-(i+1)} \triangleleft^i \mathfrak{x})$ für alle i definiert, und sie sind die Projektionsfunktionen. Es gilt dann:

$$\mathfrak{x} = \triangleright^{m-1}\mathfrak{x} \otimes \triangleright^{m-2} \triangleleft \mathfrak{x} \otimes \ldots \otimes \triangleright \triangleleft^{m-2} \mathfrak{x} \otimes \triangleleft^{m-1}\mathfrak{x}$$

Sämtliche Polynomfunktionen, welche im Folgenden auftauchen, können in dieser Algebra definiert werden. Als Grundlage dient folgender Satz.

Theorem 4.3.3 *Es sei* $p : (A^*)^m \to A^*$ *eine Funktion, welche ein Polynom in* $\cdot$ *und* I *ist. Dann existiert ein Vektorpolynom* $\mathfrak{q} : V(A) \to V(A)$ *derart, daß*

1. $\mathfrak{q}(\mathfrak{x})$ *ist nur dann definiert, wenn* $\mathfrak{x} \in (A^*)^m$.
2. *Ist* $\mathfrak{x} \in (A^*)^m$, *und* $\mathfrak{x} = \langle \vec{x}_i : i < m\rangle$, *so gilt* $\mathfrak{q}(\mathfrak{x}) = p(\vec{x}_0, \ldots, \vec{x}_{m-1})$.

Beweis. Sei p gegeben. Wir nehmen an, daß eine der Variablen mindestens einmal auftritt. (Ansonsten ist $p = \varepsilon$, und dann setzen wir $q := \varepsilon$.) Es entstehe q durch Ersetzen von x_i in p durch $z(\triangleright^{m-(i+1)} \triangleleft^i \mathfrak{x})$, für alle $i < m$. Dies definiert q. (Dies ist wohldefiniert, denn die Symbole ε, $\cdot$, I sind auch in der Signatur F^V.) Sei nun $\mathfrak{x}$ gegeben. Wie oben angemerkt, ist q auf $\mathfrak{x}$ nur dann definiert, wenn $\mathfrak{x}$ die Länge m hat. In

diesem Fall ist $\mathfrak{x} = \langle \vec{x}_i : i < n \rangle$ für gewisse $\vec{x}_i$, und wir haben $\vec{x}_i = z(\triangleright^{m-(i+1)} \triangleleft^i \mathfrak{x})$. Da die Symbole ε, $\cdot$ und I auf Zeichenketten in beiden Algebren (der der Zeichenketten und der der Vektoren) übereinstimmen, gilt $q(\mathfrak{x}) = q(\vec{x}_0, \ldots, \vec{x}_{m-1})$. $\dashv$

Ist nun $p : (A^*)^m \to (A^*)^n$ eine Polynomfunktion, so bedeutet dies, daß Polynome p_i, $i < n$, existieren mit

$$p(\vec{x}_0, \ldots, \vec{x}_{m-1}) = \langle p_i(\vec{x}_0, \ldots, \vec{x}_{m-1}) : i < n \rangle$$

Wir können also unsere Polynome auf Zeichenketten durch Vektorpolynome ablösen. Dies vereinfacht die Darstellung von LBGs sehr. Wir können jetzt eine Regel wie folgt notieren.

$$\langle q(\mathfrak{x}_0, \ldots, \mathfrak{x}_{m-1}), A, f(X_0, \ldots, X_{m-1}) \rangle \leftarrow \langle \mathfrak{x}_0, B_0, X_0 \rangle \quad \ldots \quad \langle \mathfrak{x}_{m-1}, B_{m-1}, X_{m-1} \rangle$$

Wir wollen noch einen Schritt weitergehen und die LBGs wir Kategorialgrammatiken behandeln. Dazu gehen wir zunächst zur einer Chomsky–Normalform über. Dies birgt nun wieder kleine Schwierigkeiten. Denn zu einer k–LBG existiert nicht unbedingt eine k–LBG in Chomsky–Normalform (siehe dazu die Übungen). Es existiert aber eine k'–LBG in Chomsky–Normalform für ein effektiv bestimmbares k'. Dazu betrachten wir unsere Regel. Wir führen neue Symbole Z_i, $i < m-2$, ein und ersetzen unsere Regel durch die folgenden:

$$\begin{array}{lcl}
\langle \mathfrak{x}_0 \otimes \mathfrak{x}_1, Z_0, X_0 \times X_1 \rangle & \leftarrow & \langle \mathfrak{x}_0, B_0, X_0 \rangle, \quad \langle \mathfrak{x}_1, B_1, X_1 \rangle \\
\langle \mathfrak{y}_0 \otimes \mathfrak{x}_2, Z_1, Y_0 \times X_2 \rangle & \leftarrow & \langle \mathfrak{y}_0, Z_0, Y_0 \rangle, \quad \langle \mathfrak{x}_2, B_2, X_2 \rangle \\
& \ldots & \\
\langle \mathfrak{y}_{m-4} \otimes \mathfrak{x}_{m-2}, Z_{m-3}, Y_{m-4} \times X_{m-2} \rangle & \leftarrow & \langle \mathfrak{y}_{m-4}, Z_{m-4}, Y_{m-4} \rangle, \\
& & \quad \langle \mathfrak{x}_{m-2}, B_{m-2}, X_{m-2} \rangle \\
\langle q^*(\mathfrak{y}_{m-3} \otimes \mathfrak{x}_{m-1}), A, f^*(Y_{m-3} \times X_{m-1}) \rangle & \leftarrow & \langle \mathfrak{y}_{m-3}, Z_{m-3}, Y_{m-3} \rangle, \\
& & \quad \langle \mathfrak{x}_{m-1}, B_{m-1}, X_{m-1} \rangle
\end{array}$$

Hierbei werden q^* und f^* gerade so gewählt, daß gilt:

$$\begin{array}{lcl}
q^*(\mathfrak{x}_0 \otimes \ldots \otimes \mathfrak{x}_{m-1}) & = & q(\mathfrak{x}_0, \ldots, \mathfrak{x}_{m-1}) \\
f^*(X_0 \times \ldots \times X_{m-1}) & = & f(X_0, \ldots, X_{m-1})
\end{array}$$

Es ist nicht schwer zu sehen, wie diese Funktionen durch Polynome definiert werden können. Wir können also im Folgenden annehmen, daß wir höchstens zweistellige Regeln haben. Nullstellige Regeln entsprechen den terminalen Regeln einer kontextfreien Grammatik. Eine einstellige Regel hat die folgende Form

$$\langle q(\mathfrak{x}), C, f(X) \rangle \leftarrow \langle \mathfrak{x}, A, X \rangle$$

Wir behalten das Zeichen $\langle \mathfrak{x}, A, X \rangle$, und führen ein neues Zeichen $\mathtt{Z}_\rho$ ein, das wie folgt aussieht:

$$\mathtt{Z}_\rho := \langle \lambda \mathfrak{x}.q(\mathfrak{x}), C/A, \lambda x.f(x) \rangle$$

Wir vereinbahren, daß es lediglich einen zweistelligen Modus $\mathtt{C}$ geben soll, welcher wie folgt definiert ist:

$$\mathtt{C}(\langle p, A, X\rangle, \langle q, B, Y\rangle) := \langle p(q), A \cdot B, (XY)\rangle$$

Dies ist das Schema der Applikation wie in der Kategorialgrammatik. Ein Unterschied besteht jedoch: das Polynom p ist nicht unbedingt die Konkatenation. Ferner müssen wir jetzt nicht mehr zwei unterschiedliche Modi annehmen, da wir die Möglichkeit haben, p entweder als $\lambda x.x \cdot y$ oder als $\lambda x.y \cdot x$ zu wählen. Applikation ist damit von der akzidentiellen Reihenfolge unabhängig geworden. Viele andere Operationen können hier stehen, zum Beispiel die Verdopplung. Unsere eingangs besprochene Grammatik wird jetzt durch folgende zwei Modi bestimmt.

$$\begin{array}{lcl} \mathtt{D}_0 & := & \langle \mathtt{a}, S, 0\rangle \\ \mathtt{D}_1 & := & \langle \lambda x.x \cdot x, S/S, \lambda n.n+1\rangle \end{array}$$

Dem Strukturterm $\mathtt{A}_1\mathtt{A}_1\mathtt{A}_1\mathtt{A}_0$ entspricht jetzt der Strukturterm

$$\mathtt{C}\mathtt{D}_1\mathtt{C}\mathtt{D}_1\mathtt{C}\mathtt{D}_1\mathtt{D}_0$$

Auf diese Weise ist die Grammatik zu einer **AB**–Grammatik geworden, mit den einen Unterschied, daß die Behandlung der Zeichenketten jetzt explizit festgelegt werden darf (und muß).

Es bleiben aber noch die zweistelligen Regeln. Eine zweistellige Regel hat folgende Form:

$$\langle q(\mathfrak{x}, \mathfrak{y}), C, f(X, Y)\rangle \leftarrow \langle \mathfrak{x}, A, X\rangle, \quad \langle \mathfrak{y}, B, Y\rangle$$

Wir behalten die Zeichen auf der rechten Seite und führen ein neues Zeichen ein.

$$\mathtt{Z}_\rho := \langle \lambda\mathfrak{y}.\lambda\mathfrak{x}.q(\mathfrak{x}, \mathfrak{y}), (C/A)/B, \lambda y.\lambda x.f(x, y)\rangle$$

4.4 Diskontinuität

In diesem Kapitel wollen wir einen wichtigen Typ von Grammatiken studieren, die sogenannten linearen kontextfreien Ersetzungssysteme (Englisch *Linear Context–Free Rewrite Systems*, kurz **LCFRS**). Dies wurden in [45] im Detail studiert. Wir nennen sie hier schlicht lineare LBGn (was nur ein Unterschied im Namen ist).

Definition 4.4.1 *Eine k–LBG heiße **linear**, wenn sie eine einfache k–LBG ist und jede nicht 0–stellige Regel abwärts nichtlöschend und abwärts linear, während 0–stellige Regeln die Form $X(a) \leftarrow .$ mit $a \in A$ haben.*

Mit anderen Worten, falls wir eine Regel dieser Form haben

$$A(t_0, \ldots, t_{k-1}) \leftarrow B_0(s_0^0, \ldots, s_{k-1}^0) \quad \ldots \quad B_{n-1}(s_0^{n-1}, \ldots, s_0^{n-1}) \,,$$

so ist $s_i^j = x_i^j$ für eine Variable x_i^j für jedes $i < k$ und $j < n$; diese Variablen sind paarweise verschieden, und es ist $\prod_{i<k} t_i$ ein Produkt dieser Variablen (also keine Konstanten treten auf), in welchem jede Variable genau einmal auftritt. Im Falle $k = 1$ bekommen wir damit genau die kontextfreien Grammatiken (in etwas verschleierter Form). Denn jetzt ist eine Regel der Form

$$A(\prod_{i<n} x_{\pi(n)}) \leftarrow B_0(x_0) \quad B_1(x_1) \ldots B_{n-1})$$

wobei π eine Permutation der Zahlen $< n$ ist. Ist $\rho := \pi^{-1}$ die zu π inverse Permutation, so können wir diese Regel auch so schreiben:

$$A(\prod_{i<n} x_i) \leftarrow B_{\rho(0)}(x_0) \quad B_{\rho(1)}(x_1) \ldots B_{\rho(0)}(x_{n-1})$$

(Dazu ersetzen wir die Variable x_i durch die Variable $x_{\rho(i)}$ für alle i. Anschließend vertauschen wir die Reihenfolge der B_i. Dies ist ohnehin bei LBGn unerheblich.) Dies ist, wie man leicht sieht, gerade die Form einer kontextfreien Regel. Denn es ist

$$\prod_{i<n} x_i = x_0 x_1 \ldots x_{n-1}$$

Die Regel besagt also, daß, falls wir Konstituenten $\vec{x}_i$ vom Typ $B_{\rho(i)}$ haben für $i < n$, so ist $\prod_{i<n} \vec{x}_i$ eine Konstituente vom Typ A.

Der nächst kompliziertere Fall ist $k = 2$. Dieser Fall führt zu einer Klasse von Grammatiken, welche ursprünglich in anderer Notation eingeführt wurden und sich als mächtig genug erwiesen haben, bekannte nicht kontextfreie Sprachen wie das Schweizerdeutsche zu erzeugen. In linearen 2–LGBn dürfen Regeln der folgende Gestalt auftreten.

$$\begin{array}{lcll} A(x_1x_2, y_1y_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \\ A(x_2x_1, y_1y_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \\ A(y_1x_1y_2, x_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \\ A(x_1, y_1x_2y_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \end{array}$$

Die folgenden Regeln sind jedoch ausgeschlossen.

$$\begin{array}{lcll} A(x_1, y_1y_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \\ A(x_2x_2x_1, y_1y_2) & \leftarrow & B(x_1, y_1) & C(x_2, y_2) \end{array}$$

Die erste ist aufwärts löschend, die zweite nicht aufwärts linear. Die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n : n \in \omega\}$ ist durch eine lineare 2–LBG erzeugbar, die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n : n \in \omega\}$

jedoch nicht. Die zweite Tatsache wird aus einem weiter unten bewiesenen Pumplemma folgen. Für die erste Behauptung schreiben wir folgende Grammatik als Beweis hin.

$$\begin{array}{lcl}
\mathtt{S}(y_0x_0y_1, z_0x_1z_1) & \leftarrow & \mathtt{S}(x_0, x_1) \quad \mathtt{A}(y_0, y_1) \quad \mathtt{B}(z_0, z_1). \\
\mathtt{S}(\varepsilon, \varepsilon) & \leftarrow & . \\
\mathtt{A}(\mathtt{a}, \mathtt{b}) & \leftarrow & . \\
\mathtt{B}(\mathtt{c}, \mathtt{d}) & \leftarrow & .
\end{array}$$

Dies beweist, daß 2–lineare LBGn echt stärker sind als kontextfreie Grammatiken. Als weiteres Beispiel wollen wir das Schweizerdeutsche aus Kapitel 2.6 hervorholen. Wir definieren folgende Grammatik.

$$\begin{array}{lcl}
\mathtt{NPa}(\mathtt{d'chind}) & \leftarrow & . \\
\mathtt{NPd}(\mathtt{em\ Hans}) & \leftarrow & . \\
\mathtt{NPs}(\mathtt{Jan}) & \leftarrow & . \\
\mathtt{NPa}(\mathtt{es\ huus}) & \leftarrow & . \\
\mathtt{Vdr}(\mathtt{hälfed}) & \leftarrow & . \\
\mathtt{Vf}(\mathtt{lönd}) & \leftarrow & . \\
\mathtt{Var}(\mathtt{laa}) & \leftarrow & . \\
\mathtt{Van}(\mathtt{aastriche}) & \leftarrow & . \\
\mathtt{C}(\mathtt{das}) & \leftarrow & . \\
\mathtt{NPs}(\mathtt{mer}) & \leftarrow & . \\
\mathtt{Vc}(\mathtt{säit}) & \leftarrow & . \\
\mathtt{S}(xy) & \leftarrow & \mathtt{NPs}(x) \quad \mathtt{VP}(y) \\
\mathtt{VP}(xy) & \leftarrow & \mathtt{Vc}(x) \quad \mathtt{CP}(y) \\
\mathtt{CP}(xyz_0z_1u) & \leftarrow & \mathtt{C}(x) \quad \mathtt{NPs}(y) \quad \mathtt{VI}(z_0, z_1) \quad \mathtt{Vf}(u) \\
\mathtt{VI}(xz_0, yz_1) & \leftarrow & \mathtt{NPa}(x) \quad \mathtt{Var}(y) \quad \mathtt{VI}(z_0, z_1) \\
\mathtt{VI}(x, y) & \leftarrow & \mathtt{NPa}(x) \quad \mathtt{Van}(y) \\
\mathtt{VI}(xz_0, yz_1) & \leftarrow & \mathtt{NPd}(x) \quad \mathtt{Vdr}(y) \quad \mathtt{VI}(z_0, z_1)
\end{array}$$

Diese Grammatik ist einigermaßen realistisch auch in Bezug auf die Konstituentenstrukturen, von denen noch weiter unten die Rede sein wird. Der Einfachheit halber haben wir die Stelligkeit der Prädikate variiert. Man beachte insbesondere die letzten beiden Regeln. Diese sind der eigentliche Motor. Sie erzeugen die für das Schweizerdeutsche typischen Abfolgen der Infinitive. Man kann nämlich Folgendes ableiten

$$\begin{array}{l}
\mathtt{VI}(\mathtt{d'chind}\ z_0, \mathtt{laa}\ z_1) \leftarrow \mathtt{VI}(z_0, z_1). \\
\mathtt{VI}(\mathtt{em\ Hans}\ z_0, \mathtt{hälfe}\ z_1) \leftarrow \mathtt{VI}(z_0, z_1). \\
\mathtt{VI}(\mathtt{d'chind}, \mathtt{aastriche}) \leftarrow . \\
\mathtt{VI}(\mathtt{es\ huus}, \mathtt{aastriche}) \leftarrow .
\end{array}$$

Aber man hat zum Beispiel nicht

$$\mathtt{VI}(\mathtt{em\ Hans}, \mathtt{laa}) \leftarrow .$$

Daraus leitet man induktiv ab

$$\texttt{VI}(\texttt{d'chind em Hans } z_0, \texttt{laa hälfe } z_1) \leftarrow \texttt{VI}(z_0, z_1).$$
$$\texttt{VI}(\texttt{em Hans es huus } z_0, \texttt{hälfe laa } z_1) \leftarrow \texttt{VI}(z_0, z_1).$$

Die Sätze des Schweizerdeutschen aus Abschnitt 2.6 sind also ableitbar und einige mehr, welche alle grammatisch sind.

Lineare LBGn kann man noch durch die Art der Vektorpolynome, die in ihnen vorkommen, charakterisieren. Wir wollen dies exemplarisch für lineare 2–LBGn vormachen, und auch hier nur für höchstens zweistellige Regeln. Beginnen wir mit den 1–stelligen Regeln. Diese können folgende Vektorpolynome benutzen:

$$\begin{array}{lcl}
\mathfrak{i}(x_0, x_1) & := & \langle x_0, x_1\rangle \\
\mathfrak{p}_X(x_0, x_1) & := & \langle x_1, x_0\rangle \\
\mathfrak{p}_F(x_0, x_1) & := & \langle x_0x_1, \varepsilon\rangle \\
\mathfrak{p}_G(x_0, x_1) & := & \langle x_1x_0, \varepsilon\rangle \\
\mathfrak{p}_H(x_0, x_1) & := & \langle \varepsilon, x_0x_1\rangle \\
\mathfrak{p}_K(x_0, x_1) & := & \langle \varepsilon, x_1x_0\rangle
\end{array}$$

Es gelten dann unter anderem folgende Gleichungen.

$$\begin{array}{lcl}
\mathfrak{p}_X(\mathfrak{p}_X(x_0, x_1)) & = & \mathfrak{i}(x_0, x_1) \\
\mathfrak{p}_G(x_0, x_1) & = & \mathfrak{p}_F(\mathfrak{p}_X(x_0, x_1)) \\
\mathfrak{p}_K(x_0, x_1) & = & \mathfrak{p}_H(\mathfrak{p}_X(x_0, x_1))
\end{array}$$

Dies bedeutet, daß man $\mathfrak{p}_G$ zur Verfügung hat, wenn man $\mathfrak{p}_X$ und $\mathfrak{p}_F$ hat, daß man p_F hat, wenn man auch $\mathfrak{p}_X$ und $\mathfrak{p}_G$ hat, und so weiter. Bei den 2–stelligen Polynomen wird es schon reichlich unübersichtlich. Deswegen wollen wir annehmen, wir hätten sämtliche 1–stelligen Polynome. Ein zweistelliges Vektorpolynom ist von der Form $\langle p_0(x_0, x_1, y_0, y_1), p_1(x_0, x_1, y_0, y_1)\rangle$ derart, daß $q := p_0 \cdot p_1$ linear ist. Dann können wir annehmen, daß in $q(x_0, x_1, y_0, y_1)$ immer x_0 links von x_1 und y_0 links von y_1 steht. Ferner kann man annehmen, daß x_0 links von y_0 steht (ansonsten vertausche man die x_i mit den y_i. Ferner entsteht p_0 und p_1 durch Aufteilung von q in zwei Hälften. Wir brauchen also nur die q aufzulisten; für das Vektorpolynom existieren dann insgesamt 5 (!) Möglichkeiten. Dies ergibt nach Vereinfachung folgende Polynome:

$$\begin{array}{lcl}
q_C(x_0, x_1, y_0, y_1) & := & x_0x_1y_0y_1 \\
q_W(x_0, x_1, y_0, y_1) & := & x_0y_0x_1y_1 \\
q_Z(x_0, x_1, y_0, y_1) & := & x_0y_0y_1x_1
\end{array}$$

Betrachten wir etwa q_W. Daraus erhalten wir folgende Vektorpolynome:

$$\begin{array}{lcl}
\mathfrak{q}_{W0}(x_0, x_1, y_0y_1) & := & \langle \varepsilon, x_0y_0x_1y_1\rangle \\
\mathfrak{q}_{W1}(x_0, x_1, y_0y_1) & := & \langle x_0, y_0x_1y_1\rangle \\
\mathfrak{q}_{W2}(x_0, x_1, y_0y_1) & := & \langle x_0y_0, x_1y_1\rangle \\
\mathfrak{q}_{W3}(x_0, x_1, y_0y_1) & := & \langle x_0y_0x_1, y_1\rangle \\
\mathfrak{q}_{W4}(x_0, x_1, y_0y_1) & := & \langle x_0y_0x_1y_1, \varepsilon\rangle
\end{array}$$

Wir werden in den Übungsaufgaben zeigen lassen, wie sich die Polynome in Grammatiken unterschiedlich verhalten. Wir sagen nun, eine lineare LBG habe Polynombasis Q, falls in den Regeln dieser Grammatik nur Vektorpolynome aus Q verwendet werden. Es ist leicht zu sehen, daß wenn $\mathtt{q}$ ein aus Q darstellbares Vektorpolynom ist, so kann man $\mathtt{q}$ zu Q hinzunehmen, ohne die generative Kapazität zu ändern. Man mache sich im Übrigen auch klar, daß es nichts ausmacht, wenn das Vektorpolynom auch Konstanten enthält. Haben wir zum Beispiel die Regel

$$\mathtt{X}(\mathtt{a}x\mathtt{bc}y) \leftarrow \mathtt{X}(x) \quad \mathtt{Z}(y)$$

so können wir diese durch die folgenden Regeln ersetzen.

$$\begin{array}{lcl} \mathtt{X}(uxvwy) & \leftarrow & \mathtt{A}(u) \quad \mathtt{X}(x) \quad \mathtt{B}(v) \quad \mathtt{C}(w) \quad \mathtt{Z}(y) \\ \mathtt{A}(\mathtt{a}) & \leftarrow & . \\ \mathtt{B}(\mathtt{b}) & \leftarrow & . \\ \mathtt{C}(\mathtt{c}) & \leftarrow & . \end{array}$$

Dies ist in Beweisen von Vorteil. Wir machen auf ein paar generelle Eigenschaften von Sprachen erzeugt durch lineare LBGs aufmerksam.

Proposition 4.4.2 (Vijay–Shanker & Weir & Joshi) *Sei G eine lineare k–LBG. Dann ist $L(G)$ semilinear.*

Ein spezieller Typ von linearen 2–LBGn sind die sogenannten Kopfgrammatiken. Diese Grammatiken wurden von Carl Pollard in [41] eingeführt. Die Zeichenketten, die dort manipuliert werden, sind von der Form $\vec{x}a\vec{y}$, wo $\vec{x}$ und $\vec{y}$ Zeichenketten, und $a \in A$. Man spricht in diesem Zusammenhang von diesem Vorkommen von a in der Zeichenkette als dem ausgezeichneten *Kopf*. Diesen Kopf markieren wir in der Zeichenkette durch Unterstreichen. Diese Zeichenketten heißen jetzt *markiert*. Folgende Manipulationsregeln sind gestattet.

$$\begin{array}{lcl} h_{C1}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}\underline{a}\vec{w}\vec{y}b\vec{z} \\ h_{C2}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}a\vec{w}\vec{y}\underline{b}\vec{z} \\ h_{L1}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}\underline{a}\vec{y}b\vec{z}\vec{w} \\ h_{L2}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}a\vec{y}\underline{b}\vec{z}\vec{w} \\ h_{R1}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}\vec{y}b\vec{z}\underline{a}\vec{w} \\ h_{R2}(\vec{v}\underline{a}\vec{w}, \vec{y}\underline{b}\vec{z}) & := & \vec{v}\vec{y}\underline{b}\vec{z}a\vec{w} \end{array}$$

Da man gerne auch zulassen möchte, daß der ausgezeichnete Kopf leer ist, hat man sich darauf verständigt, anstelle von markierten Zeichenketten einfach 2–Vektoren von Zeichenketten zu nehmen. Der markierte Kopf ist also das Komma. Das führt uns zu folgender Definition.

Definition 4.4.3 *Eine **Kopfgrammatik** ist eine lineare 2–LBG mit folgender Polynombasis.*

$$\begin{array}{lcl}
\mathfrak{p}_{C1}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0, x_1 y_0 y_1\rangle \\
\mathfrak{p}_{C2}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0 x_1 y_0, y_1\rangle \\
\mathfrak{p}_{L1}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0, y_0 y_1 x_1\rangle \\
\mathfrak{p}_{L2}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0 y_0 y_1, x_1\rangle \\
\mathfrak{p}_{R1}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0 y_0 y_1, x_1\rangle \\
\mathfrak{p}_{R2}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) & := & \langle x_0 y_0, y_1 x_1\rangle
\end{array}$$

Man beachte hierbei, daß es keinerlei 1–stellige Polynome gibt. Indem man jedoch einige Komponenten leer läßt, kann man einige einstellige Polynome basteln, nämlich I, p_F und p_H, andere jedoch nicht, da die Reihenfolge der Komponenten eines Vektors immer gleich bleiben. So hat man etwa

$$\mathfrak{p}_{C2}(\langle x_0, x_1\rangle, \langle y_0, y_1\rangle) = \langle x_0 x_1, \varepsilon\rangle = \mathfrak{p}_F(x_0, x_1)$$

Zunächst einmal wollen wir uns der Strukturbeschreibung widmen. Es sei dazu G eine k–lineare Grammatik. Falls nun $G \vdash A(\vec{x}_0, \ldots, \vec{x}_{k-1})$ gilt, so heißt dies, daß das k–Tupel $\langle \vec{x}_i : i < k\rangle$ eine Konstituente vom Typ A bilden kann. Um dies zu präzisieren, werden wir noch etwas Terminologie bereitstellen. Eine **Ableitung** in einer LBG ist eine Folge

$$\Gamma = \langle (\ell_i, \rho_i, H_i) : i < p\rangle$$

wo jedes H_i, $i < p$, eine endliche Folge von Literalen ist, $\ell_i < |H_i|$, ρ_i der Name einer Regel, H_i aus H_{i+1} durch Anwendung von ρ_{i+1} wie folgt hervorgeht.

$$H_i = \langle L_j : j < r\rangle; H_{i+1} = \langle L'_j : j < r + p(\rho_{i+1})\rangle$$

(man erinnere sich an die Definition der Produktivität einer Regel) sowie

$$L_{\ell_{i+1}} \leftarrow L'_{\ell_{i+1}} \quad L'_{\ell_{i+1}+1} \ldots L'_{\ell_{i+1}+p(\rho_{i+1})}$$

Instanz einer Regel (nicht notwendig eine Grundinstanz!) sowie

$$L_j = L'_j \quad (j < \ell_{i+1}), \qquad L_j = L'_{j+p(\rho_{i+1}} \quad (j > \ell_{i+1})$$

In diesem Zusammenhang heißt L_ℓ das **Hauptliteral** von H_i. Ist L_{p-1} eine Folge von Instanzen 0–stelliger Regeln, so ist Γ **vollständige Ableitung**. Eine Beispiel wird dies verdeutlichen. Man betrachte die oben angegebene Grammatik, welche die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n : n \in \omega\}$ erzeugt. Wir formulieren sie leicht um. Die folgende

Grammatik nennen wir $G^{\heartsuit}$.

$$\begin{array}{lcll}
\rho_0 & : & \mathtt{S}(y_0x_0y_1, z_0x_1z_1) & \leftarrow \quad \mathtt{S}(x_0, x_1) \quad \mathtt{X}(y_0, y_1) \quad \mathtt{Y}(z_0, z_1). \\
\rho_1 & : & \mathtt{S}(\varepsilon, \varepsilon) & \leftarrow \quad . \\
\rho_2 & : & \mathtt{X}(x, y) & \leftarrow \quad \mathtt{A}(x) \quad \mathtt{B}(y) \\
\rho_3 & : & \mathtt{A}(\mathtt{a}) & \leftarrow \quad . \\
\rho_4 & : & \mathtt{B}(\mathtt{b}) & \leftarrow \quad . \\
\rho_5 & : & \mathtt{Y}(x, y) & \leftarrow \quad \mathtt{C}(x) \quad \mathtt{D}(y) \\
\rho_6 & : & \mathtt{C}(\mathtt{c}) & \leftarrow \quad . \\
\rho_7 & : & \mathtt{D}(\mathtt{d}) & \leftarrow \quad .
\end{array}$$

Folgendes ist eine vollständige Ableitung.

$$\begin{array}{l}
\langle (0, \rho_0, \langle \mathtt{S}(\mathtt{aabb}, \mathtt{ccdd}) \rangle), \\
\ (0, \rho_0, \langle \mathtt{S}(\mathtt{ab}, \mathtt{cd}), \mathtt{X}(\mathtt{a}, \mathtt{b}), \mathtt{Y}(\mathtt{c}, \mathtt{d}) \rangle), \\
\ (1, \rho_2, \langle \mathtt{S}(\mathtt{ab}, \mathtt{cd}), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{Y}(\mathtt{c}, \mathtt{d}) \rangle), \\
\ (3, \rho_5, \langle \mathtt{S}(\mathtt{ab}, \mathtt{cd}), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{C}(\mathtt{c}), \mathtt{D}(\mathtt{d}) \rangle), \\
\ (0, \rho_0, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{X}(\mathtt{a}, \mathtt{b}), \mathtt{Y}(\mathtt{c}, \mathtt{d}), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{C}(\mathtt{c}), \mathtt{D}(\mathtt{d}) \rangle), \\
\ (1, \rho_2, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{Y}(\mathtt{c}, \mathtt{d}), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{C}(\mathtt{c}), \mathtt{D}(\mathtt{d}) \rangle), \\
\ (3, \rho_5, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{C}(\mathtt{c}), \mathtt{D}(\mathtt{d}), \mathtt{A}(\mathtt{a}), \mathtt{B}(\mathtt{b}), \mathtt{C}(\mathtt{c}), \mathtt{D}(\mathtt{d}) \rangle) \rangle
\end{array}$$

Aber auch folgendes ist eine Ableitung:

$$\langle (0, \rho_0, \langle \mathtt{S}(xy, zu) \rangle, (0, \rho_0, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{X}(x, z), \mathtt{Y}(y, u) \rangle) \rangle$$

(Dies ist einfach eine Instanz einer Regel, einmal angewendet. Man beachte, daß 0 und ρ_0 in dem ersten Folgenglied nichts zu bedeuten haben; sie sind nur aus Uniformitätsgründen anwesend.) Besteht L_0 aus einem einzigen Literal, so sagen wir auch, Γ sei eine **Ableitung von** L_0. Nun sei G k–linear; ferner nehmen wir an, 0–stellige Regeln haben die Form $X(a)$, $a \in A$. Es sei Γ gegeben, und Γ sei Ableitung von einer Grundinstanz eines Literals. Dann ersetzen wir, wo immer in Γ ein $X(a)$ vorkommt, eine Variable anstelle von a ein; bei jeder Einsetzung nehmen wir eine jeweils andere Variable. Dann ersetzen wir von rechts nach links die Vorkommen dieser Buchstaben durch die entsprechende Variable, sodaß eine Folge Γ' entsteht, welche wiederum ein Beweis ist.

$$\begin{array}{l}
\langle (0, \rho_0, \langle \mathtt{S}(x_4x_0x_1x_5, x_6x_2x_3x_7) \rangle), \\
\ (0, \rho_0, \langle \mathtt{S}(x_0x_1, x_2x_3), \mathtt{X}(x_4, x_5), \mathtt{Y}(x_6, x_7) \rangle), \\
\ (1, \rho_2, \langle \mathtt{S}(x_0x_1, x_2x_3), \mathtt{A}(x_4), \mathtt{B}(x_5), \mathtt{Y}(x_6, x_7) \rangle), \\
\ (3, \rho_5, \langle \mathtt{S}(x_0x_1, x_2x_3), \mathtt{A}(x_4), \mathtt{B}(x_5), \mathtt{C}(x_6), \mathtt{D}(x_7) \rangle), \\
\ (0, \rho_1, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{X}(x_0, x_1), \mathtt{Y}(x_2, x_3), \mathtt{A}(x_4), \mathtt{B}(x_5), \mathtt{C}(x_6), \mathtt{D}(x_7) \rangle), \\
\ (1, \rho_2, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(x_0), \mathtt{B}(x_1), \mathtt{Y}(x_2, x_3), \mathtt{A}(x_4), \mathtt{B}(x_5), \mathtt{C}(x_6), \mathtt{D}(x_7) \rangle), \\
\ (3, \rho_5, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(x_0), \mathtt{B}(x_1), \mathtt{C}(x_2), \mathtt{D}(x_3), \mathtt{A}(x_4), \mathtt{B}(x_5), \mathtt{C}(x_6), \mathtt{D}(x_7) \rangle) \rangle
\end{array}$$

Wir stellen es dem Leser als Übung anheim zu zeigen, daß Γ' bis auf Umbenennung der Variablen eindeutig aus Γ hervorgeht. Wir haben dann an erster Stelle

$X_0(p_0, p_1, \ldots, p_{k-1})$, wo p_i Polynome in den Variablen sind, und in $q := \prod_{i<k} p_i$ jede Variable höchstens einmal auftritt. Wir nennen q (welches bis auf Umbenennung eindeutig ist) das **charakteristische Polynom** von Γ und den Beweis Γ' das **Skelett**. Es heiße ein Polynom **linear**, wenn es eine Variable höchstens enthält. Dann sind alle in Γ' vorkommenden Polynome linear. Ferner, ist $X_i(\langle q^i_j : j < k\rangle)$ in Γ', so ist auch das Produkt $\prod_{j<k} q^i_j$ linear. Wir schreiben $[p]$ für die Menge aller x_i, welche in p vorkommen. Das charakteristische Polynom für unsere Ableitung ist

$$p(x_0, x_1, \ldots, x_7) := x_4x_0x_1x_5x_6x_2x_3x_7$$

Nun sei Γ wie eben eine Ableitung von $X_0(\vec{x}_i : i < k)$. Wir bekommen eine bijektive Zuordnung von Vorkommen von Buchstaben in $\vec{u} := \prod_{i<k} \vec{x}_i$ zu Variablen im charakteristischen Polynom p. Der Einfachheit halber entspreche die Variable x_i gerade dem i-ten Buchstaben. Dann entsteht $\vec{u}$ aus p durch Einsetzen des i-ten Zeichens für die Variable x_i. Unser Polynom ist also nicht das oben angegebene, sondern jetzt

$$p(x_4, x_0, x_1, x_5, x_6, x_2, x_3, x_7) = x_0x_1x_2x_3x_4x_5x_6x_7$$

Jetzt sei $M := \{\gamma_i : i < |\Gamma|\}$, $P := \{x_i : i < |\vec{u}|\}$. Es heißt $S \subseteq P$ **Konstituente** von und S' ein **Segment** von S, wenn im Skelett Γ' für ein $i < p$ und das Hauptliteral $X_i(\langle q^i_j : j < k\rangle)$ von L_i gilt: $[q^i_j] = S'$ und $S = \bigcup_{j<k}[q^j_i]$. Wir schreiben auch $C(i)$ für S. Wir setzen $x_i \sqsubset x_j$ genau dann, wenn $i < j$ ist, $x_i < \gamma_j$ genau dann, wenn $x_i \in C(j)$, $\gamma_i < \gamma_j$, falls $j < i$ (!) und $C(i) \subseteq C(j)$ ist. Die zweite Bedingung legt fest, daß Teilkonstituenten aus Teilzeichenketten bestehen müssen. Ferner sei $\gamma_i \sqsubset \gamma_j$, falls für alle $x_{i'} < \gamma_i$ und alle $x_{j'} < \gamma_j$ gilt $x_{i'} \sqsubset x_{j'}$ (also $i' < j'$). Man beachte im Übrigen daß für jedes $i < |\vec{u}|$, $\{x_i\}$ eine Konstituente ist. Schließlich sei $\ell(\gamma_i) := X_i$ und $\ell(x_i) := u_i$, wo u_i das i-te Zeichen in $\vec{u}$ ist. Setze $\mathfrak{B}(\Gamma) = \langle M \cup P, <, \sqsubset, \ell\rangle$. Dies ist ein Baum.

Theorem 4.4.4 *$\bar{i}ndexStrukturbaum\mathfrak{B}(\Gamma)$ is ein geordneter Baum, der sogenannte **Strukturbaum** von Γ.*

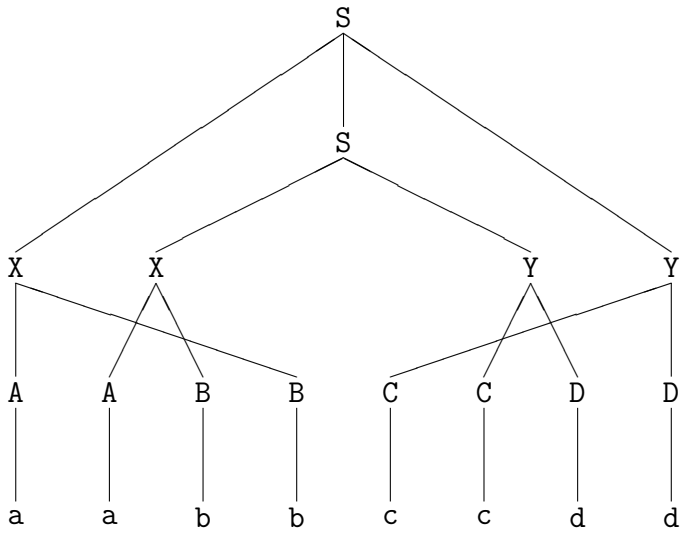
Dieser Baum sagt etwas über die Konstituentenstruktur aus, welche auf der Zeichenkette $\vec{u}$ in der Ableitung Γ liegt. Wir verdeutlichen dies an unserem Beispiel. Die erzeugte Zeichenkette ist `aabbccdd`. Wir schreiben Γ' noch einmal hin, diesmal mit den neuen Variablennamen.

$$\begin{aligned}
&\langle(0, \rho_0, \langle \mathtt{S}(x_0x_1x_2x_3, x_4x_5x_6x_7)\rangle),\\
&\ (0, \rho_0, \langle \mathtt{S}(x_1x_2, x_5x_6), \mathtt{X}(x_0, x_3), \mathtt{Y}(x_4, x_7)\rangle),\\
&\ (1, \rho_2, \langle \mathtt{S}(x_1x_2, x_5x_6), \mathtt{A}(x_0), \mathtt{B}(x_3), \mathtt{Y}(x_4, x_7)\rangle),\\
&\ (3, \rho_5, \langle \mathtt{S}(x_1x_2, x_5x_6), \mathtt{A}(x_0), \mathtt{B}(x_3), \mathtt{C}(x_4), \mathtt{D}(x_7)\rangle),\\
&\ (0, \rho_1, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{X}(x_1, x_2), \mathtt{Y}(x_5, x_6), \mathtt{A}(x_0), \mathtt{B}(x_3), \mathtt{C}(x_4), \mathtt{D}(x_7)\rangle),\\
&\ (1, \rho_2, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(x_1), \mathtt{B}(x_2), \mathtt{Y}(x_5, x_6), \mathtt{A}(x_0), \mathtt{B}(x_3), \mathtt{C}(x_4), \mathtt{D}(x_7)\rangle),\\
&\ (3, \rho_5, \langle \mathtt{S}(\varepsilon, \varepsilon), \mathtt{A}(x_1), \mathtt{B}(x_2), \mathtt{C}(x_5), \mathtt{D}(x_6), \mathtt{A}(x_0), \mathtt{B}(x_3), \mathtt{C}(x_4), \mathtt{D}(x_7)\rangle)\rangle
\end{aligned}$$

Wir haben die folgenden Hauptliterale und die davon bestimmten Konstituenten.

Zeile	Hauptliteral	Konstituente
6	$\mathtt{Y}(x_5, x_6)$	$\{x_5, x_6\}$
5	$\mathtt{X}(x_1, x_2)$	$\{x_1, x_2\}$
4	$\mathtt{S}(x_1x_2, x_5x_6)$	$\{x_1, x_2, x_5, x_6\}$
3	$\mathtt{Y}(x_4, x_7)$	$\{x_4, x_7\}$
2	$\mathtt{X}(x_0, x_3)$	$\{x_0, x_3\}$
1	$\mathtt{S}(x_0x_1x_2x_3, x_4, x_5, x_6, x_7)$	$\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$

Der Strukturbaum ist, wie man jetzt leicht nachrechnet, wie folgt.



Dieser Baum ist nicht erschöpfend geordnet. Dies ist der ganze Unterschied zu kontextfreiem Grammatiken. Man beachte, daß der Baum nicht die Position der leeren Konstituente wiedergibt. Ihre Segmente befinden sich zwischen dem zweiten und dem dritten sowie zwischen dem sechsten und dem siebten Buchstaben. Man kann den Strukturbaum allerdings auch so definieren, daß er explizit die leeren Zeichenketten vermerkt. Dazu muß man in Γ auch sämtliche Vorkommen von ε durch eine Variable ersetzen. Der Rest ist dann anolog wie eben.

Die Strukturen, die hier entstehen, kann man auch über eine kontextfreie Graphgrammatik erzeugen. Wir wollen dies jetzt vormachen. Zur Vereinfachung (welche im Übrigen nicht notwendig ist) wollen wir annehmen, daß alle Regeln *monoton* sind. Darbei heißt eine Regel monoton, falls sie die Ordnung der Segmente ihrer Literale nicht verändert.

Definition 4.4.5 *Es sei $\rho = L \leftarrow M_0 \dots M_{n-1}$ eine lineare Regel und $L = B(\langle t^j : j < k\rangle)$ sowie $M_i = A_i(\langle x_i^j : j < k_i\rangle)$. ρ heißt **monoton**, falls für jedes $i < n$ und jedes Paar $j < j' < k_i$ gilt: ist x_j im q-ten Segment und $x_{j'}$ im q'-ten Segment von L, so ist entweder $q < q'$, oder x_j tritt im Polynom t_q vor $x_{j'}$ auf.*

Folgende Regel ist monoton:

$$A(x_0y_0x_1, y_1) \leftarrow B(x_0, x_1) \quad C(y_0, y_1)$$

Denn es ist x_0 links von x_1 und y_0 im Segment vor dem Segment von y_1. Dagegen sind folgende Regeln nicht monoton:

$$A(x_1, x_0y_0y_1) \leftarrow B(x_0, x_1) \quad C(y_0, y_1); A(x_1x_0, y_0y_1) \leftarrow B(x_0, x_1) \quad C(y_0, y_1)$$

Wir wollen uns erst auf monotone Grammatiken beschränken. Wir zeigen anschließend, wie man diese Beschränkung wieder aufhebt. Es sei k gegeben. Der Einfachheit halber seien alle Prädikate k–stellig und terminale Regeln von der Form $Y(a, \varepsilon, \dots, \varepsilon)$, $a \in A$. Wir nennen eine $k \times k$–Matrix $M = (m_{ij})_{ij}$ mit Einträgen aus $\{0, 1\}$ ein k–**Schema**, falls für alle j, i und i' gilt: ist $i > i'$, so ist $m_{ij} \leq m_{i'j}$. Dieses Schema interpretieren wir so: sind $\vec{x}_i$, $i < k$, und $\vec{y}_j$, $j < k$, Folgen von Teilworten von $\vec{u}$ $\vec{u}$ wiederum ein Teilwort, so definieren wir ein Schema der $\vec{x}_i$ in Bezug auf die $\vec{y}_i$ durch $m_{ij} = 1$, falls $\vec{x}_i$ links von $\vec{y}_j$ liegt. M ist das zu diesen zwei Folgen gehörende Schema. Im Folgenden sind die $\langle \vec{x}_i : i < k\rangle$ sowie $\langle \vec{y}_j : j < k\rangle$ Konstituenten. Wir werden deswegen stets haben, daß die $\vec{x}_i$ und die $\vec{y}_j$ paarweise disjunkt sind; und ferner, daß $\vec{x}_i$ links von $\vec{x}_j$ liegt sowie $\vec{y}_i$ links von $\vec{y}_j$, falls $i > j$ (das ist eine Folge der Monotonieeigenschaft). Falls dies gegeben ist, so liegt durch Angabe eines Schemas eine und nur eine lineare Ordnung zwischen den $2k$ Segmenten fest. Wir führen dies an einem Beispiel vor. Hier sind also alle zulässigen 2–Schemata zusammen mit den Ordnungen, welche sie definieren. (Wir lassen die Vektorpfeile weg; die Ordnung ist durch 'links von in der Zeichenkette' bestimmt.)

$$\underset{x_0x_1y_0y_1}{\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}} \quad \underset{x_0y_0x_1y_1}{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}} \quad \underset{x_0y_1y_1x_1}{\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}}$$

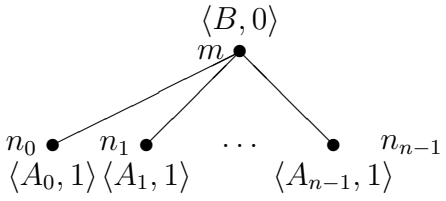
$$\underset{y_0x_0x_1y_1}{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}} \quad \underset{y_0x_0y_1x_1}{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}} \quad \underset{y_0y_1x_0x_1}{\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}}$$

Für jedes k–Schema M sei $\lambda(M)$ eine Relation. Jetzt definieren wir unsere Graph–Grammatik. Die Menge der Eckenfarben ist die Menge $F_E := N \times \{0, 1\} \cup A$, die Menge terminalen Eckenfarben ist $F_E^T := N \times \{0\} \cup A$, die Menge der Kantenfarben ist $\{<\} \cup \{\lambda(M) : M \text{ ein } k\text{–Schema}\}$. (Wie sich gleich herausstellen wird, ist $\sqsubset$ die Relation, die zu der Farbe $\lambda(\mathbb{I})$ paßt, wo $\mathbb{I} = (1)_{ij}$ die Matrix ist, die nur aus Einsen

besteht.) Der Startgraph ist der einelementige Graph $\mathfrak{S}$, welcher keine (!) Kanten besitzt, und dessen Knoten die Farbe S hat. Für jede Regel ρ fügen wir jetzt eine Graphersetzungsregel hinzu. Sei

$$\rho = B(\langle t^j : j < k\rangle) \leftarrow A_0(\langle x_0^j : j < k\rangle) \dots A_{n-1}(\langle x_{n-1}^j : j < k\rangle)$$

gegeben. Setze $p := \prod_{i<n} t^i$, das charakteristische Polynom der Regel. Dann wird die Graphersetzungsregel ρ^γ den Knoten $\langle B, 1\rangle$ durch den folgenden Graphen ersetzen:



Ferner existieren (im Bild nicht zu sehen) zwischen dem Knoten n_i und dem Knoten n_j die folgende Relation: setze $m_{i'j'} := 1$, falls $x_i^{i'}$ in p links von $x_j^{j'}$ steht. Dann setze $H_{ij} := (m_{i'j'})_{i'j'}$. Die Relation zwischen n_i und n_j ist also gleich H_{ij}. (Man bemerke hier gleich, daß nach Definition immer entweder $x_i^{i'}$ links von $x_j^{j'}$ oder rechts davon ist. Also ist die Relation zwischen n_j und n_i gerade $1 - H_{ij}$.) Dies definiert den Graphen. Jetzt muß man noch das Farbfunktional festlegen. Wie im Fall kontextfreier Grammatiken wechseln die Relationen nicht die Richtung, und so kommen wir mit den einfachen Funktionalen $\mathfrak{I}_\rho$ und $\mathfrak{O}_\rho$ aus. Es ist, da die Baumstruktur ganz analog dem kontextfreien Fall ist,

$$\mathfrak{I}_\rho(u, <) = \{<\}, \qquad \mathfrak{O}_\rho(u, <) = \{<\}\ .$$

Nun kommen wir zu den Relationen $\lambda(M)$. Sei M ein beliebiges k–Schema. Dann sei

$$M^p := (m_{ij}^p)_{ij}, \qquad m_{ij}^p = 1 \text{ gdw. ex. } j' : x_p^i \in [t^{j'}] \text{ und } m_{j'j} = 1$$

Dann setzen wir

$$\mathfrak{O}_\rho(n_p, \lambda(M)) := \{\lambda(M^p)\}; \qquad \mathfrak{O}_\rho(m, \lambda(M)) = \begin{cases} \{\lambda(\mathbb{I})\} & \text{falls } M = \mathbb{I}, \\ \varnothing & \text{sonst.} \end{cases}$$

Genauso setzen wir

$$\mathfrak{I}_\rho(n_p, \lambda(M)) := \{\lambda(M^p)\}; \qquad \mathfrak{I}_\rho(m, \lambda(M)) = \varnothing$$

Wir liefern gleich die anschauliche Erklärung mit. Die Tatsache, daß der zu ersetzende Knoten, sagen wir u, eine auslaufende Kante mit Namen $\lambda(M)$ zu einem Knoten v hat, besagt, daß das Schema M die Relationen der Segmenten von u und den

Segmenten von v in der oben beschriebenen Weise kodiert. Nun ist der Knoten n_p eine Teilkonstituente von u. Wir müssen nun sagen, ob das Segment von n_p der Nummer i links von dem Segment von v mit der Nummer j liegt. Dazu schauen wir, für welches j' die Variable $x_i^{j'}$ in $t^{j'}$ auftaucht. Denn $t^{j'}$ definiert das j'-te Segment von u. Genau dann ist unser Segment links vom j-ten Segment von u, wenn das j'-te Segment von u links von v liegt, das heißt, wenn $m_{j'j} = 1$.

Für die terminalen Regeln schließlich nehmen wir einen einfach verzweigenden Baum, wo also $p = 0$. Ferner setzen wir

$$\mathfrak{O}_\rho(n_0, M) = \begin{cases} \{\lambda(\mathbb{I})\} & \text{falls für alle } j\text{: } m_{0j} = 1, \\ \varnothing & \text{falls für alle } j\text{: } m_{0j} = 0, \\ \{\lambda(M^0)\} & \text{sonst.} \end{cases}$$

(Analog für $\mathfrak{I}_\rho$.) Diese Regel ist wiederum so gemacht, daß sie Rücksicht darauf nimmt, daß die Segmente außer dem ersten leer sind. Man beachte, daß diese Grammatik Strukturen der Form $\langle B, <, \sqsubset, \ell\rangle$ erzeugt, also geordnete Bäume, wie man leicht nachrechnet. Man beachte nämlich, daß die Relationen zwischen Knoten herausgeworfen werden, sofern sie verschieden sind von $\lambda(\mathbb{I})$ (welches gerade die $\sqsubset$–Relation ist). Die so entstehende Grammatik nennen wir γG.

Was macht man nun, wenn die Grammatik nicht monoton ist? Hier gibt es eigentlich nichts entscheidend Neues, außer daß einstellige Regeln, welche das Startsymbol entwickeln, Schwierigkeiten machen. Denn angenommen, wir haben die Anfangsregel

$$\mathtt{S}(x_1, x_0) \leftarrow \mathtt{T}(x_0, x_1)$$

so wüßten wir nicht, ob x_0 nun tatsächlich stets links von x_1 liegt oder nicht. Dies kann man erst dann wissen, wenn ein anderes Element zwischen x_0 und x_1 tritt, denn dann liegt x_0 entweder links oder rechts von diesem Element und dann ist x_1 rechts bwz. davon. Um dieses Problem zu vermeiden, kann man zu folgender Lösung greifen. Wir schreiben anstelle der Symbole $\langle B, i\rangle$, $B \in N$, $i \in \{0, 1\}$ nun Symbol der Form $\langle B, \pi\rangle$ sowie B hin, wobei π eine Permutation der Menge $\{0, 1, \ldots, k-1\}$ ist. Die Permutation π verrät uns, wie die Segmente der Konstituente im Teilwort *tatsächlich*, das heißt unabhängig von ihrem Auftreten in der Folge, geordnet sind. Man kann die Permutation nach erfolgter Ersetzung getrost wieder vergessen. Deswegen benutzen wir als Menge der Nichtterminalsymbole jetzt die Menge

$$N' := N \cup \{\langle A, \pi\rangle : A \in N, \pi \text{ Permutation von } \{0, 1, \ldots, k-1\}\} .$$

Wir verzichten auf eine genaue Spezifikation der Regeln. Es sollte klar sein, wie man verfahren muß. Wir dürfen also im Folgenden stets von Strukturbäumen reden sowie von einer direkten Erzeugung solcher Strukturbäume; denn nun existiert eine bijektive Zuordnung zwischen Ableitungen in G und Ableitungen in γG, und in jedem Schritt sind die assoziierten Strukturen in einem gewissen Sinne isomorph.

Man kann diese Struktur auch ausnutzen, um ganz analog zu den kontextfreien Sprachen ein Pumplemma zu formulieren. Wir gehen hier nicht mehr so gründlich vor wir in dem Fall der kontextfreien Sprachen und überlassen dem Leser einige Einzelheiten. Es sei zunächst vermerkt, daß eine Konstituente $\mathfrak{k}$ aus bis zu k Teilworten $\vec{v}_i$ (die wir Segmente genannt haben) besteht. Ist $\vec{x}$ ein Wort, so zerlegt eine Konstituente $\mathfrak{k}$ das Wort $\vec{x}$ in bis zu $2k+1$ Teilworte, nämlich wie folgt:

$$\vec{x} = \vec{u}_0\vec{v}_0\vec{u}_1\vec{v}_1 \dots \vec{v}_{k-1}\vec{u}_k$$

Es sei nun $\mathfrak{k}$ eine Konstituente vom Typ A, und sie enthalte echt eine weitere Konstituente $\mathfrak{m}$ vom Typ A. Dann zerlegen die Segmente von $\mathfrak{m}$, nennen wir sie $\vec{w}_i$, die Segmente von $\mathfrak{k}$ in bis zu $2k$ Stücke. Diese bilden die Differenz zwischen diesen beiden Konstituenten. Diese $2k$ Segmente sind es nun, welche pumpbar sind in dem folgenden Sinn.

Definition 4.4.6 *Es sei $S \subseteq A^*$ eine Sprache. S heißt n–**pumpbar**, falls es eine Zahl k_S gibt derart, daß jede Zeichenkette $\vec{x} \in S$ der Länge mindestens k_S wie folgt zerlegbar ist:*

$$\vec{x} = (\prod_{i<n} \vec{u}_i\vec{v}_i) \cdot \vec{u}_n$$

und es ist

$$\{(\prod_{i<n} \vec{u}_i\vec{v}_i^q) \cdot \vec{u}_n : q \in \omega\} \subseteq S$$

Dabei gilt

1. $\prod_{i<n} \vec{v}_i \neq \varepsilon$,
2. $\prod_{0<i<n} \vec{u}_i < k_S$.

Theorem 4.4.7 (Groenink) *Es sei G eine lineare k–LBG. Dann ist $L(G)$ $2k$–pumpbar.*

Analog zu den kontextfreien Sprache kann man diesen Satz verschärfen. Mann kann zum Beispiel verlangen, daß eine gegebenes Teilwort der Länge mindestens k_S eines der vier Worte enthalten muß (welches dazu noch nicht leer sein darf).

Es gibt im Übrigen einen sehr interessanten Beweis dieser Tatsache, den man für andere Zwecke gut einsetzen kann. Man nenne eine Regel **zyklisch**, falls sie folgende Form hat

$$A(\vec{u}_0x_0\vec{v}_1, \dots, \vec{u}_{n-1}x_{n-1}\vec{v}_{n-1}) \leftarrow A(x_0, \dots, x_{n-1})$$

mit $\vec{u}_i, \vec{v}_i \in A^*$ für alle $i < n$. Ist eine Zeichenkette nur hinreichend lang, so findet man eine A–Konstituente mit Segmenten $\vec{u}_i\vec{w}_i\vec{v}_i$, $i < n$, derart, daß auch die $\vec{w}_i$ eine

A–Konstituente bilden. In diesem Fall haben wir eine zyklische Regel abgeleitet. Wir können diese nun selbstverständlich beliebig oft wiederholen, und so erhalten wir die gewünschte Pumparkeit der $2k$ Segmente. Man hat hieraus den Beweis, daß $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n : n \in \omega\}$ nicht durch eine lineare 2–LBG erzeugbar ist. Denn diese Sprache ist nicht 4–pumpbar.

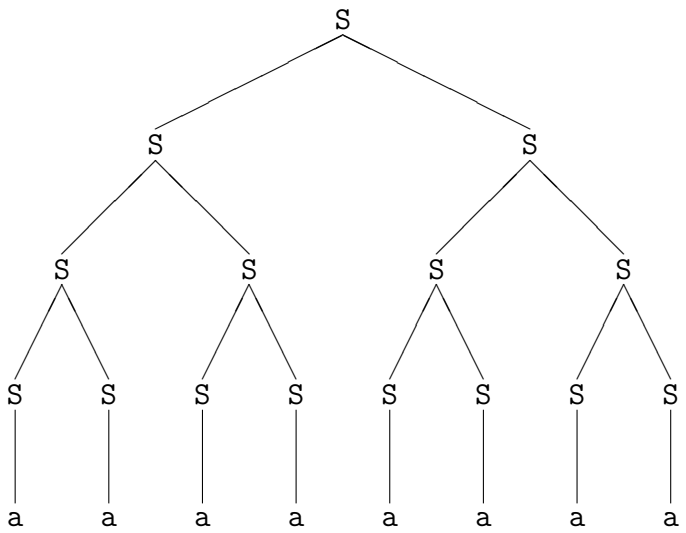
Zu guter Letzt wollen wir noch auf einfache Grammatiken eingehen, welche nicht linear sind. Auch hier ist in beschränkten Fällen ein Strukturbegriff möglich, nämlich immer dann, wenn die rechten Seiten von Regeln keine Variable zweimal enthalten. Dies unterscheiden sich also von linearen Grammatiken dadurch, daß Variablen auf der *linken* Seite mehrfach auftreten dürfen. Ein Beispiel ist die Grammatik

$$\mathtt{S}(xx) \leftarrow \mathtt{S}(x), \qquad \mathtt{S}(\mathtt{a}).$$

In diesem Fall kann man den Strukturbegriff, der eben entwickelt wurde, auf diese Grammatik übertragen. Wir tun ganz einfach so, als sei die erste Regel von dieser Form

$$\mathtt{S}(xy) \leftarrow \mathtt{S}(x) \quad \mathtt{S}(y)$$

wobei klar ist, daß x und y stets die gleiche Zeichenkette repräsentieren. Auf diese Weise erhält man folgenden Strukturbaum für aaaaaaaa.



Übung 98. Zeigen Sie, daß die Ableitung Γ' durch Γ bis auf Umbenennung der Variablen eindeutig bestimmt wird.

Übung 99. Beweisen Sie Proposition 4.4.2.

Übung 100. Bestimmen Sie die Graphgrammatik $\gamma G^{\heartsuit}$.

Übung 101. Man zeige: Es sei $N = \{x_i : i < k\} \cup \{y_i : i < k\}$ und $<$ eine lineare Ordnung auf N mit $x_i < x_j$ sowie $y_i < y_j$ für alle $i < j < k$. Dann gilt: ist $m_{ij} = 1$ genau dann, wenn $x_i < y_j$, so ist $M = (m_{ij})_{ij}$ ein k–Schema. Umgekehrt: sei M ein k–Schema und $<$ definiert durch (1) $x_i < x_j$ gdw. $i < j$, (2) $y_i < y_j$ gdw. $i < j$, (3) $x_i < y_j$ gdw. $m_{ij} = 1$. Dann ist $<$ eine lineare Ordnung. Die Beziehung zwischen Ordnungen und Schemata ist sogar eineindeutig.

Übung 102. Man zeige: sind in einer k–LBG alle Strukturbäume erschöpfend geordnet, so ist die erzeugte Baummenge kontextfrei.

4.5 Adjunktionsgrammatiken

In diesem und dem nächsten Abschnitt wollen wir uns einigen alternativen Grammatiktypen widmen, die in etwa mit den Kopfgrammatiken äquivalent sind. Diese sind die Baumadjunktionsgrammatiken, die Kombinatorischen Kategorialgrammatiken (welche jeweils eine verfeinerte Fassung der unregulierten Baumadjunktionsgrammatiken aus Kapitel 1.4 bzw. der in Kapitel 3.2 definierten Grammatiken $CCG(\mathbf{Q})$ sind) und die sogenannten linearen Indexgrammatiken.

Kommen wir also zu dem Konzept einer Baumadjunktionsgrammatik zurück. Diese ist eine Paar $G = \langle \mathsf{C}, \mathsf{A} \rangle$, wo C eine Menge von sogenannten Zentralbäumen und A eine Menge von sogenannten Adjunktionsbäumen ist. In einem Adjunktionsbaum heißt ein Knoten **zentral**, falls er oberhalb des ausgezeichneten Blattes liegt oder mit diesem identisch ist. Wir definieren eine **Ableitung** wie folgt. Eine Ableitung ist ein Baum, der durch folgende kontextfreie Grammatik erzeugt wird. Die Nichtterminalsymbole sind Paare $(\mathfrak{B}, i)$, wo $\mathfrak{B}$ ein Baum ist und i ein Knoten in $\mathfrak{B}$. Wir gehen der Einfachheit halber davon aus, daß die Knotenmenge von $\mathfrak{B}$ die Menge $j(\mathfrak{B}) = \{0, 1, \ldots, j(\mathfrak{B}) - 1\}$ ist. Terminalsymbole sind Symbole der Form i, wo i ein Knoten in einem Baum ist. Das Startsymbol ist S

$$\begin{array}{lllll} \text{(s)} & S & \to & (0, \mathfrak{C}) & \\ \text{(a)} & (j, \mathfrak{A}) & \to & X_0 & X_1 \ldots X_{j(\mathfrak{A})-1} \end{array}$$

wobei (a) als Regelschema aufzufassen ist: für jedes $\mathfrak{A}$ und jedes zulässige j hat man 2^k viele Regeln, wobei X_i stets entweder i ist oder $(i, \mathfrak{B}_i)$ für einen Baum $\mathfrak{B}_i$, welcher an i in $\mathfrak{A}$ adjungierbar ist. In (s) muß $\mathfrak{C} \in \mathsf{C}$ sein. Die Grammatik nennen wir $D(G)$ und nennen es die **Derivationsgrammatik**. Diese ist ein ziemlich nützliches Konzept.

Ist $\mathfrak{T}$ ein Baum, der durch G abgeleitet wird, so existiert ein assoziierter Derivationsbaum $D(\mathfrak{T})$, der wie folgt definiert ist. (a) Ist kein Baum an $\mathfrak{T}$ adjungiert

worden, so ist $\mathfrak{T}$ ein Zentralbaum. Der assoziierte Baum wird bestimmt durch die Ableitung

$$S \to (0, \mathfrak{T}) \to 0 \quad 1 \ \ldots \ j(\mathfrak{T}) - 1$$

Jetzt sei $\mathfrak{A}$ ein einfacher Adjunktionsbaum. Diesem ordnen wir den Baum zu folgender Ableitung zu.

$$D^j(\mathfrak{T}) := (j, \mathfrak{T}) \to 0 \quad 1 \ \ldots \ j(\mathfrak{T}) - 1$$

Hierbei ist j beliebig. Es entstehe $\mathfrak{T}$ durch Adjunktion der Bäume $\mathfrak{U}_{k_j}$ an die Knoten k_j, $j < p$, des Baumes $\mathfrak{A}$. Dann ordnen wir $\mathfrak{T}$ den folgenden Baum zu:

$$(j, \mathfrak{A}) \to X_0 \quad X_1 \ \ldots \ X_{j(\mathfrak{A})-1}$$

wobei $X_i = i$ ist, falls i kein k_j ist, ansonsten sei $X_{k_j} = D^{k_j}(\mathfrak{U}_{k_j})$. (Man beachte also, daß wir Bäume definieren und keine Ableitungen. Sonst macht diese Definition keinen Sinn.) Ist umgekehrt $\mathfrak{D}$ ein Baum, der durch $D(G)$ erzeugt wird, so kann man induktiv einen Baum $A(\mathfrak{D})$ finden, sodaß $D(A(\mathfrak{D})) \cong \mathfrak{D}$. Dies tun wir weiter unten. Insofern existiert tatsächlich die versprochene Korespondenz.

Ist eine Ableitungsbaum $\mathfrak{D} = \langle D, <, \sqsubset, \ell \rangle$ gegeben, so sind die Terminalknoten von $A(\mathfrak{D})$ in bijektiver Korrespondenz mit den Blättern des abgeleiteten Baumes. Man betrachte ein Blatt x der Ableitung. Man definiere, analog zu dem Zweigausdruck (siehe Übung 2.2) eine **Adresse** wie folgt. In der Definition sei $y \succ x$.

$$\begin{array}{lcll} \alpha(x) & := & \mathfrak{C} & \text{falls } t(x) = 1, \ell(x) = (0, \mathfrak{C}) \\ \alpha(x) & := & \alpha(y) \cdot i & \text{falls } \ell(x) = i \\ \alpha(x) & := & \alpha(y) \cdot i \cdot GA & \text{falls } \ell(x) = (i, \mathfrak{A}) \end{array}$$

Für x ein Blatt gibt dies die Adjunktionsgeschichte des Knotens assoziierten Knoten wieder. Nun sei

$$A(\mathfrak{D}) := \langle A, <^A, \sqsubset^A, \ell^A \rangle$$

wobei wir haben

$$A := \{\alpha(x) : x \text{ Blatt von } \mathfrak{D}\}$$

Ferner, ist $\alpha(x) = I \cdot GA \cdot j$, so sei $\ell^A(x) = X$ für dasjenige X, welches die Marke des Knotens j in $\mathfrak{A}$ ist. Zweitens, $\alpha(x) \sqsubset \alpha(y)$, falls $\alpha(x) = \vec{i} \cdot \mathfrak{A} \cdot j \cdot \vec{j}$ ist und $\alpha(y) = \vec{i} \cdot \mathfrak{A} \cdot j'; \vec{j'}$ für gewisse I, J, $\mathfrak{A}$ und $j \neq j'$, derart, daß $j \sqsubset j'$. Drittens, $\alpha(x) < \alpha(y)$, falls $\alpha(x) = \vec{i} \cdot \mathfrak{A} \cdot j \cdot \vec{j}$ ist und $\alpha(y) = \vec{i} \cdot \mathfrak{A} \cdot j' \cdot \vec{j'}$ für gewisse $\vec{i}$, $\vec{j}$, $\vec{j'}$, $\mathfrak{A}$ und $j \neq j'$, derart, daß (a) $j < j'$, (b) j' und sämtliche Knoten von $\vec{j'}$ sind zentral in ihren jeweiligen Bäumen.

Adjunktionsgrammatiken (BAGn) unterscheiden sich von den unregulierten Baumadjunktionsgrammatiken dadurch, daß man noch spezifizieren darf,

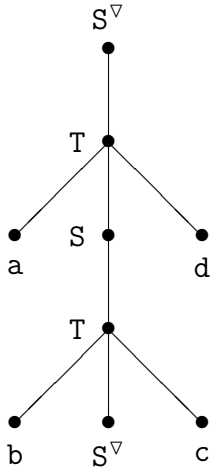
1. ob an gewissen Knoten adjungiert werden darf oder nicht,

2. welche Bäume an welche Knoten adjungiert werden dürfen,

3. ob an gewisse Knoten adjungiert werden muß.

Wir werden zeigen, daß nur die erste Änderung wirklich etwas neues bringt. Um die Kontrolle über die Ableitungen zu etablieren, müssen wir einige Änderungen in den Definitionen vornehmen. Wir beginnen mit dem ersten Punkt, dem Adjunktionsverbot. Um dies zu implementieren, nehmen wir an, daß Kategoriesymbole nunmehr von der Form a, $a \in A$, oder X bwz. $X^{\triangledown}$ sind, wo $X \in N$. Zentral- bzw. Adjunktionsbäume sind wie vorher definiert; es gibt also ein Blatt i in diesem Baum, welches die gleiche Marke trägt wie die Wurzel. Es gilt aber, daß an Knoten mit einer Marke der Form $X^{\triangledown}$ nicht adjungiert werden darf. Solche Knoten besitzen also, wie wir sagen wollen, ein *Adjunktionsverbot*. Wir erlauben nun (auch wenn dies sich als nicht notwendig erweist), daß $\mathfrak{T}$ auch dann ein Adjunktionsbaum ist, wenn ein Blatt die Marke X bzw. $X^{\triangledown}$ trägt und der Kopf die Marke $X^{\triangledown}$ bzw. X. Das ausgesonderte Blatt muß also nicht exakt die gleiche Marke tragen wie die Wurzel; sondern diese Marken dürfen sich in dem Adjunktionsvermerk unterscheiden. Mit solchen Grammatiken kann man zum Beispiel die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n : n \in \omega\}$ erzeugen. Man wähle dazu den folgenden Startbaum.

Der Adjunktionsbaum ist wie folgt.



Es ist nun nicht schwer zu zeigen, daß man sich unter Wahrung der Konstituentenstrukturen auf Adjunktionsgrammatiken beschränken kann, in denen jeweils nur die Marke der Wurzel und des ausgesonderten Blatts ein Adjunktionsverbot tragen. Dazu zwei Überlegungen. (1) Falls ein Knoten im Inneren eines Adjunktionsbaums ein Adjunktionsverbot trägt, so ersetzen wir die Marke des Knotens durch eine Marke $\notin N$. Auf diese Weise kann dann dort kein Baum mehr adjungiert werden. (2) Falls die Wurzel kein Adjunktionsverbot trägt, so füge einen neuen Wurzelknoten hinzu mit gleichem Nichtterminalsymbol, der nun ein Adjunktionsverbot erhält. Desgleichen mit dem ausgezeichneten Blatt.

Definition 4.5.1 *Eine* ***Standard Baumadjunktionsgrammatik*** *ist eine Adjunktionsgrammatik, in welcher die Adjunktionsbäume genau an der Wurzel und dem ausgezeichneten Blatt ein Adjunktionsverbot tragen.*

Kommen wir zu dem zweiten Punkt; man kann auch spezifizieren, ob an einem Knoten adjungiert werden muß. Wir haben also zusätzlich eine Funktion f, welche jedem Knoten eines Zentral- oder Adjunktionsbaums seine zulässige Menge von Adjunktionsbäumen zuordnet. (Ist $f(i) = \varnothing$, so hat der Knoten ein Adjunktionsverbot.) Wir können dies so simulieren. Es sei A die Menge der Adjunktionsbäume. Dann sind Nichtterminalsymbole jetzt von der Form $\langle X, \mathfrak{T}\rangle$ bzw. $\langle X, \mathfrak{T}\rangle^{\triangledown}$, wo $X \in N$ und $\mathfrak{T} \in \mathsf{A}$. Ein Baum $\mathfrak{T}$ wird ersetzt durch sämtliche Bäume $\mathfrak{T}'$ auf derselben Knotenmenge, bei welchen i die Marke $\langle X, \mathfrak{U}\rangle$ trägt, wenn i die Marke X in $\mathfrak{T}$ hat, falls $\mathfrak{U} \in f(i)$, bwz. die Marke $\langle X, \mathfrak{U}\rangle^{\triangledown}$, falls i die Marke $X^{\triangledown}$ in $\mathfrak{T}$ hat. Die zweite Komponente sagt also nichts weiter aus, als welcher Baum als nächstes adjungiert wird.

Als letztes betrachten wir den dritten Punkt, die Adjunktionspflicht. Dies können wir implementieren, indem wir nun auch Marken der Form $X^{\bullet}$ einführen. (Da sich Pflicht und Verbot ausschließen, tritt $\bullet$ nur auf, wenn $\triangledown$ nicht auftritt.) Ein Baum ist erst dann fertig, wenn keine Knoten mit Adjunktionspflicht das heißt mit einer Marke der Form $X^{\bullet}$ vorhanden sind. Wir zeigen nun, daß zu jeder Adjunktionsgrammatik mit Pflicht eine (die gleichen Bäume erzeugende) Adjunktionsgrammatik ohne Pflicht gibt. Zu jedem Zentralbaum adjungieren wir so oft, bis die Adjunktionspflicht erlischt. Desgleichen für Adjunktionsbäume. Als neue Zentral- bzw. Adjunktionsbäume betrachten wir solche, die minimal sind mit der Eigenschaft, daß an keinen Knoten adjungiert werden muß. Daß solche existieren ist klar; daß es nur endlich viele gibt, muß man allerdings begründen. Dazu betrachte man einen Baum ohne Adjunktionspflicht, und einen Knoten. Dieser hat eine Adjunktionsgeschichte. Er ist durch sukzessives Adjungieren von Bäumen an einen Zentral- oder Adjunktionsbaum entstanden. Falls diese Folge von Bäumen einen Baum doppelt enthält, können wir diese Schleife eliminieren. (Die Details überlassen wir dem Leser.) Diese Grammatik leitet alle Bäume wie die gegebene ab. Somit dürfen wir bei unserem vereinfachten Format bleiben.

Wir wollen nun als erstes den Nachweis erbringen, daß Adjunktionsgrammatiken nicht mehr Sprachen erzeugen als lineare 2–LBGn. Daraus ergibt sich sofort, daß das Parsingproblem in polynomieller Zeit lösbar ist.

Theorem 4.5.2 (Weir) *Zu jeder Adjunktionsgrammatik G existiert eine Kopfgrammatik K mit $L(K) = L(G)$.*

Beweis. Es sei G gegeben. Wir nehmen an, die Bäume haben paarweise disjunkte Mengen von Knoten. Wir können annehmen, daß die Bäume höchstens zweifach verzweigen. (Es kommt nur auf schwache Äquivalenz an.) Ferner können wir annehmen, daß Knoten strikt verzweigen, wenn sich nicht präterminal sind. Die Menge aller Knoten sei M. Das Alphabet der Nichtterminalsymbole sei $N' := \{i^a : i \in M\} \cup \{i^n : i \in M\}$. Als Startsymbole haben wir die Menge aller i^a bzw. i^n, wo i die Wurzel eines Zentralbaums ist. Man kann durch leichte Massage eine Grammatik mit nur einem Startsymbol basteln. Nun definieren wir die Regeln. Für einen lokalen Baum $i \to j$ setzen wir die Regel

$$(t) \qquad i(a, \varepsilon) \leftarrow . \,,$$

wo j Blatt ist mit Terminalzeichen a. Ist i ein ausgezeichnetes Blatt eines Adjunktionsbaumes, so nehmen wir auch noch die Regel

$$(e) \qquad i^n(\varepsilon, \varepsilon) \leftarrow .$$

auf. Jetzt sei $i \to j \quad k$ ein verzweigender lokaler Baum. Dann nehmen wir die folgende Regel hinzu:

$$(p) \qquad i^a(x_0x_1, y_0y_1) \leftarrow j^n(x_0, x_1) \quad k^n(y_0, y_1)$$

Ferner haben wir noch folgende Regeln: ist i ein Knoten, an den ein Baum mit Wurzel j adjungierbar ist, so sei auch dies eine Regel

$$(f) \qquad i^n(x_0y_0, y_1x_1) \leftarrow j^n(x_0, x_1) \quad i^a(y_0, y_1) \,.$$

Ist Adjunktion nicht notwendig (oder verboten) bei i, so fügen wir schließlich noch die folgende Regel hinzu.

$$(n) \qquad i^n(x_0, x_1) \leftarrow i^a(x_0, x_1) \,.$$

Dies beendet die Definition von K. Es ist angesichts der Regeln (p) noch nicht klar, daß wir es mit einer Kopfgrammatik zu tun haben. Man ersetze daher die Regeln (p) jeweils durch folgende Regeln:

$$\begin{array}{lcl} i^a(x_0, x_1y_0y_1) & \leftarrow & j^{n\bullet}(x_0, x_1) \quad k^{n\bullet}(y_0, y_1) \\ j^{n\bullet}(x_0x_1y_0, y_1) & \leftarrow & j^n(x_0, x_1) \quad \mathtt{L}(y_0, y_1) \\ k^{n\bullet}(x_0, x_1y_0y_1) & \leftarrow & \mathtt{L}(x_0, x_1) \quad k^n(y_0, y_1) \\ \mathtt{L}(\varepsilon, \varepsilon) & \leftarrow & . \end{array}$$

Diese sind Regeln einer Kopfgrammatik; (p) läßt sich aus ihnen ableiten. Aus diesem Grund bleiben wir jetzt bei den Regeln (p).

Es bleibt nun zu zeigen, daß $L(K) = L(G)$. Zunächst die Inklusion $L(G) \subseteq L(K)$. Wir zeigen dabei folgendes. Sei $\mathfrak{T}$ ein lokaler Teilbaum, welcher genau ein ausgezeichnetes Blatt enthält, und Nichtterminalknoten x_i, $i < n$, mit Marken k_i hat. Sei also $j < i$ ausgezeichnet. Wir ordnen $\mathfrak{T}$ ein Vektorpolynom $\mathfrak{p}(\mathfrak{T})$ zu, welches zu gegebenen Paaren von Zeichenketten $\langle \vec{y}_i, \vec{y}'_i \rangle$ das Paar

$$\langle \prod_{i<j} \vec{y}_i \vec{z}_i, \prod_{j<i<n} \vec{y}_i \vec{z}_i \rangle$$

auswirft. Durch Induktion über $\mathfrak{T}$ läßt sich zeigen, daß es eine K–Ableitung

$$i^n(\mathfrak{p}(\mathfrak{T})(\langle \langle \vec{y}_i, \vec{z}_i \rangle : i < n \rangle)) \leftarrow^* k_0^n(\langle \vec{y}_0, \vec{z}_0 \rangle) \quad \dots k_{n-1}^n(\langle \vec{y}_{n-1}, \vec{z}_{n-1} \rangle)$$

gibt. Ist in $\mathfrak{T}$ kein Blatt ausgezeichnet, so sei der Wert von $p(\mathfrak{T})$ genau

$$\langle \vec{y}_0 \vec{z}_0, \prod_{0<i<n} \vec{y}_i \vec{z}_i \rangle$$

Diese Behauptung läßt sich induktiv über die Ableitung von $\mathfrak{T}$ in G beweisen. Daraus folgt dann aber sofort, daß $\vec{x} \in L(K)$, falls $\vec{x} \in L(G)$. Für umgekehrte Inklusion muß man einen anderen Beweis wählen. Es sei $\vec{x} \in L(K)$. Wir wählen eine K–Ableitung von $\vec{x}$. Angenommen, keine Regel vom Typ (f) wurde verwendet. Dann ist $\vec{x}$ die Zeichenkette eines Zentralbaumes, wie man leicht sieht. Nun nehmen wir an, die Behauptung sei gezeigt für Ableitungen mit weniger als n Anwendungen von (f), und es habe der Beweis genau n Anwendungen. Wir schauen die letzte Anwendung an. Auf diese folgen nur noch Anwendungen von (p), (t) und (e). Diese kommutieren, wenn sie zu verschiedenen Teilbäumen gehören. Man kann sie also so umordnen, daß auf unsere Anwendung von (f) gerade die Anwendungen von (p), (t) und (e) folgen, die zu diesem Teilbaum gehören. Diese leiten ab

$$i^a(\vec{x}_0, \vec{x}_1)$$

wo i die linke Seite der Anwendung von (f) ist, und $\langle \vec{x}_0, \vec{x}_1 \rangle$ das Paar des Adjunktionsbaumes, dessen Wurzel i ist. ($\vec{x}_0$ liegt links von dem ausgezeichneten Blatt, $\vec{x}_1$ rechts.) Davor haben wir die Anwendung unserer Regel (f):

$$j^a(\vec{x}_0 \vec{y}_0, \vec{y}_1 \vec{x}_1) \leftarrow i^a(\vec{x}_0, \vec{x}_1) \quad j^n(\vec{y}_0, \vec{y}_1)$$

Jetzt streichen wir diesen Teil der Ableitung. Dies bedeutet, daß wir anstelle von $j^a(\vec{x}_0 \vec{y}_0, \vec{y}_1 \vec{x}_1)$ nur noch $j^n(\vec{y}_0, \vec{y}_1)$ haben. Dies ist aber herleitbar (die Ableitung existiert schon). Aber auf der Adjunktionsseite entspricht dies genau dem Aushängen des dazugehörigen Adjunktionsbaumes. $\dashv$

Es stellt sich nun die Frage, ob auch die Umkehrung gilt. Dies ist nicht der Fall. Als Beispiel betrachten wir folgende Grammatik G.

$$\begin{array}{lcll}
\mathtt{S}(y_0x_0y_1, x_1) & \leftarrow & \mathtt{T}(x_0, x_1) & \mathtt{H}(y_0, y_1) \\
\mathtt{T}(x_0, \mathtt{c}x_1\mathtt{d}) & \leftarrow & \mathtt{U}(x_0, y_1) & \\
\mathtt{U}(\mathtt{a}x_0\mathtt{b}, x_1) & \leftarrow & \mathtt{S}(x_0, x_1) & \\
\mathtt{S}(\mathtt{ab}, \mathtt{cd}) & \leftarrow & . & \\
\mathtt{H}(\mathtt{t}x_0\mathtt{u}, x_1) & \leftarrow & \mathtt{K}(x_0, x_1) & \\
\mathtt{K}(x_0, \mathtt{v}x_1\mathtt{w}) & \leftarrow & \mathtt{H}(x_0, x_1) & \\
\mathtt{H}(\varepsilon, \varepsilon) & \leftarrow & . &
\end{array}$$

Man sieht zunächst einmal, daß dies sogar eine Kopfgrammatik ist. Um die erzeugte Sprache zu untersuchen, halten wir folgende Fakten fest.

Lemma 4.5.3 *Genau dann gilt* $\mathtt{H}(\vec{x}, \vec{y})$*, wenn* $\langle \vec{x}, \vec{y} \rangle = \langle \mathtt{t}^n\mathtt{u}^n, \mathtt{v}^n\mathtt{w}^n \rangle$ *für ein gewisses* $n \in \omega$*.*

Zum Beweis überlege man sich, daß erstens $\vdash_G \mathtt{H}(\varepsilon, \varepsilon)$ und zweitens gilt

$$\vdash_G \mathtt{H}(\mathtt{t}x_0\mathtt{u}, \mathtt{v}x_1\mathtt{w}) \text{ gdw. } \vdash_G \mathtt{H}(x_0, x_1)$$

Daraus kann man dann folgende Charakterisierung ableiten.

Lemma 4.5.4 *Es sei* $\vec{x}_n := \mathtt{t}^n\mathtt{u}^n$ *und* $\vec{y}_n := \mathtt{v}^n\mathtt{w}^n$*. Dann ist*

$$L(G) = \{\mathtt{a}\vec{x}_{n_0}\mathtt{a}\vec{x}_{n_1}\mathtt{a}\ldots\vec{x}_{n_{k-1}}\mathtt{ab}\vec{y}_{n_{k-1}}\mathtt{b}\ldots\mathtt{b}\vec{y}_{n_1}\mathtt{b}\vec{y}_{n_0}\mathtt{bc}^k\mathtt{d}^k : k \in \omega, n_i \in \omega\}$$

Insbesondere ist für jedes $\vec{x} \in L(G)$

$$\mu(\vec{x}) = m(\mathtt{a} + \mathtt{b} + \mathtt{c} + \mathtt{d}) + n(\mathtt{t} + \mathtt{u} + \mathtt{v} + \mathtt{w})$$

für gewisse natürliche Zahlen m *und* n*.*

Also zum Beispiel

$$\mathtt{aabbccdd},\ \mathtt{atuabvwbccdd},\ \mathtt{attuuatuabbvwbvvwwbcccddd},\ \ldots$$

aber nicht

$$\mathtt{atuabbcd},\ \mathtt{attuuatuabvwbvvwwbccdd}$$

Nun also zu dem angestrebten Beweis, daß es keine Adjunktionsgrmmatik gibt, die diese Sprache erzeugen kann. Sei das Gegenteil der Fall, und sei H eine Adjunktionsgrammatik mit $L(H) = L(G)$. Dann gilt

Lemma 4.5.5 *Es sei H eine Adjunktionsgrammatik mit $L(H) = L(G)$ und $\mathfrak{B}$ ein Zentral- oder Adjunktionsbaum. Dann ist*

$$\mu(\mathfrak{B}) = m_{\mathfrak{B}}(\mathtt{a} + \mathtt{b} + \mathtt{c} + \mathtt{d}) + n_{\mathfrak{B}}(\mathtt{t} + \mathtt{u} + \mathtt{v} + \mathtt{w})$$

für gewisse natürliche Zahlen $m_{\mathfrak{B}}$ und $n_{\mathfrak{B}}$.

Wir setzen $\rho_{\mathfrak{B}} := n_{\mathfrak{B}}/m_{\mathfrak{B}}$. (Dies sei ∞, falls $m = 0$.) Gewiß existiert das Minimum aller $\rho_{\mathfrak{B}}$ für alle Adjunktionsbäume. Es ist leicht zu zeigen, daß dieses $= 0$ sein muß. Es existiert also ein Adjunktionsbaum, der nur aus $\mathtt{t}$, $\mathtt{u}$, $\mathtt{v}$ und $\mathtt{w}$ besteht, jeweils in gleicher Anzahl. Ferner existiert ein Adjunktionsbaum, welcher $\mathtt{a}$ enthält.

Wir betrachten nun eine Zeichenkette $\vec{x}$ aus $L(G)$ mit folgender Eigenschaft. Es ist

$$\mu(\vec{x}) = m(\mathtt{a} + \mathtt{b} + \mathtt{c} + \mathtt{d}) + n(\mathtt{t} + \mathtt{u} + \mathtt{v} + \mathtt{w})$$

für gewisse natürliche Zahlen m und n derart, daß (a) m größer ist als jedes $m_{\mathfrak{B}}$, und (b) n/m ist kleiner als jedes $\rho_{\mathfrak{B}}$, welches nicht 0 ist. Man überlegt sich leicht, daß ein solches $\vec{x}$ existieren muß. Sind m und n gewählt, so erfüllt folgende Zeichenkette unseren Zweck.

$$\mathtt{a}\mathtt{t}^n\mathtt{u}^n\mathtt{a}^{m-1}\mathtt{b}^{n-1}\mathtt{v}^n\mathtt{w}^n\mathtt{b}\mathtt{c}^m\mathtt{d}^m$$

Diese Zeichenkette ist aus einem Zentralbaum entstanden, indem (a′) ein $\mathfrak{A}$ adjungiert worden ist, in dem $\mathtt{a}$ vorkommt, (b′) ein $\mathfrak{B}$ adjungiert worden ist, in welchem $\mathtt{a}$ nicht vorkommt. Wir betrachten nun die Punkte, in welche $\mathfrak{B}$ eingehängt worden ist. Diese können nur wie folgt sein:

$$\mathtt{a}\mathtt{t}^n \bullet \mathtt{u}^n\mathtt{a}^{m-1}\mathtt{b}^{n-1}\mathtt{v}^n \bullet \mathtt{w}^n\mathtt{b}\mathtt{c}^m\mathtt{d}^m$$

Betrachten wir nun aber, wo der Adjunktionsbaum $\mathfrak{A}$ eingehängt worden ist:

$$\mathtt{a}\mathtt{t}^n\mathtt{u}^n\mathtt{a}^{m-1} \circ \mathtt{b}^{n-1}\mathtt{v}^n\mathtt{w}^n\mathtt{b}\mathtt{c}^m \circ \mathtt{d}^m$$

Überlagern wir dies, so erhalten wir

$$\mathtt{a}\mathtt{t}^n \bullet \mathtt{u}^n\mathtt{a}^{m-1} \circ \mathtt{b}^{n-1}\mathtt{v}^n\mathtt{w}^n \bullet \mathtt{b}\mathtt{c}^m \circ \mathtt{d}^m$$

Nun haben wir einen Widerspruch: diese Adjunktionspunkte dürfen nicht überkreuzen! Dann das zwischen den $\bullet$ liegende Stück muß eine Konstituente sein, ebenso das zwischen den $\circ$ liegende Stück. Aber diese Konstituenten liegen nicht ineinander. (Damit daraus ein Beweis wird, muß man sich überlegen, daß die Konstituentstruktur durch Adjunktion nicht geändert wird. Dies ist eine Übungsaufgabe.)

Nun haben wir also eine Kopfgrammatik, welche eine Sprache erzeugt, die nicht durch eine BAG erzeugt werden kann. Wir zeigen nun obendrein, daß Kopgfgrammatiken schwächer sind als 2–verzweigende lineare 2–LBGs, und diese wiederum schwächer als lineare 2–LBGs. Einige Teile dieser Argumentation werden allerdings in die Übungen verlagert, da sie nicht von zentralem Interesse sind.

Definition 4.5.6 *Eine lineare LBG heißt n–**verzweigend**, falls die Polynombasis aus höchstens k–stelligen Vektorpolynomen besteht.*

Der Grund für diese Definition ist der folgende Sachverhalt.

Proposition 4.5.7 *Es sei $S = L(G)$ für eine n–verzweigende, k–lineare LBG. Dann existiert eine k–lineare LBG H mit $L(H) = S$, bei welcher jede Regel höchstens n–stellig ist.*

Dazu muß man sich überlegen, daß eine höherstellige Regel in kanonischer Weise durch eine Folge n–stellige Regeln ersetzt werden kann, falls das zugehörige Vektorpolynom durch höchstens n–stellige Polynome erzeugt wird. Auf der anderen Seite ist nicht garantiert, daß keine n–verzweigende Grammatik existiert, falls höherstellige Polynome verwendet wurden. Trotzdem läßt sich immer eine Sprache konstruieren, bei der die höherstelligen Polynome wesentlich verwendet werden. Sei zum Beispiel diese Sprache gegeben. Wir defineren

$$\vec{x}_n := \mathtt{t}^n\mathtt{u}^n, \qquad \vec{y}_n := \mathtt{v}^n\mathtt{w}^n \ .$$

Das folgende Polynom ist nicht durch zwei- oder dreistellige Polynome darstellbar.

$$\mathfrak{q}(\langle w_0, w_1\rangle, \langle x_0, x_1\rangle, \langle y_0, y_1\rangle, \langle z_0 z_1\rangle) := \langle w_0 x_0 y_0 z_0, y_1 w_1 z_1 x_1\rangle$$

Daraus läßt sich ein Beweis stricken, daß die folgende Sprache nicht durch eine 2–verweigende LBG erzeugt werden kann.

$$L = \{\vec{x}_{n_0}\vec{x}_{n_1}\vec{x}_{n_2}\vec{x}_{n_3}\vec{y}_{n_2}\vec{y}_{n_0}\vec{y}_{n_3}\vec{y}_{n_1} : n_0, n_1, n_2, n_3 \in \omega\}$$

Wir wollen dies nicht ausführen, sondern uns der Frage nach den Kopfgrammatiken stellen. Wir wollen zeigen, daß es eine Sprache gibt, welche durch eine 2–verzweigende LBG erzeugt werden kann, aber nicht durch eine Kopfgrammatik. Dies ist die Sprache

$$M := \{\vec{x}_{n_0}\vec{x}_{n_1}\vec{y}_{n_0}\vec{y}_{n_1} : n_0, n_1 \in \omega\}$$

Zunächst einmal die Grammatik, welche diese Sprache erzeugt.

$$\begin{array}{lcl} \mathtt{S}(x_0x_1, y_0y_1) & \leftarrow & \mathtt{H}(x_0, x_1), \mathtt{H}(y_0, y_1) \\ \mathtt{H}(\varepsilon, \varepsilon) & \leftarrow & . \\ \mathtt{H}(\mathtt{t}x_0\mathtt{u}, \mathtt{v}x_0\mathtt{w}) & \leftarrow & \mathtt{H}(x_0, x_1) \end{array}$$

Offensichtlich ist dies eine 2–verzweigende LBG. Wir können annehmen, diese Grammatik ist monoton. Nun werden wir zeigen, daß Kopfgrammatiken diese Sprache nicht erzeugen können. Dazu eine Definition. Es sei $\vec{x}$ ein Wort, und $C = \langle \vec{y}_0, \vec{y}_1 \rangle$ sowie $D = \langle \vec{z}_0, \vec{z}_1 \rangle$ disjunkte Konstituenten. Wir sagen, C liege **links** (**rechts**) von D, falls alle Segmente von C links (rechts) von jedem Segment von D liegen; C ist **zentral** in D, falls $\vec{z}_0 \sqsubset \vec{y}_0$ und $\vec{y}_1 \sqsubset \vec{z}_1$. Schließlich sagen wir, C und D **kreuzen**, falls $\vec{y}_0 \sqsubseteq \vec{z}_0 \sqsubset \vec{y}_1 \sqsubset \vec{z}_1$ oder $\vec{z}_0 \sqsubset \vec{y}_0 \sqsubset \vec{z}_1 \sqsubset \vec{y}_1$.

Lemma 4.5.8 *Es sei G eine Kopfgrammatik und $\mathfrak{T}$ ein durch γG erzeugter Strukturbaum. Dann existieren in $\mathfrak{T}$ keine kreuzenden Konstituenten.*

Nun kommen wir zu unserer Sprache M. Wir werden zeigen, daß jede sie erzeugende Grammatik zwei kreuzende Konstituenten besitzen muß. Es sei eine Zeichenkette $\vec{x}_0\vec{x}_1\vec{y}_0\vec{y}_1$ gegeben. Ist n_0 und n_1 groß genug, existiert eine Konstituente vom Typ A, welche eine weitere Konstituente vom Typ A enthält. Man überlegt sich nun leicht, daß A entweder in $\vec{x}_0\vec{y}_0$ oder in $\vec{x}_1\vec{y}_1$ enthalten sein muß. (Denn wird A gepumpt, so muß die gleiche Anzahl von , $\mathtt{u}$, $\mathtt{v}$ und $\mathtt{w}$ gepumpt werden. Man bekommt automatisch vier Teilworte der Form $\mathtt{t}^p$, $\mathtt{u}^p$, $\mathtt{v}^p$ und $\vec{\mathtt{w}}^p$, welche gleichzeitig gepumt werden. Damit dies aufgeht, müssen sie alle Teilworte von $\vec{x}_0\vec{y}_0$ oder von $\vec{x}_1\vec{y}_1$ sein.) Nun seien n_0 und n_1 hinreichend groß. Dann enthält sowohl $\vec{x}_0$ als auch $\vec{x}_1$ zwei (echt ineinander enthaltene) Konstituenten gleichen Typs. Das bedeutet, daß wir jetzt annehmen dürfen, $\vec{x}_0\vec{y}_0$ enthält zwei verschiedene Konstituenten C_0 und C_1 vom Typ A und $\vec{x}_1\vec{y}_1$ zwei Verschiedene Konstituenten D_0 und D_1 vom Typ B. Man sieht jetzt leicht, daß C_0 und D_0 kreuzende Konstituenten sind. Denn weder C_0 noch D_0 sind leer. Ferner enthält C_0 mindestens ein $\mathtt{t}$, ein $\mathtt{u}$, ein $\mathtt{v}$ und ein $\mathtt{v}$; desgleichen für D_0. Also muß $C_0 = \langle \mathtt{t}^{p_0}\mathtt{u}^{p_1}, \mathtt{v}^{p_2}\mathtt{w}^{p_3} \rangle$ sowie $D_0 = \langle \mathtt{t}^{q_0}\mathtt{u}^{q_1}, \mathtt{v}^{q_2}\mathtt{w}^{q_3} \rangle$ sein. Das linke Segment von C_0 ist in $\vec{x}_0$ enthalten, das rechte in $\vec{y}_0$; das linke Segment von D_0 ist in $\vec{x}_1$ enthalten, das rechte in $\vec{y}_1$.

Übung 103. Es sei $\mathfrak{B}$ ein Baum und $\mathfrak{A}$ ein Adjunktionsbaum. Es sei $\mathfrak{C}$ das Resultat der Adjunktion von $\mathfrak{A}$ an x in $\mathfrak{B}$. Wir fassen $\mathfrak{B}$ in natürlicher Weise als Teilbaum von $\mathfrak{C}$ auf, ist x der untere Knoten von $\mathfrak{A}$ in $\mathfrak{C}$. Man zeige: die Konstituenten von $\mathfrak{B}$ sind gerade die Schnitte von Konstituenten von $\mathfrak{C}$ mit der Menge der Knoten von $\mathfrak{B}$.

Übung 104. Zeigen Sie, daß die Sprache $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n : n \in \omega\}$ nicht durch eine unregulierte BAG erzeugt werden kann. *Hinweis.* Gehen Sie wie in dem obigen Beweis vor. Betrachten Sie eine Zeichenkette, die groß genug ist, daß ein Baum adjungiert worden ist und analysieren Sie die Adjunktionsstellen.

Übung 105. Zeigen Sie, daß in dem Beispiel oben $min\{\rho_{\mathfrak{B}} : \mathfrak{B} \in \mathsf{A}\} = 0$. *Hinweis.* Vergleichen Sie die Ausführungen in Abschnitt 2.6.

Übung 106. Zeigen Sie: Zu jeder Baumadjunktionsgrammatik G existiert eine BAG $G^\diamondsuit$ in Standardform, derart, daß $G^\diamondsuit$ und G dieselben Konstituentenstrukturen er-

zeugen. Was kann man über die Markierungsfunktionen sagen?

Übung 107. Zeigen Sie Proposition 4.5.7.

Übung 108. Beweisen Sie Lemma 4.5.8.

4.6 Indizierte Grammatiken

Indizierte Grammatiken erweitern das Konzept der kontextfreien Grammatiken auf eine sehr spezielle Weise. Sie erlauben, zusätzlich zu einem Nichterminalsymbol eine Folge von Indizes zu manipulieren. Wir können indizierte Grammatiken allerdings auch als Regelschemata auffassen. Dazu sei wie üblich A unser Alphabet, N die Menge der Nichtterminalsymbole (disjunkt zu A) und I eine zu A und N disjunkte endliche Menge von sogenannten **Indizes**. Ferner sei $\sharp$ ein Symbol, welches nicht in $A \cup N \cup I$ auftritt. Ein Indexschema σ hat die Form

$$A \cdot \vec{\alpha} \to B_0 \cdot \vec{\beta}_0 \quad \dots \quad B_{n-1} \cdot \vec{\beta}_{n-1}$$

beziehungsweise die Form

$$A \cdot \vec{\alpha} \to a$$

wo $\vec{\alpha}, \vec{\beta}_i \in I^* \cup \{\sharp\}$ für $i < n$, sowie $a \in A$. Die Schemata der zweiten Art heißen **terminale Schemata**. Eine **Instantiierung** von σ ist eine Regel

$$A \cdot \vec{x}\vec{\alpha} \to B_0 \cdot \vec{y}_0\vec{\beta}_0 \quad \dots \quad B_{n-1} \cdot \vec{y}_{n-1}\vec{\beta}_{n-1}$$

wobei Folgendes gilt:

1. Ist $\vec{\alpha} = \sharp$, so $\vec{x} = \varepsilon$ und $\vec{y}_i = \varepsilon$ für alle $i < n$.
2. Ist $\vec{\alpha} \neq \sharp$, so gilt für alle $i < n$: $\vec{y}_i = \varepsilon$ oder $\vec{y}_i = \vec{x}$.
3. Für alle $i < n$: ist $\vec{\beta}_i = \sharp$, so $\vec{y}_i = \varepsilon$.

Für ein terminales Schema gelten analoge Bedingungen: ist $\vec{\alpha} = \sharp$, so gilt $\vec{x} = \varepsilon$. Ein Regelschema bezeichnet die Menge aller seiner Instantiierungen. Ist in dem Regelschema σ $\vec{\alpha} = \sharp$ sowie $\vec{\beta}_i = \sharp$ für alle $i < n$, so haben wir den klassischen Fall einer kontextfreien Regel. Wir nennen deshalb ein Indexschema **kontextfrei**, falls es diese Form hat. Wir nennen es **linear**, wenn $\vec{\beta}_i \neq \sharp$ für höchstens ein $i < n$. Kontextfreie Schemata sind also linear, aber nicht notwendig umgekehrt. Im Übrigen bedient man sich der folgenden suggestiven Notation. Man denotiert mit $A[\,]$ ein A mit einem (beliebigen) Stapel, während A kurz für $A\sharp$ steht. Man betrachte etwa diese Regel:

$$A[i] \to B[\,] \quad A \quad C[ij]$$

Diese ist eine andere Form für das Schema

$$Ai \to B \quad A\sharp \quad Cij\ ,$$

welches wiederum alle Regeln der folgenden Form beinhaltet:

$$A\vec{x}i \to B\vec{x} \quad A\sharp \quad C\vec{x}ij$$

,

Definition 4.6.1 *Eine **Indexgrammatik** ist ein Quintupel $G = \langle S, A, N, I, R\rangle$, wobei A, N, I paarweise disjunkte endliche Mengen sind, $S \in N$ das Startsymbol und R eine endliche Menge von Indexschemata über A, N und I. G heißt **linear**, falls alle seine Indexschemata linear sind.*

Der Ableitungsbegriff kann über Zeichenketten wie auch über Bäume formuliert werden. (Dazu benötigt man allerdings die Disjunktheit von A, N und I. Denn sonst sind die Kategoriesymbole nicht eindeutig aus der Zeichenkette rekonstruierbar.) Am einfachsten ist es, wenn man sich vorstellt, die Indexgrammatik sei eine Grammatik $\langle S, N, A, R\rangle$, wo im Unterschied zu einer kontextfreien Grammatik R jetzt nicht endlich ist sondern die Instantiierungsmenge einer endlichen Menge von Indexschemata. Dann übertragen sich alle Begriffe automatisch. Man überlege sich, daß zu jeder Indexgrammatik G eine Indexgrammatik in 2–Standardform (Chomsky–Normalform) existiert, welche die gleiche Sprache erzeugt.

Als Beispiel einer Indexgrammatik geben wir hier die folgende Grammatik G. Es sei $A = \{\mathtt{a}\}$, $N = \{\mathtt{S}, \mathtt{T}, \mathtt{U}\}$, $I = \{\mathtt{i}, \mathtt{j}\}$, sowie

$$\begin{array}{lcll} \mathtt{S}[\,] & \to & \mathtt{T}[\mathtt{j}] & \\ \mathtt{T}[\,] & \to & \mathtt{T}[\mathtt{i}] & \\ \mathtt{T}[\mathtt{i}] & \to & \mathtt{U}[\,] & \\ \mathtt{U}[\mathtt{i}] & \to & \mathtt{U}[\,] & \mathtt{U}[\,] \\ \mathtt{U}[\mathtt{j}] & \to & \mathtt{a} & \end{array}$$

Es gilt $L(G) = \{\mathtt{a}^{2^n} : n \in \omega\}$. Zum Beispiel betrachte man folgende Ableitung.

$$\begin{array}{lcllcl} \mathtt{S} & \to & \mathtt{Tj} & \to & \mathtt{Tji} \\ & \to & \mathtt{Tjii} & \to & \mathtt{Tjiii} \\ & \to & \mathtt{Ujii} & \to & \mathtt{UjiUji} \\ & \to & \mathtt{UjiUjUj} & \to & \mathtt{UjUjUjUj} \\ & \to & \mathtt{aUjUjUj} & \to & \mathtt{aaUjUj} \\ & \to & \mathtt{aaaUj} & \to & \mathtt{aaaa} \end{array}$$

Indexgrammatiken sind also ziemlich stark. Dennoch stellt sich heraus, daß Indexgrammatiken **PTIME**–Sprachen erzeugen können. (Im Wesentlichen kann man auch

für Indexgrammatiken einen Chart–Algorithmus angeben. Auch dieser benötigt nur polynomiell viel Zeit, analog wie bei den einfachen LBGn.) Von besonderem Interesse sind hier die linearen Indexgrammatiken.

Nun kommen wir zu der Gleichwertigkeit von LIGs und BAGs. Sei G eine LIG; wir wollen eine BAG konstrueiren, welche die gleichen Konstituentenstrukturen erzeugt. Wir werden in etwa den gleichen Beweis anstreben wie für kontextfreie Grammatiken. Die Idee ist wiederum, in einem Baum Knoten x und y gleicher Marke $X\vec{x}$ aufzusuchen. Dies kann jedoch fehlschlagen. Denn einerseits können wir zwar erwarten, daß wir Knoten finden, welche dasselbe Nichtterminalsymbol tragen, aber wir wissen dann nicht, ob sie den gleichen Stapel haben. Es kann sogar vorkommen, daß überhaupt kein solches Paar Knoten mit gleichem Stapel existiert. Dazu werden wir eine erste Vereinfachung vornehmen. Wir erlauben G nur Regeln der folgenden Form zu haben:

$$\begin{array}{lcl} X[i] & \to & Y_0 \dots Y_{j-1}\ Y_j[\,]\ Y_{j+1} \dots Y_{n-1} \\ X[\,] & \to & Y_0 \dots Y_{j-1}\ Y_j[i]\ Y_{j+1} \dots Y_{n-1} \\ X & \to & Y_0 \dots Y_{n-1} \\ X & \to & a \end{array}$$

Mit anderen Worten, wir erlauben nur, jeweils ein einzelnes Symbol auf den Stapel zu tun oder wegzunehmen. Eine solche Grammatik wollen wir **einfach** nennen. Es ist klar, daß wir G unter Beibehaltung der erzeugten Konstituentenstrukturen auf eine einfache Form bringen können. Dann haben wir stets folgende Eigenschaft: ist x ein Knoten mit Marke $X\vec{x}$ und dominiert x unmittelbar den Knoten x' mit Marke $Y\vec{x}i$, so existiert ein Knoten $y' \leq x'$ mit Marke $V\vec{x}i$, welcher unmittelbar einen Knoten mit Marke $W\vec{x}$ dominiert. Zumindest sind dann die Stapel gleich, aber wir haben noch nicht $Y = V$. Um dies zu erreichen, müssen wir noch einen zweiten Schritt tun. Wir erweitern die Menge der Nichtterminalsymbole zu der Menge aller Paare $\langle A, B\rangle$ mit $A, B \in N$. Ferner bekommt jedes Paar ein Superskript aus $\{o, a, e\}$. Die Superskripte notieren, ob wir an dieser Stelle einen Index kellern (a), hervorholen (e) oder gar nichts tun (o). Das Indexalphabet ist jetzt $N^2 \times I$. Die obenstehenden Regeln werden jetzt wie folgt reformiert (der Übersichtlichkeit halber sei stets $n = 3$ und $j = 1$). Für eine Regel vom zweiten Typ setzen wir alle Regeln der Form

$$\langle X, X'\rangle^a \to \langle Y_0, Y_0'\rangle^{a/o} \quad \langle Y_1, Y_1'\rangle^{a/o}[\langle X, X', i\rangle] \quad \langle Y_2, Y_2'\rangle^{a/o}$$

Wir kellern also zusätzlich zu dem Index i noch die Information über die Marke, mit welcher wir begonnen haben. Das Superskript a ist obligatorisch für $\langle X, X'\rangle$! Aus den Regeln des ersten Typs werden folgende Regeln:

$$\langle X, X'\rangle^{a/o}[\langle W, Y_1', i\rangle] \to \langle Y_0, Y_0'\rangle^{a/o} \quad \langle W, Y_1'\rangle^{e} \quad \langle Y_2, Y_2'\rangle^{a/o}$$

Wir fügen allerdings noch die folgenden Regeln hinzu:

$$(\S) \qquad \langle Y_1, Y_1'\rangle^e \to \langle Y_1', Z\rangle^{a/o}$$

für alle $Y_1, Y_1', Z \in N$. Die Regeln des dritten Typs werden wie folgt ersetzt.

$$\langle X, X' \rangle^o \to \langle Y_0, Y_0' \rangle^{a/o} \quad \langle Y_1, Y_1' \rangle^{a/o} \quad \langle Y_2, Y_2' \rangle^{a/o}$$

Schließlich ersetzen wir die Regeln des vierten Typs durch

$$\langle X, X' \rangle^o \to a$$

Wir nennen diese Grammatik $G^{\clubsuit}$. Wir wollen zunächst sehen, warum G und $G^{\clubsuit}$ die gleichen Konstituentenstrukturen erzeugen. Sei dazu eine $G^{\clubsuit}$–Ableitung gegeben. Wir erhalten dann eine G–Ableitung wie folgt. Jedes Symbol der Form $\langle X, X' \rangle^{a/e/o}$ wird ersetzt durch X, jedes Stapelsymbol $\langle X, X', i \rangle$ durch i. Anschließend werden die Regeln vom Typ (§) übersprungen. Dies liefert eine G–Ableitung, wie man sich leicht überlegt und erzeugt dieselbe Konstituentenstruktur. Sei umgekehrt eine G–Ableitung gegeben mit assoziiertem geordneten Baum $\mathfrak{B}$. Dann gehen wir von oben nach unten gehend im Baum wie folgt vor. Angenommen, eine Regel zweiten Typs wurde bei Knoten x angewendet und der Index i gekellert. Man suche den höchsten Knoten $y < x$ auf, bei dem der Index i hervorgeholt wird. Habe y die Marke B, x die Marke A. Dann ersetze A durch $\langle A, B \rangle^a$ und den Index i auf allen Knoten bis y durch $\langle A, B, i \rangle$. Zwischen x und y fügen wir einen Knoten y^* mit Marke $\langle A, B \rangle^e$ ein. y^* habe y als einzige Tochter. y behält zunächst noch die Marke B. Falls bei x allerdings kein Symbol gekellert wird, so wechsle man die Marke A aus durch $\langle A, A' \rangle^o$, wo A' beliebig ist. Ist man am Baum unten angekommen, so erhält man einen $G^{\clubsuit}$–Baum. Wiederum sind die Konstituentenstrukturen erhalten geblieben, da nur einstellige Regeln eingefügt worden sind.

Jetzt gilt Folgendes. Wird bei x der Index $\langle A, B, i \rangle$ gekellert, so hat x die Marke $\langle A, B \rangle^a$ und es existiert ein Knoten y unterhalb von x, bei dem dieser Index wieder entfernt wird und der die Marke $\langle A, B \rangle^e$ trägt. Wir sagen, y sei **assoziiert** zu x. Man definiere jetzt also wie im Falle kontextfreier Sprachen Zentralbäume als Bäume, deren assoziierte Zeichenkette terminal ist und in denen kein Paar assoziierter Knoten existiert. Man überlegt sich, daß in solchen Bäumen kein Symbol gekellert wird. Kein Knoten trägt daher ein Kellersymbol. Daher existieren auch nur endlich viele solche Bäume. Nun definiert man die Adjunktionsbäume. Dies sind Bäume, bei denen die Marke der Wurzel die Form $\langle A, B \rangle^a$ hat, genau ein Blatt eine nichtterminale Marke, und diese ist $\langle A, B \rangle^e$. Ferner soll innerhalb des Baum kein Paar von assoziierten Knoten existieren. Wiederum ist klar, daß es nur endlich viele solcher Bäume geben kann. Dies bilden unsere Adjunktionsbäume, wobei wir nur noch Folgendes tun. Die Marken $\langle X, X' \rangle^o$ ersetzen wir durch $\langle X, X' \rangle$, die Marken $\langle X, X' \rangle^a$ und $\langle X, X' \rangle^e$ durch $\langle X, X' \rangle^{\triangledown}$. (Wurzel und assozierter Knoten bekommen also ein Adjunktionsverbot.) Ab jetzt läuft der Beweis wie im kontextfreien Fall.

Nun sei umgekehrt eine Baumadjunktionsgrammatik $G = \langle \mathsf{C}, \mathsf{A} \rangle$ gegeben. Wir wollen eine LIG bauen, welche die gleichen Konstituentenstrukturen erzeugt. Dazu

nehmen wir an, alle Bäume aus C und A seien auf paarweise disjunkten Mengen von Knoten realisiert. Es sei K die Vereinigung dieser Knotenmengen. Diese ist gleichzeitig unsere Menge der Nichtterminalsymbole; die Menge A ist unsere Menge der Indizes. Nun formulieren wir die Regeln. Es sei $i \to j_0 \quad j_1 \dots j_{n-1}$ ein lokaler Teilbaum eines Baumes. (A) Es sei i nicht zentral. Dann sei

$$i \to j_0 \quad j_1 \dots j_{n-1}$$

eine Regel. (B) Es sei i Wurzel von $\mathfrak{T}$ und j_k zentral (und damit nicht ausgezeichnetes Blatt). Dann sei

$$i[\,] \to j_0 \quad j_{k-1} \quad j_k[\mathfrak{T}] \quad j_{k+1} \dots j_{n-1}$$

eine Regel. (C) Es sei j_k ausgezeichnetes Blatt in $\mathfrak{T}$. Dann sei

$$i[\mathfrak{T}] \to j_0 \quad j_{k-1} \quad j_k[\,] \quad j_{k+1} \dots j_{n-1}$$

eine Regel. (D) Es sei i in $\mathfrak{T}$ zentral, aber keine Wurzel, und j_k zentral aber nicht ausgezeichnetes Blatt. Dann sei

$$i[\,] \to j_0 \dots j_{k-1} \quad j_k[\,] \quad j_{k+1} \dots j_{n-1}$$

eine Regel. Nichts sonst sei eine Regel. Dies definiert die Grammatik G^I. (Diese hat, wie man leicht sieht, offensichtlich Startbäume über verschiedenen Startsymbolen. Dies läßt sich aber leicht beheben.) Wir behaupten nun, daß diese Grammatik dieselben Konstituentenstrukturen über A erzeugt. Dies tun wir durch Induktion über die Ableitung. Sei $\mathfrak{T}$ ein Zentralbaum, etwa $\mathfrak{T} = \langle B, <, \sqsubset, \ell\rangle$. Dann sei $\mathfrak{T}^I := \langle B, <, \sqsubset, \ell^I\rangle$, wo $\ell^I(i) := i$, falls i nichtterminal, und $\ell^I(i) := \ell(i)$ sonst. Man überlegt sich leicht, daß dieser Baum ableitbar ist. Nun seien $\mathfrak{T} = \langle B, <, \sqsubset, \ell\rangle$ und $\mathfrak{T}^I = \langle B, <, \sqsubset, \ell^I\rangle$ bereits konstruiert; es entstehe $\mathfrak{U} = \langle C, <', \sqsubset', \ell'\rangle$ aus $\mathfrak{T}$ durch Adjungieren eines Baumes $\mathfrak{B}$ an einen Knoten x. Indem wir x zur Wurzel des adjungierten Baumes machen, haben wir $B \subset C$, $<' \cap B^2 = <$, $\sqsubset' \cap B^2 = \sqsubset$ und $\ell' \restriction B = \ell$. Nun ist $\mathfrak{U}^I = \langle C, <', \sqsubset', \ell'^I\rangle$. Ferner existiert ein Isomorphismus zwischen dem Adjunktionsbaum $\mathfrak{B}$ und dem lokalen Teilbaum induziert auf $C \cup \{x\}$. Es sei $\pi : C \cup \{x\} \to B$ dieser Isomorphismus. Es sei $\ell'^I(y) := \ell^I(y)$, falls $y \in B - C$. Es sei $\ell'^I(y) := \pi(y)$, falls $\pi(y)$ nicht zentral; es sei $\ell'^I(y) := \ell'^I(x) := \ell^I(x)$, falls y ausgezeichnetes Blatt ist. Schließlich sei $\ell^I(x) = X\vec{x}$, so X ein Nichtterminalsymbol und $\vec{x} \in I^*$. Ist dann y zentral aber nicht Wurzel oder Blatt, so setze nunmehr

$$\ell'^I(y) := \pi(y)\vec{x}\mathfrak{B}$$

Nun rechnet man leicht nach, daß der so definierte Baum tatsächlich in G^I ableitbar ist. Nun muß man ebenso zeigen, daß wenn $\mathfrak{U}$ in G^I ableitbar ist, so existiert ein Baum $\mathfrak{U}^A$ mit $(\mathfrak{U}^A)^I \cong \mathfrak{U}$, welcher in G ableitbar ist. Dazu bedient man sich der Methode des Aushängens. Man sucht Knoten x und y derart, daß sie gleichen Stapel besitzen, $x > y$, keine Element dazwischen gleichen Stapel besitzt. Ferner soll kein

solches Paar in $\downarrow x - (\downarrow y \cup \{x\})$ existieren. Hat man solch einen Baum, so sieht man leicht, daß er isomorph zu einem Adjunktionsbaum ist. Man hängt diesen aus und bekommt so einen echt kleineren Baum. (Die Existenz eines solchen Baumes muß man sich noch sichern. Das geschieht aber wie im kontextfreien Fall. Wähle nämlich x von minimaler Höhe derart, daß ein $y < x$ mit gleichem Stapel existiert. Anschließend wähle y maximal mit dieser Eigenschaft. In $\downarrow x - (\downarrow y \cup \{x\})$ kann nun kein weiteres Paar Knoten x', y' gleichen Stapels existieren mit $y' < x'$. Sonst wäre x ja nicht minimal.) Fassen wir zusammen.

Theorem 4.6.2 *Genau dann ist eine Menge von Konstituentenstrukturen erzeugt von einer lineare Indexgrammatik, wenn sie durch eine Baumadjunktionsgrammatik erzeugt wird.*

Man sagt auch, diese Grammatiktypen sind konstituentenäquivalent.

Eine Regel heißt **rechtslinear**, falls der Index nur an die rechte Tochter vererbt wird. Es ist also die linke Regel rechtslinear, die rechte nicht:

$$A[\,] \to B \quad C[i] \quad B, \qquad A[\,] \to B \quad C \quad B[i]$$

Eine Indexgrammatik heißt **rechtslinear**, falls jede Regel rechtslinear ist. (Damit ist sie automatisch auch linear.)

Theorem 4.6.3 (Michaelis & Wartena) *Genau dann wird eine Sprache durch eine rechtslineare Indexgrammatik erzeugt, wenn sie kontextfrei ist.*

Beweis. Es sei G rechtslinear, $X \in N$. Definiere H_X wie folgt. Das Nichtterminalalphabet hat die Form $T := \{X^\diamond : X \in N\}$, das Terminalalphabet ist das von G, ebenso das Indexalphabet. Das Startsymbol sei X. Nun wird zu jeder Regel

$$A[\,] \to B_0 \dots B_{n-1} \quad B[i]_n$$

die Regel

$$A^\diamond[\,] \to A \quad B^\diamond[i]$$

aufgenommen. Diese Grammatik ist sogar rechtsregulär und erzeugt eine kontextfreie Sprache (siehe die Übungen). Es existiert mithin eine kontextfreie Grammatik $L_X := \langle S_X^L, N_X^L, N, R_X^L \rangle$, welche $L(H_X)$ erzeugt. (Hiebei ist also N das Nichtterminalalphabet von G, aber das Terminalalphabet von L_X.) Wir nehmen an, daß N_L^X zu unseren bisherigen Alphabeten disjunkt ist. Wir setzen jetzt $N' := \bigcup N_X^L \cup N$, sowie $R' := \bigcup R_X^L \cup R \cup R^-$, wo R die Menge der kontextfreien Regeln von G ist und R^- die Menge der Regeln $A[\,] \to B_0 \dots B_{n-1}$ derart, daß $A[\,] \to B_0 \dots B_{n-1} \quad B_n[i] \in R$. Endlich sei $G' := \langle S_L, N', A, R' \rangle$. G' ist sicher kontexfrei. Es bleibt zu zeigen, daß

$L(G') = L(G)$. Dazu sei $\vec{x} \in L(G)$. Es existiert folglich ein Baum $\mathfrak{B}$ mit assoziierter Zeichenkette $\vec{x}$, welcher von G abgeleitet wird. Induktiv über die Höhe dieses Baumes wird gezeigt, daß $\vec{x} \in L(G')$. Die Induktionsbehauptung ist genauer diese: *zu jedem G–Baum $\mathfrak{B}$ mit assoziierte Zeichenkette $\vec{x}$ existiert ein G'–Baum $\mathfrak{B}'$ mit assoziierter Zeichenkette $\vec{x}$; und falls die Wurzel von $\mathfrak{B}$ die Marke $X\vec{x}$ trägt, so trägt die Wurzel von $\mathfrak{B}'$ die Marke X.* Falls $\mathfrak{B}$ keine Stapelsymbole enthält, ist die Behauptung wahr; man nehme $\mathfrak{B}' := \mathfrak{B}$. Ferner ist die Behauptung leicht zu sehen, falls die Wurzel durch eine kontextfreie Regel expandiert wurde. Sei dies jetzt nicht mehr der Fall; habe der Baum eine Wurzel mit Marke U. Es sei P die Menge der rechten Knoten von $\mathfrak{B}$. Für jedes $x \in P$ sei $B(x)$ derjenige Baum, der alle Knoten enthält, welche unterhalb von x sind aber nicht unterhalb von einem $y \in P$ mit $y < x$. Es ist leicht zu sehen, daß diese Mengen eine Partition von $\mathfrak{B}$ bilden. Sei $u \prec x$, $u \notin P$. Der von u dominierte Baum kann nach Induktionsvoraussetzung in einen Baum $\mathfrak{T}_u$ umgebaut werden, welcher dieselbe assoziierte Zeichenkette besitzt sowie dieselbe Wurzelmarke und der von G' erzeugt wird. Der lokale Baum von x in $B(x)$ ist damit eine Instanz einer Regel aus R^-. Wir bezeichnen den zu x erhaltenen Baum mit $\mathfrak{B}'_x$. Es ist $\mathfrak{B}'_x$ ein G'–Baum. Ferner gilt: ist $y < x$, $y \in P$, und ist $u < x$, so ist $u \sqsubset y$. Es sei also $P = \{x_i : i < n\}$ eine Aufzählung mit $x_i > x_{i+1}$ für alle $i < n-1$. Sei A_i die Wurzelmarke von x_i in $\mathfrak{B}'_{x_i}$. Das Wort $\prod_{i<n} A_i$ ist ein Wort von H_U. Infolgedessen wird es von L_U erzeugt. Daher wird es auch von G' erzeugt. Es existiert damit ein Baum $\mathfrak{C}$ zu diesem Wort. Seien die Blätter des Baumes gerade die x_i und habe x_i die Marke A_i. Dann hängen wir jetzt $\mathfrak{B}'_{x_i}$ an die Stelle von x_i für alle $i < n$. Dies definiert $\mathfrak{D}$. Es ist $\mathfrak{D}$ ein G'–Baum mit assoziierter Zeichenkette $\vec{x}$. Die Umkehrung sei dem Leser überlassen. $\dashv$

Kombinatorische Kategorialgrammatiken hatten wir schon im Abschnitt 3.2 eingeführt. Das Konzept war allerdings sehr weit gefaßt. In der Literatur wird, Mark Steedman folgend, unter einer CCG im Wesentlichen eine solche um Kombinatoren erweiterte Kategorialgrammatik verstanden, bei der die Kombinationsregeln durch die generalisierten Funktionskompositionen sowie Applikation definiert sind. Um eine möglichst einheitliche Notation zu ermöglichen, definieren wir wie folgt:

$$\begin{aligned} \alpha \mid^+ \beta &:= \alpha/\beta \\ \alpha \mid^- \beta &:= \beta\backslash\alpha \end{aligned}$$

Wir nehmen p_i als Variable für Zeichen aus $\{+,-\}$. Eine **Kategorie** ist eine wohlgeformte Zeichenkette über $\{B, (,), |^+, |^-\}$. Wie üblich wird Linksklammerung vereinbahrt, womit jetzt Klammern entfallen. Diese sollen allerdings jetzt in der Zeichenkette nicht existieren. Also ist $\mathtt{a} \mid^+ \mathtt{b} \mid^- \mathtt{c}$ eine Kategorie wie auch $\mathtt{a} \mid^+ \mathtt{(b} \mid^- \mathtt{c)}$, $\mathtt{((a} \mid^+ \mathtt{b)} \mid^- \mathtt{c)}$ sowie $\mathtt{(a} \mid^+ \mathtt{(b} \mid^- \mathtt{c))}$ jedoch nicht. Ein **Block** ist eine Kette der Form $|^+ \beta$ oder der Form $|^- \beta$, wo β ein Kategoriesymbol ist. Eine **p–Kategorie** ist eine Folge von Blöcken, aufgefaßt als Zeichenkette. Damit ist eine Kategorie schlicht eine Zeichenkette der Form $\alpha \cdot \Delta$, wo Δ eine p–Kategorie ist. Sind Δ und Δ' p–Kategorien, so auch $\Delta \cdot \Delta'$. Ist α eine Kategorie, so wird der Kopf von α, $K(\alpha)$

induktiv wie folgt definiert.

1. $K(b) := b$.
2. $K(\alpha \mid^p \beta) := K(\alpha)$.

Lemma 4.6.4 *Jede Kategorie α kann zerlegt werden in $\alpha = K(\alpha) \cdot \Delta$, wo Δ eine p–Kategorie ist.*

Fassen wir die Folge schlicht als Zeichenkette auf, so können wir $\cdot$ als Konkatenation von Blöcken beziehungsweise Folgen von Blöcken benutzen. Wir erlauben folgende Operationen.

$$\begin{array}{llll} \alpha \mid^+ \beta & \circ_1 & \beta & := \alpha \\ \beta & \circ_2 & \alpha \mid^- \beta & := \alpha \\ \alpha \mid^+ \beta & \circ_3^n & \beta \cdot \Delta^n & := \alpha \cdot \Delta^n \\ \beta \cdot \Delta^n & \circ_4^n & \alpha \mid^- \beta & := \alpha \cdot \Delta^n \end{array}$$

Hierbei ist Δ^n eine Variable für p–Kategorien bestehend aus n Blöcken. Es ist zusätzlich möglich, die Wahl der Köpfe von α und β einzuschränken. Dies bedeutet, daß wir Operationen $\circ_i^{F,A,n}$ definieren müssen, dergestalt, daß

$$\alpha \circ_i^{L,R,n} \beta := \begin{cases} \alpha \circ^n \beta, & \text{falls } K(\alpha) \in L, K(\beta) \in R \\ \star, & \text{sonst.} \end{cases}$$

Dies bedeutet also, daß wir von unserem Ideal abweichen, die Typen allein implizit durch die Kombinatoren bestimmt zu lassen.

Definition 4.6.5 *Eine **Kombinatorische Kategorialgrammatik** ist eine Kategorialgrammatik, welche endlich viele der Operationen aus $\{\circ_1^{L,R}, \circ_2^{L,R} : L, R \subseteq B\} \cup \{\circ_3^{L,R,n}, \circ_4^{L,R,n} : n \in \omega, L, R \subseteq B\}$ verwendet.*

Man beachte im Übrigen, daß $\circ_1 = \circ_3^0$ und $\circ_2 = \circ_4^n$ ist. Dies erleichtert das Rechnen.

Theorem 4.6.6 *Zu jeder Kombinatorischen Kategorialgrammatik G existiert eine lineare Indexgrammatik H, welche die gleichen Bäume erzeugt.*

Beweis. Es sei G gegeben. Insbesondere ordnet G jedem Buchstaben $a \in A$ eine endliche Menge $\zeta(a)$ von Kategorien zu. Wir betrachten die Menge M der Teilterme von Kategorien aus $\bigcup\langle\zeta(a) : a \in A\rangle$. Dies ist eine endliche Menge. Wir setzen $N := M$ und $I := M$. Kategorien können wir gemäß Lemma 4.6.7 immer als Paare

der Form $\alpha[\Delta]$ schreiben, wo $\alpha \in N$ und Δ eine p–Kategorie über I. Ferner existieren endlich viele Operationen, welche wir auch als Regeln schreiben können. Es sei etwa $\circ_1^{L,R}$ eine Operation. Dies bedeutet, daß wir Regeln der Form

$$\alpha \to \alpha|^+\beta \quad \beta$$

besitzen, wo $K(\beta) \in L$ und $K(\alpha) \in R$. Dies schreiben wir in lineare Indexregeln um. Man beachte, daß in jedem Fall $\beta \in M$ sein muß, wegen Lemma 4.6.7. Ferner muß $\alpha \in M^+$ sein. Wir schreiben also alle Regeln der Form

$$(\dagger) \qquad \alpha \to \delta[\Delta] \quad \beta$$

auf, wo $\delta[\Delta] = \alpha|^+\beta$ ist für irgendwelche $\alpha, \Delta \in M^*$ und $\delta, \beta \in M$. Diese kann man in endlich viele lineare Regelschemata zusammenfassen. Man fixiere dazu β, wobei $K(\beta) \in R$. Es sei $\mathbb{B}$ die Menge aller Folgen $\langle \gamma_i : i < p \rangle \in M^*$, deren Konkatenation gerade $|^+\beta$ ist. $\mathbb{B}$ ist endlich. Jetzt setze für (†) alle Regeln der Form

$$(\ddagger) \qquad \alpha'[] \to \alpha'[\Delta] \quad \beta$$

wobei $\alpha' \in M$ beliebig mit $K(\alpha) \in L$ und $\Delta \in \mathbb{B}$. Jetzt kann man leicht sehen, daß jede Instanz von (†) eine Instanz von (‡) ist und umgekehrt.

Analog geht man für die Regel der folgenden Form vor.

$$\beta \to \alpha|^+\beta \quad \beta$$

Desgleichen erhalten wir aus den Operationen $\circ_3^{L,R,n}$ Regeln der Form

$$(\star) \qquad \alpha \cdot \Delta^n \to \alpha|^+\beta \quad \beta \cdot \Delta^n$$

wo $K(\alpha) \in L$ und $K(\beta) \in R$. Nun gilt, wiederum wegen Lemma 4.6.7, daß $\Delta^n \in M^n$ und daß $\beta \in M$. Einzig α ist also wieder beliebig groß. Dennoch bekommen wir $\alpha \in M^+$, aufgrund von Lemma 4.6.7. Deswegen entspricht $(\star)$ lediglich endlich vielen linearen Indexschemata. ⊣

Es gilt im Übrigen nicht die Umkehrung; denn die Bäume, welche eine LIG erzeugt, müssen nicht 2–fach verzweigend sein. Allerdings sind beide Grammatiktypen schwach äquivalent, das heißt, sie erzeugen die gleichen Sprachen.

Lemma 4.6.7 *Es sei G eine Kombinatorische Kategorialgrammatik über A und M die Menge der Kategorien, welche Teilkategorien eines $\alpha \in \zeta(a)$, $a \in A$, sind. Dann gilt: ist $\vec{x}$ eine Zeichenkette des Typs α in G, so ist $\alpha = \beta \cdot \Delta$, wo $\alpha \in M$ und Δ eine p–Kategorie über M ist.*

Der Beweis erfolgt durch Induktion über die Länge von $\vec{x}$ und wird als Übung überlassen.

Übung 109. Eine lineare Indexgrammatik heißt *rechtsregulär*, falls Regeln entweder die Form $X \to aY[i]$, die Form $X[i] \to aY$, oder die Form $x \to aY$ haben, mit X und Y Nichtterminalsymbole. Zeigen Sie: genau dann ist S kontextfrei, wenn es die Sprache einer rechtsregulären linearen Indexgrammatik ist. *Hinweis.* Modellieren Sie die Aktionen eines Kellerautomaten mit Hilfe von rechtsregulären Regeln.

Übung 110. Zeigen Sie die folgende Behauptung. *Zu jeder Indexgrammatik G existiert eine Indexgrammatik H in 2–Standardform derart, daß $L(G) = L(H)$. Ist G linear (kontextfrei), so kann auch H linear (kontextfrei) gewählt werden.*

Übung 111. Beweise Sie Lemma 4.6.7.

4.7 Kompositionalität

In diesem Anschnitt wollen wir noch einmal die Diskussion um Kompositionalität aufgreifen. Wie wir schon in Abschnitt 3.1 gesehen haben, gibt es einige Begriffe von Kompositionalität, welche nicht so sehr die Sprache selbst als vielmehr die Wahl der Grammatik einschränken. Also sind in diesem Sinne nicht Sprachen kompositional, sondern eigentlich nur Grammatiken. Wir teilen diese Ansicht nicht. Unseres Erachtens gibt es durchaus nicht–kompositionale Sprachen. Entscheidend hierbei ist die Frage nach den zulässigen Modi. Ein n–stelliger Modus wird bestimmt durch ein Tripel n–stelliger Funktionen, welche jeweils die Exponenten, die Typen und die Bedeutungen manipulieren. Wir fordern nun insbesondere auf der Ebene der Exponenten, daß die auftretenden Funktionen nichts zerstören sollen. Auf diese Weise gewinnt der Begriff des Zusammenfügens wirklich einen Sinn. (Kompositionalität kommt vom Lateinischen compositiō, *das Zusammenfügen.*) Mehr noch: wir fordern, daß es keine sogenannten synkategorematischen Symbole gibt, das heißt, solche Symbole, die ein Modus einfügt, ohne daß sie zu einem der Teile gehören. Auf diese Weise ist jede Zeichenkette das Ergebniss eines Polynoms, angewendet auf gewisse Vektoren, und dieses Polynom bestimmt den strukturellen Aufbau sowie — indirekt — die Bedeutung und den Typ. Zunächst einmal die relevanten Definitionen.

Ist $\mathfrak{q} : (A^*)^{mn} \to (A^*)^n$ ein Vektorpolynom, so existieren Polynome p_i, $i < n$, derart, daß

$$\mathfrak{q}(\langle \vec{x}_i^j : i < n, j < m\rangle) = \langle p_i(\langle \vec{x}_i^j : j < m\rangle) : i < n\rangle$$

Wir nennen $\prod_{i<n} p_i(\langle x_i^j : j < n\rangle)$ das zu $\mathfrak{q}$ **assoziierte Polynom**. (Beachte: keine Vektorpfeile!) Dies ist ein gewöhnliches Zeichenkettenpolynom.

Definition 4.7.1 *Ein Polynom $p(\vec{x})$ heiße **rein**, wenn es keine Konstantensymbole*

enthält. Ein n–stelliges reines Polynom $p(\vec{x}) := \prod_{j<m} x_{i_j}$ *heiße* ***echt****, falls jedes* x_i *die Form* x_{j_k} *hat für ein* $k < m$*. Ein Vektorpolynom* $\mathfrak{q}$*, welches eine Funktion* $(A^*)^{mn} \to (A^*)^n$ *repräsentiert, heißt* ***rein*** *beziehungsweise* ***echt****, falls das assoziierte Polynom rein beziehungsweise echt ist.*

Nachdem wir nun die Wahl der Exponenten und die Funktionen darauf wie auch die Bedeutungen eingeschränkt haben, wollen wir als letztes noch die Wahl der Typen betrachten. Im Folgenden wollen wir annehmen, daß Zeichengrammatiken nur echte Polynome verwenden. Dies erlaubt uns, von Konstituenten zu reden, wie in Abschnitt 4.4 definiert. Denn sei $\vec{x}$ gegeben. Ist $\vec{x}$ ableitbar und wird bei der Ableitung ein Zeichen σ eingesetzt, so ist $\vec{x}$ das Ergebnis der Anwendung eines echten Polynoms auf den Exponenten von σ. Indem wir dies für alle Teilkonstituenten tun, bekommen wir eine Konstituentenstruktur. Nun definieren wir Folgendes.

Definition 4.7.2 *Eine Folge* $C = \langle \vec{w}_i : i < n+1 \rangle$ *heißt ein n–****Kontext****. Eine Folge* $\mathfrak{v} = \langle \vec{v}_i : i < n \rangle$ ***kommt in*** $\vec{x}$ ***im Kontext*** C ***vor****, falls gilt*

$$\vec{x} = (\prod_{i<n} \vec{w}_i \vec{v}_i) \cdot \vec{w}_n \ .$$

Wir schreiben dann auch $C(\mathfrak{v})$ *anstelle von* $\vec{x}$*.*

Man beachte, daß eine n–Folge von Zeichenketten auch wahlweise als $n-1$–Kontext betrachtet werden kann. Es sei nun Σ ein Zeichensystem, und $\mathtt{S}$ der ausgezeichnete Typ. Ferner sei

$$S(\Sigma) := \{\prod_{i<n} \vec{x}_i : \textit{es existiert } \mu : \langle \langle \vec{x}_i : i < n \rangle, \mathtt{S}, \mu \rangle \in \Sigma\}$$

Jetzt sei $\vec{x}$ ein Wort aus $S(\Sigma)$, welches wie folgt dekomponierbar ist:

$$\vec{x} = (\prod_{i<n} \vec{w}_i \vec{v}_i) \cdot \vec{w}_n$$

Ferner sei eine Zeichengrammatik G über n–Vektoren von Zeichenektten gegeben, welche Σ erzeugt, derart, daß das Vorkommen von $\mathfrak{v} := \langle \vec{v}_i : i < n \rangle$ in $\vec{x}$ eine Konstituente ist. Jetzt definieren wir

$$\mathbb{C}_\Sigma(\mathfrak{v}) := \{C : C(\mathfrak{v}) \in S(\Sigma)\}$$

Dies ist die **Kontextklasse** von $\mathfrak{v}$. Die grundlegende Idee ist nun, daß Typen nicht feiner unterscheiden dürfen als Kontextklassen.

Definition 4.7.3 *Es sei Σ ein Zeichensystem und G eine Grammatik, welche sie erzeugt. G ist **natürlich**, falls gilt: ist $\mathbb{C}_\Sigma(\mathfrak{v}) = \mathbb{C}_\Sigma(\mathfrak{w})$, so haben $\mathfrak{v}$ und $\mathfrak{w}$ stets denselben Typ. Dies bedeutet: ist $\langle \mathfrak{v}, \tau, \mu\rangle \in \Sigma$, so existiert ein μ' mit $\langle \mathfrak{w}, \tau, \mu'\rangle \in \Sigma$, und ist $\langle \mathfrak{w}, \tau, \mu\rangle \in \Sigma$, so existiert ein μ' mit $\langle \mathfrak{v}, \tau, \mu'\rangle \in \Sigma$.*

Definition 4.7.4 *Ein **vektorielles Zeichensystem** ist eine Menge $\Sigma \subseteq (A^*)^n \times T \times M$ für ein gewisses $n \in \omega$. Σ ist **kompositional**, falls es eine endliche Signatur Ω gibt und partielle Ω–Algebren $\mathfrak{Z} := \langle (A^*)^n, \{f^{\mathfrak{Z}} : f \in F\}\rangle$, $\mathfrak{T} := \langle T, \{f^{\mathfrak{T}} : f \in F\}\rangle$ und $\mathfrak{M} := \langle M, \{f^{\mathfrak{M}} : f \in M\}\rangle$ derart, daß alle Funktionen berechenbar sind und Σ genau die Menge der 0–stelligen Zeichen aus $\mathfrak{Z} \times \mathfrak{T} \times \mathfrak{M}$ ist. Σ ist **streng kompositional**, falls eine natürliche Zeichengrammatik für Σ existiert, in der für jedes $f \in F$ die Funktion $f^{\mathfrak{Z}}$ ein echtes Vektorpolynom ist.*

Die Definition von Kompositionalität ist in etwa der in der Literatur gebräuchliche (modulo Adaptation auf Zeichensysteme), während der Begriff der strengen Kompositionalität in etwa derjenige ist, welchen wir für den genuinen Begriff von Kompositionalität halten. Wir sagen, ein Vektorpolynom sei **inkrementell**, falls es nicht die nullstellige Konstante ε und nicht die Identität ist.

Theorem 4.7.5 *Es sei Σ ein streng kompositionales vektorielles Zeichensystem. Ferner sei zu jedem Modus M die Funktionen M^ε, M^τ und M^μ in **EXPTIME** und M^ε stets inkrementell. Dann existiert ein polynomieller Algorithmus, welcher zu gegebenem Vektor $\mathfrak{v}$ eine Bedeutung m und einen Typ t angibt, derart, daß $\langle \mathfrak{v}, t, m\rangle \in \Sigma$, falls diese existieren, und $\star$ sonst. Insbesondere ist die durch Σ definierte Zeichenkettensprache in **EXPTIME**.*

Zum Beweis notieren wir folgenden Sachverhalt.

Lemma 4.7.6 *Es sei Σ ein streng kompositionelles Zeichensystem mit natürlicher Grammatik G, in welcher alle Polynome echt und inkrementell sind. Es sei $\mathfrak{v}$ ein Vektor, dessen assoziierte Zeichenkette die Länge n hat, und t ein Strukturterm, dessen assoziierter Vektor gerade $\mathfrak{v}$ ist. Dann ist $|t| \leq n$.*

Der Beweis ist als Übung überlassen. Man muß nun einfach sämtliche Strukturterme der Länge $\leq n$ auflisten und schauen, ob sich unser Vektor unter den assoziierten Vektoren dieser Terme befindet. Für jedem Term benötigt man $O(2^{cn})$ Schritte, und es existieren $O(2^{dn})$ Terme (c und d sind positive reelle Zahlen). Man benötigt also insgesamt $O(2^{(c+d)n})$ Schritte. Falls man nun verlangt, die Funktionen M^ε, M^τ und M^μ seien in **PTIME**, so läßt sich dennoch nicht die Zeitschranke auf **PTIME** drücken. Ein Chart–Algorithmus ist ja nur deswegen polynomiell, weil er zwischen

verschiedenen Bedeutungen eines Vektors mit gewissem Typ nicht unterscheiden muß. Aber da die Bedeutungsfunktionen sensitiv gerade bezüglich Bedeutungen sind, kann es notwendig sein, alle verschiedenen Bedeutungen (also exponentiell viele) abzuspeichern. Dies kann zu exponentiellem Aufand führen. Wie wir weiter unten sehen werden, ist der Zusatz, daß die Funktionen inkrementell sein müssen, wichtig.

Es ist nicht unwichtig, noch einige Begriffsklärungen vorzunehmen.

Definition 4.7.7 *Es sei f ein Modus. Wir nehmen an, f^3 sei ein Vektorpolynom* $\mathtt{q}$*. Ist* $\mathtt{q}$ *nicht rein, sondern ist es ein Polynom, welches auch noch von einer Konstanten a Gebrauch macht, so heißt jedes Vorkommen von a, welches durch* $\mathtt{q}$ *in die Zeichenkette eingefügt wird, ein* ***synkategorematisches Vorkommen*** *von a.*

Ein wichtiges Beispiel sind die Klammern. In den gebräuchlichen Lehrbüchern zur Logik redet man von $\wedge$, $\vee$, $\rightarrow$ als logischen Symbolen, von den Klammern (und) dagegen als synkategorematischen Zeichen. Was dies besagt, ist, daß eine Zeichenkette der Form $\mathtt{(p0} \wedge \mathtt{p1)}$ als Konstituenten nur $\mathtt{p0}$, $\wedge$ sowie $\mathtt{p1}$ enthält. Die Klammern kommen gewissermaßen von außen dazu. Das zu der ersten der folgenden kontextfreien Regeln

$$\begin{array}{lcl} \mathtt{T} & \rightarrow & \mathtt{TF_2T} \\ \mathtt{F_2} & \rightarrow & \wedge \mid \vee \mid \rightarrow \end{array}$$

gehörende Polynom ist zum Beispiel das Polynom

$$p(x_0, x_1, x_2) := \mathtt{(} \cdot x_0 \cdot x_1 \cdot x_2 \cdot \mathtt{)}$$

Sind die Klammern allerdings nicht synkategorematisch, so haben wir zwei Möglichkeiten. Erstens: die Klammern sind Symbole der Sprache und eigenständige Konstituenten. Dann haben wir etwa folgende, kontextfreie Regeln.

$$\begin{array}{lcl} \mathtt{T} & \rightarrow & \mathtt{LTF_2TR} \\ \mathtt{F_2} & \rightarrow & \wedge \mid \vee \mid \rightarrow \\ \mathtt{L} & \rightarrow & \mathtt{(} \\ \mathtt{R} & \rightarrow & \mathtt{)} \end{array}$$

Oder es bilden die linke Klammer, das Infixsymbol und die rechte Klammer eine Einheit, welche zusammen eine Konstituente bildet. In diesem Fall müssen wir mit diskontinuierlichen Konstituenten arbeiten, welche aus drei Teilen bestehen.

$$\begin{array}{lcl} \mathtt{T}(\varepsilon, z_0x_0x_1x_2z_1y_0y_1y_2z_2, \varepsilon) & \leftarrow & \mathtt{T}(x_0, x_1, x_2) \quad \mathtt{T}(y_0, y_1, y_2) \quad \mathtt{F_2}(z_0, z_1, z_2) \\ \mathtt{F_2}(\mathtt{(}, \wedge, \mathtt{)}) & \leftarrow & . \\ \mathtt{F_2}(\mathtt{(}, \vee, \mathtt{)}) & \leftarrow & . \\ \mathtt{F_2}(\mathtt{(}, \rightarrow, \mathtt{)}) & \leftarrow & . \end{array}$$

In der Regel ist bei Sprachen die Menge der zulässigen Exponenten und deren Bedeutung bekannt. Füe einen Sprecher des Deutschen bedeutet dies, daß er jeden Satz daraufhin prüfen kann, ob er grammatisch ist, und wenn ja, was er bedeutet. Es ist dahingegen nicht klar, was denn die Typen, das heißt die grammatischen Kategorien unserer Sprache sind. Wir haben bisher gesagt, daß diese Teil der Sprache sind, insofern nicht frei wählbar. Dies birgt allerdings eine Gefahr, denn es ist am wenigsten klar, ob gewisse Ausdrücke nun einer verschiedenen Kategorie angehören oder nicht. Im schlimmsten Falle kann man schlicht behaupten, jedes Zeichen besitze einen anderen Typ (zum Beispiel seinen Strukturterm); dies führt dann natürlich unsere Definitionen ad absurdum. Deswegen verlangen wir, daß Konstituentenvorkommen nicht weiter differenziert werden dürfen als ihre Substitutionsklasse. Diese ist nur mit Rückgriff auf das ausgezeichnete Symbol S und die Zeichenkettensprache definiert. Wir geben ein Beispiel. Es seien $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Als Zeichenkettensprache haben wir lediglich $\{\mathtt{ab}, \mathtt{ac}\}$. Es bedeute `b` und `c` die Zahl 1, `ab` die Zahl 5, `ac` die Zahl 3. Es bestehe `ab` aus den Buchstaben `a` und `b` und `ac` aus den Buchstaben `a` und `c` (sodaß sie also keine Elementarzeichen sind). Welche Bedeutung kann man für `a` annehmen? (Dies ist ein Beispiel von Zadrozny [53].) Hier ist eine Lösung. Man gebe `a` den Typ A, `b` den Typ B und `c` den Typ C. Jetzt definiere man zwei binäre Modi. Der eine verbindet `a` und `b` und wirft 5 aus, der andere verbindet `a` und `c` und wirft 3 aus. Dies ist kompositional. Es ist aber nicht streng kompositional, denn die Substitutionsklassen von `b` und `c` sind gleich. Also müssen `b` und `c` den gleichen Typ haben. Es gibt tatsächlich keine streng kompositionale Grammatik für diese Sprache. Denn jeder Modus, der auf `a` und `b` anwendbar ist, ist auch auf `a` und `c` anwendbar: sie unterscheiden sich nicht im Typ und nicht in der Bedeutung. Ein Unterschied allein in dem Exponenten ist allerdings nicht einschlägig, weil die Exponenten nur durch echte Polynome manipuliert werden dürfen und daher auf die Form des Exponenten keine Rücksicht nehmen. Es folgt nun unmittelbar, daß `ab` und `ac` in einer streng kompositionalen Grammatik die gleichen Bedeutungen haben müssen. Es bleibt allerdings unbenommen, `ab` und `ac` als Exponenten gewisser 0–stelliger Modi zu sehen, womit sie keiner Analyse zugänglich sind. Dann fungieren sie aber genauso wir Idiome, wie etwa `nicht alle Tassen im Schrank haben`, welche eine Bedeutung haben, die eben nicht aus den Einzelbedeutungen durch reguläre Prozesse entsteht.

Nun jedoch wollen wir einige Beispiele diskutieren, damit klar wird, welche Sprachen jetzt kompositional sind und welche nicht. Wir beginnen mit einem negativen Beispiel, die Aussagenlogik. Eine Definition ist wie folgt. Das Alphabet besteht aus $V = \{\mathtt{p}_i : i \in \omega\}$ sowie den Symbolen $\top$, $\bot$, $\neg$, $\wedge$, und $\vee$ (dies ist nur eine der möglichen Mengen von elementaren Junktoren). Es sind $\mathtt{p}_i$, $\top$ und $\bot$ 0–stellig, $\neg$ einstellig und alle anderen Symbole zweistellig. Wir schreiben der Einfachheit halber in Polnischer Notation. Damit ist die Menge der Aussagen definiert. Syntaktische Typen haben wir nur einen, `S`. Alle Zeichen haben also den gleichen Typ. Nun zu der Bedeutung. Eine Funktion von V nach $\{0, 1\}$ heißt **Belegung**. Die Bedeutung

$[\vec{x}]$ einer Aussage $\vec{x}$ ist eine Funktion von Belegungen in die Menge $\{0,1\}$. Diese wird induktiv definiert. Sei dazu β eine Belegung.

* $[\mathtt{p}_i](\beta) = \beta(\mathtt{p}_i)$.
* $[\top](\beta) = 0$, $[\bot](\beta) = 1$.
* $[\neg\vec{x}](\beta) = 1 - [\vec{x}](\beta)$.
* $[\wedge\vec{x}\vec{y}](\beta) = min\{[\vec{x}](\beta), [\vec{y}](\beta)\}$.
* $[\vee\vec{x}\vec{y}](\beta) = max\{[\vec{x}](\beta), [\vec{y}](\beta)\}$.

Es ist klar, daß dies keinesfalls eine kompositionale Sprache ist. Der Schönheitsfehler dieser Definitionen ist, daß wir ein unendliches Alphabet haben. Wir haben schon früher darauf hingewiesen, daß man die Indizes als Binärfolgen kodieren kann (welche im Computer zum Beispiel den Adressen entsprechen), sodaß das Alphabet jetzt die Menge $\{\mathtt{p}, \mathtt{0}, \mathtt{1}, \top, \bot, \neg, \wedge, \vee\}$ ist. Dies rettet uns aber nicht, sofern wir darauf beharren, daß $\mathtt{p}\vec{x}$, mit $\vec{x}$ eine Binärfolge, nicht weiter analysierbar ist. Dies ist aber die gängige Praxis in der Logik. Man könnte, um alles zu retten, noch darauf verfallen, die Variablen $\mathtt{p}\vec{x}$ aus dem Buchstaben $\mathtt{p}$ durch sukzessives Anhängen von $\mathtt{0}$ und $\mathtt{1}$ zu erzeugen. Dadurch hangelt man sich von Variable zu Variable. Allerdings hat dies zwei Probleme: erstens, die angehängte $\mathtt{0}$ oder $\mathtt{1}$ ist damit zum eigenständigen Symbol geworden (denn synkategorematische Zeichen sind verboten), zweitens kann man den Wert von $[\mathtt{p}\vec{x}\mathtt{0}](\beta)$ bzw. den Wert von $[\mathtt{p}\vec{x}\mathtt{0}](\beta)$ nicht aus dem von $[\mathtt{p}\vec{x}](\beta)$ berechnen. Es gibt nur einen Ausweg: wir müssen einen eigenen syntaktischen Typ annehmen, den eines Registers, welcher gerade durch die Menge der Binärfolgen repräsentiert wird. Der Rest ist jetzt ganz kanonisch. Die Menge der Terme wird durch folgende kontextfreie Grammatik erzeugt.

$$\begin{array}{lcl}
\mathtt{S} & \to & \top \mid \bot \mid \mathtt{V} \mid \mathtt{F}_1\mathtt{S} \mid \mathtt{F}_2\mathtt{SS} \\
\mathtt{F}_1 & \to & \neg \\
\mathtt{F}_2 & \to & \wedge \mid \vee \\
\mathtt{V} & \to & \mathtt{PR} \\
\mathtt{P} & \to & \mathtt{p} \\
\mathtt{R} & \to & \mathtt{0} \mid \mathtt{1} \mid \mathtt{R0} \mid \mathtt{R1}
\end{array}$$

Daraus kann man jetzt leicht eine interpretierte lineare 1–LBG machen. Die letzte Regel baut eine Binärfolge auf; diese wird (zum Beispiel) durch sich selbst interpretiert. Belegungen sind jetzt Funktionen von nichtleeren Binärfolgen nach $\{0,1\}$. Wenn $\mathtt{p}$ mit einem Register verklebt wird, so wird in diesem Moment β auf das Register angewendet. Diese Beispiel ist sehr instruktiv; es zeigt uns, daß die Verwendung von Indizes eine sehr problematische Angelegenheit ist, und daß man besser daran tut, sie als offizielle Mitglieder der Sprache aufzunehmen.

Nun sind wir allerdings noch immer nicht am Ende unserer Schwierigkeiten. Wir haben als letztes noch nachzuweisen, daß unsere Funktionen alle berechenbar sind. Kritisch ist hier die Anwendung einer Belegung auf ein Register. Dies kann man nicht konkret berechnen, wenn eine Belegung eine beliebige Funktion von V nach $\{0,1\}$ ist. Es eröffnen sich mehrere Möglichkeiten.

1. Belegungen sind partielle Funktionen von V nach $\{0,1\}$ mit endlichem Definitionsbereich. Diese lassen sich als Listen ablegen und auf einfache Weise auf ein Register auswerten. Dann muß man allerdings geeignet mit der Partialität umgehen.

2. Belegungen sind berechenbare Funktionen von V nach $\{0,1\}$. Diese lassen sich zum Beispiel in Form ihres Turing–Kodes angeben. Nach Voraussetzung ist die Anwendung auf ein Register auch berechenbar.

3. Belegungen sind alle Funktionen von V nach $\{0,1\}$, welche für fast alle Elemente 0 sind oder für fast alle Elemente 1.

Die zweite Möglichkeit ist aus einem banalen Grund schlecht: es ist nicht entscheidbar, gegeben ein Turing–Kode, ob die Turing Maschine eine Funktion berechnet. Nimmt man M die Menge aller Turing–Kodes von Funktionen von V nach $\{1,0\}$, so ist das Problem gelöst. Allerdings ist dann M selbst nicht aufzählbar. (Man beachte außerdem noch, daß M Kodes m und n enthalten kann, welche dieselbe Funktion berechnen. Dies ist allerdings nicht störend, wie man sich leicht überlegt. Es ist wiederum unentscheidbar, ob zwei Maschinen dieselbe Funktion berechnen.)

Gehen wir nun zur Prädikatenlogik über (siehe Abschnitt 3.6). Wie im Falle der Aussagenlogik überlegt man sich, daß man anstelle der Indizes für Variablen nunmehr Registerfolgen nehmen muß. Wir werden auf dieses Detail jedoch jetzt nicht eingehen. Die Bedeutung einer Formel φ ist der Definition nach eine Funktion von Paaren $\langle \mathfrak{M}, \beta \rangle$ nach $\{0,1\}$, wo $\mathfrak{M}$ ein Modell ist und β eine Funktion von Variablen in den Individuenbereich des Modells. Wiederum stellt sich uns das Problem, eine endliche oder wenigstens berechenbare Vorschrift anzugeben. Wir zeigen zwei Wege auf, die zu grundsätzlich verschiedenen Ergebnissen führen. Der erste Versuch ist es, sich auf endliche Modelle zu beschränken. Dann ist also $\mathfrak{M}$, und inbsesondere der Individuenbereich D von $\mathfrak{M}$, endlich. Eine Belegung ist eine partielle Funktion von V nach D mit endlichem Definitionsbereich. Ein Term erhält als Bedeutung eine Funktion von Belegungen nach $D \cup \{\star\}$. (Ist nämlich x_α in t, und ist β auf x_α nicht definiert, so ist t^β undefiniert.) Eine Formel erhält als Bedeutung eine Funktion von Belegungen nach $\{0,1,\star\}$. Diese lassen sich anhand der Definitionen aus Abschnitt 3.6 induktiv definieren. D muß endlich sein, da wir den Wert von $\forall x_\alpha.\varphi(x_\alpha)$ in der Regel nicht bestimmen können, ohne alle Werte von x_α auszuprobieren.

Dieses Vorgehen hat einen schwerwiegenden Nachteil: es führt zu falschen Ergebnissen. Die Logik der endlichen Strukturen ist nämlich stärker als die Logik aller Strukturen. So ist die folgende Menge auf den endlichen Strukturen nicht erfüllbar, aber wohl auf unendlichen. (Hierbei ist 0 eine Konstante und $\mathtt{s}$ ein 1–stelliges Funktionssymbol.)

$$\{\forall \mathtt{x}_0.\neg(\mathtt{s}\mathtt{x}_0 \doteq \mathtt{0}), \forall \mathtt{x}_0.\forall \mathtt{x}_1.(\mathtt{s}\mathtt{x}_0 \doteq \mathtt{s}\mathtt{x}_1 \to \mathtt{x}_0 \doteq \mathtt{x}_1)\}$$

Denn sei D endlich. So ist für ein n und ein $k > 0$: $s^{n+k}0 = s^n 0$. Daraus folgt mit der zweiten Formel $s^k 0 = 0$. Da $k > 0$, so ist die erste Formel verletzt. Es gibt aber sicher ein unendliches Modell für diese Formeln. Dies stößt uns in ein Dilemma: offensichtlich kann man Bedeutungn nicht über die Strukturen direkt berechnen, sondern muß gewissermaßen abstrakte Mechanismen angeben. Ein solcher ist die deduktive Methode. Wir definieren $[\varphi] := \{\psi : \vdash \varphi \leftrightarrow \psi\}$ und legen fest, daß die Bedeutung von φ gerade $[\varphi]$ sei. Damit bedeutet eine Formel einfach die Menge aller zu ihr äquivalenten Formeln. Da wir diese aber auch nicht hinschreiben können — sie ist aufzählbar aber nicht entscheidbar —, ändern wir die Definition noch einmal. Wir sagen, φ bedeute jede Formel, welche zu ihr äquivalent ist. Genauso bezeichne ein Term jeden anderen Term, der zu ihm beweisbar gleich ist. Wir betrachten mithin folgende Zeichensprache:

$$\{\langle \varphi, F, \psi\rangle : \vdash \varphi \leftrightarrow \psi\} \cup \{\langle t, T, s\rangle : \vdash t \doteq s\}$$

Es gibt nur zwei Typen, den des Terms und den einer Formel. (Wie gesagt, ignorieren wir hier die Frage nach Registern.) Wie kann man diese Sprache erzeugen? Zunächst einmal genehmigen wir uns genug Modi, um die Sprache Δ zu erzeugen.

$$\Delta = \{\langle \varphi, F, \varphi\rangle : \varphi \ \mathit{Formel}\} \cup \{\langle t, T, t\rangle : t \ \mathit{Term}\}$$

Wir nehmen an, wir haben ein Symbol $\top$, welches immer wahr ist. Wir nehmen zunächst folgenden Modus $\mathtt{EQ}$ hinzu.

$$\begin{array}{lcl} \mathtt{EQ}(\langle \varphi, F, \psi\rangle, \langle \top, F, \psi \leftrightarrow \chi\rangle) & := & \langle \varphi, F, \chi\rangle \\ \mathtt{EQ}(\langle t, T, s\rangle, \langle \top, F, s \doteq u\rangle) & := & \langle t, T, u\rangle \end{array}$$

Dieser Modus erlaubt uns, eine neue Bedeutung für eine Formel oder einen Term anzunehmen, falls nur die neue mit der alten äquivalent ist. Alles, was wir jetzt noch tun müssen, ist, alle Bedeutung für $\top$ zu generieren. Dies ist nun aber genau das, was ein logischer Kalkül leistet. Ein Kalkül besteht aus einer Menge von Regeln. Diese haben die Form $\langle \Delta, \varphi\rangle$, wobei Δ eine endliche Menge von Formeln ist und φ eine einzige Formel. Zum Beispiel ist $\langle \{\mathtt{p}_0, \mathtt{p}_1\}, \mathtt{p}_0 \wedge \mathtt{p}_1\rangle$ eine aussagenlogische Regel. Eine **Instanz** von ρ entsteht durch Einsetzen von beliebigen Aussagen für die Aussageveriable (im Falle der Aussagenlogik), oder dem Einsetzen von beliebigen Formeln für Aussagevariable, und Termen für Objektvariable. Dabei bedeutet Einsetzen hier: Substitution im logischen Sinne. Eine **Ableitung** ist eine endliche Folge

$\langle \chi_i : i < n \rangle$, derart, daß für jedes $i < n$ eine Menge $\Delta_i \subseteq \{\chi_j : j < i\}$ gibt, sodaß $\langle \Delta_i, \chi_i \rangle$ Instanz einer Regel ist. Falls nun ein Kalkül durch endlich (oder aufzählbar viele) Regeln aufgeschrieben werden kann, so ist die Menge der ableitbaren Formeln ihrerseits aufzählbar. Für die Prädikatenlogik ist das der Fall, allerdings muß man dennoch wiederum Vorsicht walten lassen, da die Substitution ja keine Zeichenkettenersetzung ist. Was man benötigt, ist einen Kalkül mit endlich vielen Regeln, der nur auf Zeichenkettenersetzung beruht. Wenn dieser existiert, so ist die oben beschrieben Zeichensprache tatsächlich streng kompositional. Natürlich können die Polynome nicht allesamt inkrementell sein, sonst wäre eine Formel nicht unendlich oft ambig.

Kommen wir nun auf natürliche Sprachen zu sprechen. Im Arabischen besteht eine Wortwurzel üblicherweise aus drei Konsonanten. Ein sehr oft zitiertes Beispiel ist `ktb`, welches *schreiben* bedeutet, oder `klb`, welches *Hund* bedeutet. Es gibt auch vierkonsonantige Wurzeln, zum Beispiel `drhm` (von griechisch *Drachme*), welches eine Münzeinheit benennt. Von dieser Wurzel werden durch einfügen von Vokalen und Verdopplungen sogenannte *Binyanim*, zu Deutsch etwa Wortklassen gebildet. Hier sind ein paar davon. Zunächst bilden wir nur andere Verben.

(4.1)

I	`katab`	schreiben
II	`kattab`	schreiben machen
III	`kaatab`	korrespondieren
VI	`takaatab`	einander schreiben
VIII	`ktatab`	schreiben, eingeschrieben sein

Von diesen Verben kann man jetzt zum Beispiel verschiedene Zeiten bilden.

(4.2)

	Perf. Akt.	Perf. Pass.	Imp. Akt.	Imperf. Pass
I	`katab`	`kutib`	`aktub`	`uktab`
II	`kaatab`	`kuttib`	`ukattib`	`ukattab`
III	`kaatab`	`kuutib`	`ukaatib`	`ukaatab`
VI	`takaatab`	`tukuutib`	`atakattab`	`utakattab`
VIII	`ktatab`	`ktutib`	`aktatib`	`uktatab`

Wir haben hier nur die transparenten Beispiele gezeigt, es gibt andere Klassen, die nicht mit solch einfachen Mitteln gebildet werden, aber dies soll dennoch zur Illustration ausreichen. Es ist intuitiv klar, daß die sinntragende Einheit durch alle diese Worte hindurch — also das Wurzelmorphem — `ktb` ist. Doch dies tritt nie als Einheit auf, und die Konsonanten werden durch Vokale getrennt. Man überlege sich an dieser Stelle, daß es durchaus kompositionale Grammatiken des Arabischen (zumindest des Teils, der hier steht) geben kann, die kontextfrei sind. Allerdings müssen diese für einen einzelnen Konsonanten, zum Beispiel `b` eine Bedeutung postulieren, und gleichzeitig gestatten, daß die Bedeutung von `ktb` sich daraus berechnet. Ohne den Beweis hier anzutreten, merken wir nur an, daß dies nicht gehen kann, wenn man nicht annimmt, daß ein Buchstabe sich selbst bedeuten kann. Dies hieße allerdings, Form und Bedeutung in einen Topf zu werfen. Gehen wir also stattdessen

davon aus, daß ktb unsere Einheit ist. Wir nehmen also an, daß wir ein 0–stelligen Zeichen der folgenden Form haben.

$$\mathtt{W}_0 = \langle \mathtt{k} \otimes \mathtt{t} \otimes \mathtt{b}, V[\checkmark], \lambda x.\mathsf{schreiben}'(x)\rangle$$

Hierbei ist $V[\checkmark]$ der Typ der Verbalwurzeln. Die Binyanim werden durch Komposition mit abstrakten Morphemen gebildet, welche zunächst nur den Konsonantismus ändern.

$$\begin{array}{lcl}
\mathtt{B}_1 & := & \langle \lambda x \otimes y \otimes z.x \otimes y \otimes z, V[b]/V[\checkmark], \lambda\mathcal{P}.\mathcal{P}\rangle \\
\mathtt{B}_2 & := & \langle \lambda x \otimes y \otimes z.x \otimes yy \otimes z, V[b]/V[\checkmark], \lambda\mathcal{P}.\lambda x.\lambda y.\mathsf{machen}'(y, \mathcal{P}(x))\rangle \\
\mathtt{B}_8 & := & \langle \lambda x \otimes y \otimes z.xy \otimes y \otimes z, V[b]/V[\checkmark], \lambda\mathcal{P}.\mathcal{P}\rangle
\end{array}$$

$V[b]$ ist der Typ eines einmal abgeleiteten Verbums. II ist ein Kausativum, daher die angegebene Semantik. In Ermangelung besseren Wissens haben wir die Identität für VIII eingesetzt. Hiermit ist alles erklärt, denn es gibt nur einen mehrstelligen Modus, das zweistellige $\mathtt{C}_>$ beziehungsweise $\mathtt{C}_<$, welche hier natürlich zusammenfallen. Man rechnet nach, daß

$$\mathtt{C}_>\mathtt{B}_2\mathtt{W}_0 = \langle \mathtt{k} \otimes \mathtt{tt} \otimes \mathtt{b}, V[\mathrm{II}], \lambda x.\lambda y.\mathsf{machen}'(y, \mathsf{schreiben}'(x))\rangle$$

Man beachte, daß wir in der ersten Komponente, welche den Exponenten vorbehalten ist, nunmehr explizite Vektorpolynome hingeschrieben haben, was eine externe Spezifizierung unnötig macht. Dies haben wir schon am Ende von Abschnitt 4.3 angedeutet. Die Konvention, die wir hier anwenden, ist wiederum, daß der syntaktische Funktor auch phonologisch die Funktion ist. Doch davon gleich.

Die verschiedenen Zeiten und Genera Verbi dieser Klassen lassen sich ebenso bilden.

$$\begin{array}{lcllll}
\mathtt{P}_0 & := & \langle \lambda x \otimes y \otimes z.x\mathtt{a}y\mathtt{a}z, & V[\mathtt{PfAkt}]/V[b], & \lambda\mathcal{P}.\mathsf{Perf}'(\mathcal{P})\rangle \\
\mathtt{P}_1 & := & \langle \lambda x \otimes y \otimes z.x\mathtt{u}y\mathtt{i}z, & V[\mathtt{PfPass}]/V[b], & \lambda\mathcal{P}.\mathsf{Perf}'\mathtt{Pass}'(\mathcal{P})\rangle \\
\mathtt{P}_2 & := & \langle \lambda x \otimes y \otimes z.\mathtt{a}xy\mathtt{u}z, & V[\mathtt{PfAkt}]/V[b], & \lambda\mathcal{P}.\mathsf{Imp}'(\mathcal{P})\rangle \\
\mathtt{P}_3 & := & \langle \lambda x \otimes y \otimes z.\mathtt{u}xy\mathtt{a}z, & V[\mathtt{ImpPass}]/V[b], & \lambda\mathcal{P}.\mathsf{Imp}'\mathsf{Pass}'(\mathcal{P})\rangle
\end{array}$$

Wir explizieren hier nicht die Bedeutungen von Pass'. Sie interessieren uns nur peripher, lassen sich aber durchaus explizieren. Dies erklärt nun ganz hübsch, wie die Binyanim namens I, II und VIII zustandekommen. Wie aber sollen wir uns die Binyanim III und VI vorstellen? Offensichtlich sind die Binyanim III und VI in ihrem Vokalismus verschieden, nicht in ihrem Konsonantismus, aber wir haben noch gar keine Vokale! Auch hier schafft der λ–Kalkül Abhilfe. Wir definieren den Binyanim so, daß sie Funktionen werden, welche einen gegeben Vokalismus nehmen, und ihn verändern, bevor sie ihn mit der Wurzel vermischen.

$$\begin{array}{lcllll}
\mathtt{B}_1 & := & \langle \lambda x \otimes y \otimes z.\lambda u \otimes v.xuyvz, & V[b]/V[\checkmark], & \lambda\mathcal{P}.\mathcal{P}\rangle \\
\mathtt{B}_2 & := & \langle \lambda x \otimes y \otimes z.\lambda u \otimes v.xuyyvz, & V[b]/V[\checkmark], & \lambda\mathcal{P}.\lambda x.\lambda y.\mathsf{machen}'(y, \mathcal{P}(x))\rangle \\
\mathtt{B}_3 & := & \langle \lambda x \otimes y \otimes z.\lambda u \otimes v.xuuyvz, & V[b]/V[\checkmark], & \lambda\mathcal{P}.\mathsf{Freq}'(\mathcal{P})\rangle \\
\mathtt{B}_6 & := & \langle \lambda x \otimes y \otimes z.\lambda u \otimes v.xuyvvyvz, & V[b]/V[\checkmark], & \lambda\mathcal{P}.\mathsf{Rez}'(\mathcal{P})\rangle \\
\mathtt{B}_8 & := & \langle \lambda x \otimes y \otimes z.\lambda u \otimes v.yxuyvz, & V[b]/V[\checkmark], & \lambda\mathcal{P}.\mathcal{P}\rangle
\end{array}$$

Hierbei sind Freq' und Rez' Operatoren, welche das Frequentativum beziehungsweise das Reziproke Verb bilden. Es sind allerdings die Klassen nicht immer semantisch transparent, sodaß man — wie bei vielen Wortbildungsprozessen insgesamt — eigentlich gar nicht von einer regelmäßigen, das heißt kompositionalen Wortbildung sprechen kann, sondern eher von einem Instrument, mit dem sich die Sprache einen riesigen Bodensatz an eigenständigen Worten 'zimmert', die nur an der phonologischen Ebene regelmäßig gebildet sind. (Man denke nur an solche Verbalpräfixe wie `ver` oder `auf` im Deutschen.) Hier soll uns aber eher der phonologische Aspekt interessieren. Der Ansatz wie oben beschrieben, wird den Fakten leider nicht gerecht, insbesondere bekommen wir für das Imperfekt Aktiv sowie das Imperfekt Passiv inkorrekte Formen. Dies liegt natürlich auch daran, daß diese keinen einheitlichen Vokalismus haben. Mit einer Feinanalyse kann man dies natürlich alles in den Griff kriegen, und — das zumindest behaupten wir — den Regelmäßigkeiten in großem Maße Rechnung tragen. Bei komplizierten Binyanim, welche auch neue Konsonanten einstreuen, sowie vierkonsonantigen Wurzeln mag das schon wieder anders sein. Eine genaue Analyse steht noch aus.

Betrachten wir aber trotzdem das Erreichte. Wir haben das Alphabet nunmehr auf die Phoneme beziehungsweise einzelne Buchstaben festgelegt. Exponenten sind nun λ–Terme über der Algebra der Vektorpolynome. (Die Notwendigkeit von Vektorpolynomen mag sich zum Beispiel bei der Betrachtung des Arabischen erschließen. Denn nur so können wir unsere Wurzeln als Vektoren aufschreiben.) Üblicherweise ist der syntaktische Funktor auch der phonologische, aber dies ist nicht immer so. Die Verbalklassen sind Argumente des Operators Perfekt oder Aktiv, aber sie sind phonologisch Funktoren. Eine gezielte Typenanhebung des Perfekts/Aktivs kann das allerdings wieder korrigieren. Morphe sind also nicht mehr Zeichenketten, und auch nicht unbedingt Vektoren, sondern oft nur Anweisungen (als λ–Terme kodiert), wie ein Vektor zu behandeln ist. Morphe treten hier also auf einer ganz anderen Ebene in Erscheinung: sie sind, wenn man so will, die Grundzeichen, also das, was das Lexikon ausmacht. Falls man im Übrigen ein Morph so definiert, daß es alles ist, was nicht weiter kompositional analysierbar ist, so wird es schlicht synonym mit 0–stelligem Grundzeichen. Bei Ableitungsprozessen tritt, wie oben schon erwähnt, ein Morph nur über seinen Exponenten und seinen syntaktischen Typ aber ohne seine Bedeutung auf. Ein solcher formaler Gebrauch von Zeichen ist typisch für Sprachen, gehört aber — leider — nicht in den Rahmen dieser Ausführungen. Ein Morphem ist nun eine Abstraktion eines Morphs, welche man sich so vorstellen darf. Falls wir Morphe M_i, $i < n$, haben, welche gleichen Typ und gleiche Semantik haben, so dürfen wir sie als zu einem Morphem gehörig betrachten. Falls wir in unseren Polynomen auch auch disjunktive Spezifikation zulassen, lassen sich Morpheme auch als Einzelzeichen schreiben. Worte sind nun schlicht solche Zeichenketten, welche zwischen zwei Satzzeichen oder Leerzeichen auftreten. So lösen sich also die abstrakten Beschreibungsebenen in einem kohärenten System auf. Die übliche Sichtweise, daß wir es mit einer linearen Progression von Phonologie über Morphologie zur Syntax

und schließlich zur Semantik haben, verdeckt die Tatsache, daß erst auf der morphologischen Ebene Bedeutung entsteht, und die Semantik zunächst keine Syntax kennt. Wenn man alle drei Ebenen zusammendenkt, so sind die kleinsten Einheiten sogleich bedeutungstragend. Trotzdem gibt es so etwas wir eine phonologische Ebene, und diese gehorcht ihren eigenen Gesetzen genauso wie die Semantik. Dazwischen scheint es aber keine sinnvoll zu ziehenden Grenzen zu geben (Morphologie–Syntax). Dies geht zwar gegen eine allgemeine Lehrmeinung, aber die Lastenverteilung zwischen Morphologie und Syntax ist in verschiedenen Sprachen so ungleich und so unvorhersagbar, daß man das Wort besser als eine phonologische Kategorie betrachtet. Dies würde einiges transparenter machen. Zum Beispiel ist der Unterschied zwischen Kasus und Prä- oder Postositionen für die Syntax — nach allem, was man weiß — unerheblich. Wichtig ist allein die Tatsache der Selektion. Selegierte Präpositionalphrasen sind also syntaktisch genauso zu stellen wie kasusmarkierte Argumente, kasusmarkierte Adjunkte genauso wie Präpositionaladjunkte.

Es ist im Übrigen sicher kein Zufall, daß das Chinesische für jedes Wort ein Zeichen hat, welches keinen Aufschluß auf die Lautung gibt, und gleichzeitig Chinesisch keinerlei Morphologie besitzt. Hätte Chinesisch eine Morphologie gehabt wie etwa das Arabische, so wären die Zeichen sicher mit Hinweisen auf die Lautung versehen worden.

Wir haben früher schon erwähnt (Abschnitt 1.3), daß im Indonesischen der Plural durch Verdopplung gebildet wird. Im Chinesischen wird eine ja–nein–Frage auf folgende Weise gebildet. Ein einfacher Aussagesatz hat die Form (4.3) und seine Verneinung die Form (4.4). Chinesisch ist eine SVO–Sprache, und so folgt die Verbalphrase auf das Subjekt. Zwischen diese beiden schiebt sich das Wort `bu`. [2]

(4.3)	`Ta`	`zai`	`jia.`	
	Er/Sie/Es	zu	Hause	
(4.4)	`Ta`	`bu`	`zai`	`jia.`
	Er/Sie/Es	nicht	zu	Hause

Die ja–nein–Frage wird gebildet, indem nach dem Subjekt die unverneinte und die verneinte Verbalphrase hintereinandergeschrieben werden.

(4.5) `Ta zai jia bu zai jia?`
Ist er/sie/es zu Hause?

Wie Radzinski [42] zeigt, müssen die beiden Verbalphrasen exakt identisch sein. So ist zum Beispiel (4.6) grammatisch, (4.7) aber ungrammatisch. Allerdings ist (4.8) wiederum grammatisch und bedeutet so viel wie (4.6).

[2] Wir notieren keine Töne.

(4.6)	`Ni`	`xihuan`	`ta-de`	`chenshan`	`bu`	`xihuan`
	Du	mögen	sein	Hemd	nicht	mögen
	`ta-de`	`chenshan?`				
	sein	Hemd?				
	Magst Du sein Hemd?					
(4.7)	*`Ni`	`xihuan`	`ta-de`	`bu`	`xihuan`	`ta-de`
	Du	mögen	sein	nicht	mögen	sein
	`chenshan?`					
	Hemd?					
(4.8)	`Ni`	`xihuan`	`bu`	`xihuan`	`ta-de`	`chenshan?`
	Du	mögen	nicht	mögen	sein	Hemd?

Es gibt verschiedene Wege, um (4.8) zu erklären. Der eine ist, daß eine ja–nein–Frage auch durch Koordination von dem Verb und seiner Verneinung entstehen kann, oder aber, daß `xihuan` in Beispiel (4.8) tatsächlich eine Verbalphrase ist. Wir wollen dies nicht weiter vertiefen. Für unsere Zwecke soll es genügen, anzunehmen, daß man Fragen auf die Weise gewinnen kann, wie es (4.5) und (4.8) zeigen. Das Schema ist intuitiv einfach: man muß eine Konstituente nehmen, welche das Verb enthält aber nicht das Subjekt, und diese dann unverneint und verneint hinschreiben. Wie kann eine Regel aussehen, die dies beschreibt? In LBGs ist dies auf ganz einfache Weise möglich. Wir führen einen Typ `VP[b]` für bloße Verbalphrasen und einen Typ `VP[Q]` für ja–nein–Verbalphrasen ein. Die relevante Regel für (4.5) ist jetzt wie folgt.

$$\langle xyx, \texttt{VP[Q]}, \lambda x.? : X(x)\rangle \leftarrow \langle x, \texttt{VP[b]}, X\rangle \quad \langle y, \texttt{Neg}, Y\rangle$$

Es bedeute hierbei $? : \varphi$ die Frage, ob φ der Fall ist. Man beachte, daß die Semantik des Negationswortes irrelevant ist. Wir hätten es daher als synkategorematisches Symbol einführen können.

$$\langle x\ \texttt{bu}\ x, \texttt{VP[Q]}, \lambda x.? : X(x)\rangle \leftarrow \langle x, \texttt{VP[b]}, X\rangle$$

Allerdings hatten wir ausgeschlossen, daß kompositionale Grammatiken synkategorematische Symbole einführen. Falls man auch ja–nein–Fragen für transitive Verben nach dem Muster (4.8) formen möchte, muß man auch die folgende Regel hinzunehmen.

$$\langle xyx, \texttt{Vt[Q]}, \lambda x.\lambda y.? : X(y)(x)\rangle \leftarrow \langle x, \texttt{Vt[b]}, X\rangle \quad \langle y, \texttt{Neg}, Y\rangle$$

Hierbei ist `Vt[b]` der Typ eines transitiven bloßen Verbs, `Vt[Q]` der Typ eines transitiven Verbs in einer ja–nein–Frage. In Radzinski [42] wird das Phänomen der Verdopplung benutzt, um zu zeigen, daß Mandarin Chinesisch in einem gewissen Sinne nicht kontextfrei ist. Das Argument ist diffizil, weil man nicht zeigen kann, daß Mandarin nicht kontextfrei ist. Das Problem ist, daß Ketten der Form $\vec{x}\ \texttt{bu}\ \vec{y}$ wahlweise als Konstituenten eines Aussagesatzes verstanden werden können. Nur wenn $\vec{y} = \vec{x}$ ist, können sie als Konstituenten eines Fragesatzes verwendet werden. Das bedeutet also, daß Chinesisch zwar schwach kontextfrei ist, aber als Zeichensystem nicht

kontextfrei. Dies muß man allerdings streng beweisen. Wir nehmen dazu an, daß Mandarin durch eine interpretierte lineare 1–LBG erzeugt werden kann in der Weise, daß das Zeichensystem streng kompositional ist. Betrachten wir das Vorkommen eines Verbs in einer ja–nein–Frage wie (4.8). Wir betrachten den Strukturterm als Funktion von dem Zeichen für das transitive Verb `xihuan`, welches wir hier mit `x` bezeichnen.

$$t = t(\mathtt{x})$$

Wir erhalten einen anderen Strukturterm, wenn wir ein anderes Zeichen `y` einsetzen. Die Frage ist nun: wann ist dieser neue Strukturterm definit? Offensichtlich ist er stets orthographisch definit. Wie steht es mit der syntaktischen Definitheit? Wir dürfen das transitive Verb durch ein anderes ersetzen, ohne die Grammatikalität zu verletzen. (Wir verlieren dabei lediglich die Interpretation als ja–nein–Frage.) Also ist das Zeichen $t(\mathtt{y})$ auch syntaktisch definit. Es ist aber auch semantisch definit, denn eine Interpretation existiert. Alles in allem darf man also für `xihuan` jedes transitive Verb einsetzen. Die Interpretation von $t(\mathtt{y})$ ist aber nur dann eine ja–nein–Frage, wenn `y` denselben Exponenten hat wie `x`. (Radzinski zeigt explizit, daß die Interpretation als ja–nein–Frage von der Interpretation unabhängig ist.) Aber die Interpretationen hängen nicht davon ab, welchen Exponenten ein Zeichen hat. Die syntaktischen Typen auch nicht. Deswegen erhält man also entweder sowohl für $t(\mathtt{y})$ als auch für $t(\mathtt{x})$ eine ja–nein–Frage oder für keinen von beiden. Das geht jedoch nicht. Das ist der gewünschte Widerspruch. Es erscheint intuitiv plausibel, daß man auch keine interpretierte lineare k–LBG für Mandarin schreiben kann. Dies ist aller Wahrscheinlichkeit nach falsch. Denn einmaliges Kopieren (und darum handelt es sich hier) kann man mit einer linearen 2–LBG durchaus modellieren.

Übung 112. Beweisen Sie Lemma 4.7.6.

Kapitel 5

Linguistische Strukturen

5.1 Kategorien

Bisher haben wir in unserer Beschreibung von syntaktischen Strukturen lediglich zu Nichtterminalsymbolen gegriffen, welche keinerlei innere Struktur haben. Dies ist zwar oft keine wesentliche Einschränkung, führt aber dazu, daß wesentliche Regularitäten nicht repräsentiert werden können. Dazu ein Beispiel. Das finite Verb kongruiert im Deutschen und in vielen anderen Sprachen mit dem Subjekt und zwar im Deutschen in Person und Numerus. Dies bedeutet, daß das Verb veschiedene Formen hat, je nachdem, ob das Subjekt in der 1., in der 2. oder der 3. Person steht, und je nachdem, ob es im Singular oder im Plural steht. So sind die folgenden Sätze grammatisch:

(5.1) `Ich sehe.`
(5.2) `Du siehst.`
(5.3) `Er/Sie/Es sieht.`
(5.4) `Wir sehen.`
(5.5) `Ihr seht.`
(5.6) `Sie sehen.`

Aber die folgenden sind ungrammatisch.

(5.7) *`Ich siehst/sieht/sehen/seht.`
(5.8) *`Du sehe/sieht/sehen/seht.`

Wie kann man dem Rechnung tragen? Wir können schlicht 6 verschiedene Subjekte unterscheiden (1/2/3 Person, Singular/Plural) sowie 6 verschiedene Verben (von denen zwei homophon sind, nämlich diejenigen für die 1. und die 3. Person Plural). Ein anderer Weg wurde in der sogenannten **Generalisierten Phrasenstrukturgrammatik** (**GPSG**, siehe [15]) vorgeschlagen. Wir gehen von der folgenden uniformen

Regel aus.

$$(*) \qquad \texttt{S} \rightarrow \texttt{NP} \quad \texttt{VP}$$

Hierbei sind nun aber die Symbole `S`, `NP` und `VP` nicht die Symbole für einzelne Kategorien sondern nur Eigenschaften von Kategorien. Dies bedeutet, daß sie mit logischen Junktoren wie Negation und Konjunktion verknüpft werden können. Führen wir zum Beispiel die Eigenschaften `1`, `2` und `3` sowie `Sg` und `Pl` ein, dann können wir unsere Regel $(*)$ weiter verfeinern:

$$\texttt{S} \rightarrow \texttt{NP} \wedge \texttt{1} \wedge \texttt{Sg} \quad \texttt{VP} \wedge \texttt{1} \wedge \texttt{Sg}$$

Ferner haben wir diese terminale Regeln.

$$\texttt{NP} \wedge \texttt{1} \wedge \texttt{Sg} \rightarrow \texttt{ich}, \qquad \texttt{VP} \wedge \texttt{1} \wedge \texttt{Sg} \rightarrow \texttt{sehe}$$

Hierbei ist also $\texttt{NP} \wedge \texttt{1} \wedge \texttt{Sg}$ eine Beschreibung einer Kategorie, welche verlangt, daß diese eine Nominalphrase (`NP`) in der ersten Person (`1`) Singular (`Sg`) ist. Dies bedeutet nun, daß zum Beispiel der Satz (5.1) abgeleitet werden kann. Damit allerdings die Sätze aus (5.7) nicht ableitbar sind, müssen wir die Regel $(*)$ nunmehr entfernen. Um (5.2) – (5.6) zu bekommen, müssen wir allerdings noch fünf weitere Regeln einführen. Diese lassen sich zu einer schematischen Regel zusammenfassen. Dazu schreiben wir anstelle von `NP` nunmehr $[\textsc{kat} : np]$, statt `1` $[\textsc{pers} : 1]$, und anstelle von `Pl` schreiben wir $[\textsc{num} : pl]$. Dabei heißen KAT, PER und NUM **Attribute**, und np, vp, 1, etc. heißen **Werte**. Im Zusammenhang $[\textsc{kap} : np]$ sagen wir, das Attribut KAP habe den Wert np. Eine Menge von Paaren $[A : w]$, wo A ein Attribut und w ein Wert ist, heißt auch **Attribut–Wert Struktur**.

Die Regel $(*)$ wird jetzt durch die schematische Regel $(\ddagger)$ ersetzt.

$$(\ddagger) \qquad \left[\begin{array}{lcl} \textsc{kat} & : & s \end{array}\right] \rightarrow \left[\begin{array}{lcl} \textsc{kat} & : & np \\ \textsc{per} & : & \alpha \\ \textsc{num} & : & \beta \end{array}\right] \qquad \left[\begin{array}{lcl} \textsc{kat} & : & vp \\ \textsc{per} & : & \alpha \\ \textsc{num} & : & \beta \end{array}\right]$$

Dabei sind α und β Variable. Allerdings nehmen sie verschiedene Werte an; α kann alle Werte aus $\{1, 2, 3\}$ annehmen, β alle Werte aus $\{sg, pl\}$. Dies wird weiter unter noch problematisiert werden. Man muß nun als Erstes beachten, daß die kongruenzanzeigenden Eigenschaften weitergereicht werden. Dies führt dann dazu, daß etwa die Regel

$$(\dagger) \qquad \texttt{VP} \rightarrow \texttt{V} \quad \texttt{NP}\,,$$

welche eine Verbalphrase in ein (transitives) Verb plus ein direktes Objekt analysiert, nunmehr analog verfeinert werden muß.

$$\left[\begin{array}{lcl} \textsc{kat} & : & vp \\ \textsc{per} & : & \alpha \\ \textsc{num} & : & \beta \end{array}\right] \rightarrow \left[\begin{array}{lcl} \textsc{kat} & : & v \\ \textsc{per} & : & \alpha \\ \textsc{num} & : & \beta \end{array}\right] \qquad \left[\begin{array}{lcl} \textsc{kat} & : & np \end{array}\right]$$

Es gibt nun Sprachen, in denen ein Verb nicht nur mit dem Subjekt sondern auch mit dem Objekt kongruiert. Dies bedeutet, daß es nicht ausreicht, einfach nur [PER : α] zu schreiben; sondern wir müssen sagen, ob α die Subjektsperson oder die Objektsperson benennt. Also bettet man die Struktur noch weiter ein und schreibt jetzt so.

$$\begin{bmatrix} \text{KAT} & : & vp \\ \text{PER} & : & \alpha \\ \text{NUM} & : & \beta \end{bmatrix} \rightarrow \begin{bmatrix} \text{KAT} & : & v \\ \text{AGRS} & : & \begin{bmatrix} \text{PER} & : & \alpha \\ \text{NUM} & : & \beta \end{bmatrix} \\ & : & \\ \text{AGRO} & : & \begin{bmatrix} \text{PER} & : & \alpha' \\ \text{NUM} & : & \beta' \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \text{KAT} & : & np \\ \text{PER} & : & \alpha' \\ \text{NUM} & : & \beta' \end{bmatrix}$$

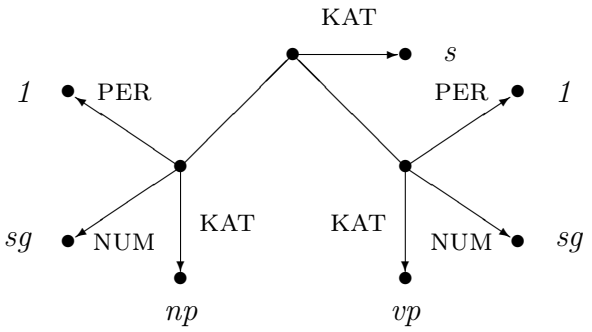
Es ist klar, daß diese Regel das Gewünschte leistet. Man kann sie noch weiter verschönern, indem man zum Beispiel auch bei der Nominalphrase die Numerus und Personmerkmale in AGR einbettet; dies wird weiter unten aber noch vorgeführt. Man sieht nun, daß der Wert eines Attributs nicht allein ein einzelner Wert sein muß, sondern wiederum eine Attribut–Wert Struktur sein kann. Man spricht dann von *1*, *sg* als **atomaren Werten**. Attribute, welche nur atomare Werte haben, heißen vom **Typ 0**, alle anderen vom Typ 1. So ist dies in [14] schon vorgesehen worden. In der sogenannten **Head Driven Phrase–Structure Grammar** (**HPSG**, siehe [40]) wurde dies allerdings noch weiter getrieben und elaborierte Strukturen definiert, in denen sämtliche Information kodiert ist. Diese Strukturen werden wir in diesem Abschnitt vom theoretischen Blickwinkel her studieren, schon weil sie in vielen anderen, wie HPSG vorwiegend computerorientierten Grammatikformalismen auch benutzt werden. Bevor wir dies tun wollen, werden wir noch einen Schritt weiter gehen. Die Regeln, welche wir oben eingeführt haben, benutzen Variable für Werte von Attributen. Dies ist sicher ein gangbarer Weg; allerdings hat man in HPSG eine andere Richtung eingeschlagen. Es werden sogenanne *Strukturvariable* eingeführt, welche dafür sorgen sollen, daß gewisse Werte gleich sind, weil die betreffende Struktur 'geteilt' wird. Dazu führen wir unser Beispiel weiter. Eine Nominalphrase beschreiben wir nun nicht einfach durch eine flache Attribut–Wert Struktur, sondern wir fassen auch hier die Kongruenzmerkmale in einer eigenen Teilstruktur zusammen. Wir repräsentieren eine NP im Plural und der 3. Person jetzt so.

$$\begin{bmatrix} \text{KAT} & : & np \\ \text{AGR} & : & \begin{bmatrix} \text{NUM} & : & pl \\ \text{PER} & : & 3 \end{bmatrix} \end{bmatrix}$$

Der Wert des Attributs AGR ist jetzt genauso strukturiert wie die Werte von AGRS und AGRO. Jetzt können wir unsere Regeln mit Hilfe von Strukturmarken wie folgt schreiben. Die Regel (‡) sieht so aus.

$$(\ddagger) \qquad \left[\text{KAT} : s \right] \rightarrow \begin{bmatrix} \text{KAT} & : & np \\ \text{AGR} & : & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \text{KAT} & : & vp \\ \text{AGRS} & : & \boxed{1} \end{bmatrix}$$

Abbildung 5.1: Eine syntaktische Struktur



Aus der Regel, welche das Objekt einführt, wird jetzt diese Regel:

$$\begin{bmatrix} \text{KAT} & : & vp \\ \text{AGRS} & : & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{KAT} & : & v \\ \text{AGRS} & : & \boxed{1} \\ \text{AGRO} & : & \boxed{2} \end{bmatrix} \qquad \begin{bmatrix} \text{KAT} & : & np \\ \text{AGR} & : & \boxed{2} \end{bmatrix}$$

Die Marken $\boxed{1}$ und $\boxed{2}$ sind hier Variable für Attribut–Wert Strukturen. Tritt eine Marke mehrmals auf, so muß für sie jedesmal die gleiche Attribut–Wert Struktur ersetzt werden. Auf diese Weise wird Kongruenz ganz automatisch hergestellt. Dies ist ein sehr eleganter Weg, um zu vermeiden, daß Kongruenz den Regelapparat aufbläht. Die Regeln sind durch die Einführung von Strukturvariablen wieder klein und handlich geworden. Allerdings taugt diese Methode nur beschränkt für die Beschreibung von Kongruenz.

Wenn nun die Attribut–Wert Strukturen lediglich Beschreibungen sind, was sind dann die Kategorien? Dazu wird vereinbahrt, daß jedem Attribut schlicht eine 2–stellige Relation und jedem atomaren Wert eine Eigenschaft von Elementen entspricht. Eine syntaktische Struktur ist dann nicht mehr nur ein erschöpfend geordneter Baum, sondern ein erschöpfend geordneter Baum mit zusätzlichen 2–stelligen Relationen, jede entsprechend einem Attribut. Die Figur 5.1 zeigt das Beispiel einer Struktur, die — wie man sagt — von der Regel (‡) lizensiert wird. Die Literatur über Attribut–Wert Strukturen ist reich (siehe [27], [6]). In ihren Grundzügen sind sie jedoch recht einfach. Zunächst einmal ist es nicht notwendig, die Werte von Attributen als Objekte aufzufassen. Dies ist insbesondere deshalb nicht vorteilhaft, weil die Attribut–Wert Strukturen selbst keine Objekte sind, sondern nur Beschreibungen von Objekten, aber ihrerseits als Werte von Attributen auftreten können. Deshalb behandeln wir Werte wie *np*, *1* nunmehr auch als Eigenschaften, welche man mit den üblichen booleschen Operationen, also $\neg$, $\wedge$, $\vee$ oder $\rightarrow$, miteinander verbinden kann. Das hat zum Beispiel den Vorteil, daß wir die Verbform `sehen` auf

eine der folgenden Weisen repräsentieren können.

$$\begin{bmatrix} \text{KAT} & : & v \\ \text{PER} & : & 1 \vee 3 \\ \text{NUM} & : & pl \end{bmatrix} \qquad \begin{bmatrix} \text{KAT} & : & v \\ \text{PER} & : & \neg 2 \\ \text{NUM} & : & pl \end{bmatrix}$$

Man beachte im Übrigen, daß die Zusammenfassung in eine Attribut–Wert Struktur nichts anderes als die Konjunktion repräsentiert. So kann man die linke AWS auch folgendermaßen schreiben.

$$[\text{KAT} : v] \wedge [\text{PER} : 1 \vee 3] \wedge [\text{NUM} : pl]$$

Man nennt im Übrigen die Tatsache, daß eine Repräsentation nicht alle Möglichkeiten ausschließt, auch **Unterspezifizierung**. Dabei ist die explizite Disjunktion nicht der genuine Fall; man meint eher damit die Tatsache, daß gewisse Eigenschaften gar nicht hingeschrieben werden müssen. So kann man das Englische `saw` wie folgt repräsentieren.

$$\begin{bmatrix} \text{KAT} & : & v \\ \text{TEMP} & : & verg \end{bmatrix}$$

Dies besagt, daß wir eine Verb in der Vergangenheit haben. Der Numerus und die Person ist völlig unerwähnt geblieben. Wir können — müssen aber nicht — diese Merkmale explizit hinschreiben.

$$\begin{bmatrix} \text{KAT} & : & v \\ \text{TEMP} & : & verg \\ \text{NUM} & : & \top \\ \text{PER} & : & \top \end{bmatrix}$$

Hierbei ist $\top$ das maximal unspezifische Merkmal. Es gilt — dies ist ein linguistisches Faktum —:

$$[\text{PER} : 1 \vee 2 \vee 3]$$

Daraus können wir ableiten, daß `saw` auch die folgende Repäsentation haben kann.

$$\begin{bmatrix} \text{KAT} & : & v \\ \text{TEMP} & : & verg \\ \text{NUM} & : & \top \\ \text{PER} & : & 1 \vee 2 \vee 3 \end{bmatrix}$$

Dieses sogenannte Faktum ist ein Axiom, welches wir explizit fordern müssen. Doch davon später.

Da nun die Attribut–Wert Paare ebenfalls Aussagen sind, so können wir auch sie in der gleichen Weise verbinden. So hätte dann das Wort `see` im Englischen folgende grammatische Repräsentation.

$$\neg \begin{bmatrix} \text{KAT} & : & v \\ \text{PER} & : & 3 \\ \text{NUM} & : & sg \end{bmatrix} \vee \begin{bmatrix} \text{KAT} & : & v \\ \text{NUM} & : & pl \end{bmatrix}$$

Dies kann man auch so schreiben.

$$[\textsc{kat} : v] \wedge (\neg([\textsc{per} : 3] \wedge [\textsc{num} : sg]) \vee [\textsc{num} : pl])$$

Dies läßt sich zu Folgendem vereinfachen.

$$[\textsc{kat} : v] \wedge (\neg[\textsc{per} : 3] \vee [\textsc{num} : pl])$$

Grundlage für diese Umformungen sind folgende allgemeinen Rechengesetze. Wir nennen eine AVS **allgemeingültig**, falls sie stets wahr ist, das heißt, auf jedes Objekt zutrifft.

- Ist φ eine Tautologie der Ausagenlogik, so gilt φ unter jeder Einsetzung von Attribut–Wert Strukturen oder Variablen für AWSn für Aussagevariable.
- Ist φ eine allgemeingültige AVS, so auch $[\mathrm{X} : \varphi]$.
- $[\mathrm{X} : \varphi \to \chi]. \to .[\mathrm{X} : \varphi] \to [\mathrm{X} : \chi]$ ist allgemeingültig.
- Ist φ allgemeingültig und $\varphi \to \chi$, so auch χ.

Die meisten Attribute sind definit, sie haben nur einen Wert. Für sie gilt zusätzlich

$$[\mathrm{X} : \varphi] \wedge [\mathrm{X} : \chi]. \to .[\mathrm{X} : \varphi \wedge \chi]$$

Gelegentlich werden auch sogenannte **mengenwertige** Attribute postuliert, für die dies nicht gilt.

In der Literatur sind zwei verschiedene logische Sprachen zur formalen Behandlung dieser Strukturen vorgeschlagen worden. Die erste ist die sogenannte monadische Prädikatenlogik 2. Stufe (MSO). Diese geht über die Prädikatenlogik, die uns schon des öfteren begegnet ist, hinaus. Sie besitzt Variable für einstellige Prädikate, und dafür auch Quantoren, welche $\forall$ und $\exists$. Es ist also $P := \{P_i : i \in \omega\}$ eine Menge von einstelligen Prädikatvariablen zusätzlich zu der Menge $V := \{x_i : i \in \omega\}$ der Objektvariablen. Wir schreiben also $P_i(x)$. Ist dann φ eine Formel, so auch $(\forall P_i)\varphi$ und $(\exists P_i)\varphi$. Die Strukturen sind die gleichen wie in der Prädikatenlogik (siehe Abschnitt 3.6). Sie sind also Tripel $\mathfrak{M} = \langle M, \{f^{\mathfrak{M}} : f \in F\}, \{r^{\mathfrak{M}} : r \in R\}\rangle$, wo M eine nichtleere Menge ist, $f^{\mathfrak{M}}$ die Interpretation der Funktion f in $\mathfrak{H}$ ist und $r^{\mathfrak{m}}$ die Interpretation der Relation r. Ein **Modell** ist ein Tripel $\langle \mathfrak{M}, \gamma, \beta\rangle$, wo $\mathfrak{M}$ eine Struktur ist, $\beta : V \to M$ eine Funktion, welche jeder Variablen ein Element aus M zuordnet, und $\gamma : P \to \wp(M)$ eine Funktion, welche jedem Prädikat aus P eine Teilmenge von M zuordnet. Die Beziehung $\langle \mathfrak{M}, \gamma, \beta\rangle \models \varphi$ wird induktiv definiert. Wir haben

$$\langle \mathfrak{M}, \gamma, \beta\rangle \models P_i(x_j) \qquad \Leftrightarrow \qquad \beta(x_j) \in \gamma(P_i)$$

Wir definieren $\gamma \sim_P \gamma'$ falls $\gamma'(Q) = \gamma(Q)$ für alle $Q \neq P$. Jetzt ist

$$\begin{array}{lll} \langle \mathfrak{M}, \gamma, \beta \rangle \models (\forall P)\varphi & \Leftrightarrow & \text{für alle } \gamma' \sim_P \gamma : \langle \mathfrak{M}, \gamma', \beta \rangle \models \varphi \\ \langle \mathfrak{M}, \gamma, \beta \rangle \models (\exists P)\varphi & \Leftrightarrow & \text{für ein } \gamma' \sim_P \gamma : \langle \mathfrak{M}, \gamma', \beta \rangle \models \varphi \end{array}$$

Es ist $\mathfrak{M} \models \varphi$, falls für alle γ und β gilt $\langle \mathfrak{M}, \gamma, \beta \rangle \models \varphi$.

Andere Typen von Sprachen, die vorgeschlagen worden ist, sind Modale Sprachen. (Siehe [3] and [32].) Wir wollen eine spezielle herausgreifen, die hier von Bedeutung ist, nämlich die quantifizierte Modallogik (QML). Diese besitzt einer abzählbaren Menge $PV := \{p_i : i \in \omega\}$ von Aussagevariablen, einer Menge M von sogenannten Modalitäten, C eine Menge von Konstanten. Schließlich gibt es neben den Klammern noch die Symbole $\neg$, $\wedge$, $\vee$, $\rightarrow$, $[-]$, $\langle - \rangle$, $\forall$ und $\exists$. Formeln werden induktiv wie folgt definiert.

1. p_i, $i \in \omega$, und c, $c \in C$, sind Aussagen.
2. Ist φ eine Aussage, so auch $\neg\varphi$, $(\forall P_i)\varphi$ und $(\exists P_i)\varphi$.
3. Ist φ eine Aussage und $m \in M$ eine Modalität, so sind $[m]\varphi$ und $\langle m \rangle \varphi$ Aussagen.
4. Sind φ_1 und φ_2 Aussagen, so auch $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ sowie $\varphi_1 \rightarrow \varphi_2$.

Wir sagen, $[m]$ und $\langle m \rangle$ sind **Modaloperatoren**.

Definition 5.1.1 *Eine **Kripke–Struktur** ist ein Tripel $\mathfrak{F} = \langle F, R, K \rangle$, wo F eine nichtleere Menge ist, $R : M \rightarrow \wp(F \times F)$ eine Funktion, welche jedem Modaloperator eine zweistellige Relation auf F zuordnet und $K : C \rightarrow \wp(F)$ eine Funktion, welche jeder Konstanten eine Teilmenge von F zuordnet. Ein **Kripke–Modell** auf $\mathfrak{F}$ ist ein Paar $\langle \mathfrak{F}, \beta \rangle$, wo $\mathfrak{F}$ eine Kripke–Struktur ist und $\beta : PV \rightarrow \wp(F)$.*

Es sei $x \in F$. Dann definieren wir

$$\begin{array}{lll} \langle \mathfrak{F}, \beta, x \rangle \models p_i & \Leftrightarrow & x \in \beta(p_i) \\ \langle \mathfrak{F}, \beta, x \rangle \models c & \Leftrightarrow & x \in K(c) \\ \langle \mathfrak{F}, \beta, x \rangle \models \neg\varphi & \Leftrightarrow & \langle \mathfrak{F}, \beta, x \rangle \not\models \varphi \\ \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \text{ und } \langle \mathfrak{F}, \beta, x \rangle \models \varphi_2 \\ \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \text{ oder } \langle \mathfrak{F}, \beta, x \rangle \models \varphi_2 \\ \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \rightarrow \varphi_2 & \Leftrightarrow & \text{wenn } \langle \mathfrak{F}, \beta, x \rangle \models \varphi_1 \text{ dann } \langle \mathfrak{F}, \beta, x \rangle \models \varphi_2 \\ \langle \mathfrak{F}, \beta, x \rangle \models [m]\varphi & \Leftrightarrow & \text{für alle } y \text{ mit } x\, R(m)\, y : \langle \mathfrak{F}, \beta, y \rangle \models \varphi \\ \langle \mathfrak{F}, \beta, x \rangle \models \langle m \rangle \varphi & \Leftrightarrow & \text{für ein } y \text{ mit } x\, R(m)\, y : \langle \mathfrak{F}, \beta, y \rangle \models \varphi \\ \langle \mathfrak{F}, \beta, x \rangle \models (\forall p)\varphi & \Leftrightarrow & \text{für alle } \beta' \text{ mit } \beta' \sim_p \beta : \langle \mathfrak{F}, \beta', x \rangle \models \varphi \\ \langle \mathfrak{F}, \beta, x \rangle \models (\exists p)\varphi & \Leftrightarrow & \text{für ein } \beta' \text{ mit } \beta' \sim_p \beta : \langle \mathfrak{F}, \beta', x \rangle \models \varphi \end{array}$$

Es ist $\langle \mathfrak{F}, \beta \rangle \models \varphi$, falls für alle $x \in F$ gilt $\langle \mathfrak{F}, \beta, x \rangle \models \varphi$; sowie $\mathfrak{F} \models \varphi$, falls für alle β gilt $\langle \mathfrak{F}, \beta \rangle \models \varphi$.

Wir definieren eine Einbettung von **QML** in **MSO** wie folgt. Es sei $R := \{r^m : m \in M\}$ und $C := \{Q^c : c \in K\}$. Dann definiere $\varphi^\dagger$ induktiv über den Aufbau.

$$\begin{array}{lcl}
p_i^\dagger & := & P_i(x_0) \\
c^\dagger & := & Q^c(x_0) \\
(\neg\varphi)^\dagger & := & \neg\varphi^\dagger \\
(\varphi_1 \wedge \varphi_2)^\dagger & := & \varphi_1^\dagger \wedge \varphi_2^\dagger \\
(\varphi_1 \vee \varphi_2)^\dagger & := & \varphi_1^\dagger \vee \varphi_2^\dagger \\
(\varphi_1 \to \varphi_2) & := & \varphi_1^\dagger \to \varphi_2^\dagger \\
((\forall p_i)\varphi)^\dagger & := & (\forall P_i)\varphi^\dagger \\
((\exists p_i)\varphi)^\dagger & := & (\exists P_i)\varphi^\dagger \\
([m]\varphi)^\dagger & := & (\forall x_0)(r^m(x_0, x_i) \to [x_i/x_0]\varphi^\dagger) \\
(\langle m \rangle \varphi)^\dagger & := & (\exists x_0)(r^m(x_0, x_i) \wedge [x_i/x_0]\varphi^\dagger)
\end{array}$$

Hier ist x_i in den letzten zwei Kauseln eine Variable, die nicht bereits in $\varphi^\dagger$ auftritt. Es gilt nun der

Theorem 5.1.2 *Es sei φ eine Formel der quantifizierten Modallogik. Dann ist $\varphi^\dagger$ eine Formel der monadischen Prädikatenlogik 2. Stufe über der entsprechenden Signatur, und es gilt für jede Kripke–Struktur $\mathfrak{F}$: $\mathfrak{F} \models \varphi$ genau dann, wenn $\mathfrak{F} \models \varphi^\dagger$.*

Beweis. Wir zeigen genauer den folgenden Sachverhalt: ist $\beta : PV \to \wp(F)$ eine Belegung in $\mathfrak{F}$, und $x \in F$, sowie $\gamma : P \to \wp(F)$ und $\delta : V \to F$ Belegungen für die Prädikat- und Objektvariablen. Ist dann $\gamma(P_i) = \beta(p_i)$ für alle $i \in \omega$ und $\delta(x_0) = w$, so gilt

$$(\ddagger) \qquad \langle \mathfrak{F}, \beta, w \rangle \models \varphi \qquad \Leftrightarrow \qquad \langle \mathfrak{F}, \gamma, \delta \rangle \models \varphi^\dagger$$

Die behauptete Tatsache folgt dann so. Ist β, w gegeben, mit $\langle \mathfrak{F}, \beta, w \rangle \not\models \varphi$, so lassen sich γ und δ nicht (notwendig eindeutig) angeben mit $\langle \mathfrak{F}, \gamma, \delta \rangle \not\models \varphi^\dagger$; und sind γ und δ gegeben mit $\langle \mathfrak{F}, \gamma, \delta \rangle \not\models \varphi^\dagger$, so lassen sich β und x angeben mit $\langle \mathfrak{F}, \beta, x \rangle \not\models \varphi$. Nun aber zu dem Beweis von (‡). Der Beweis wird induktiv geführt. Ist $\varphi = p_i$, so ist $\varphi^\dagger = P_i(x_0)$ und die Behauptung gilt aufgrund der Tatsache, daß $\beta(p_i) = \gamma(P_i)$ und $\gamma(x_0) = x$ ist. Ebenso für $\varphi = c \in C$. Die Schritte für $\neg$, $\wedge$, $\vee$ und $\to$ sind Routineangelegenheiten. Betrachten wir also $\varphi = (\exists p_i)\eta$. Sei $\langle \mathfrak{F}, \beta, w \rangle \models \varphi$. Dann gilt für ein β', das sich höchstens in p_i von β unterscheidet: $\langle \mathfrak{F}, \beta', w \rangle \models \eta$. Setze γ' wie folgt: $\gamma'(P_i) := \beta'(p_i)$, für alle $i \in \omega$. Nach Induktionsannahme ist dann $\langle \mathfrak{F}, \gamma', \delta \rangle \models \eta^\dagger$ und γ' unterscheidet sich höchstens in P_i von γ. Dann ist also $\langle \mathfrak{F}, \gamma, \delta \rangle \models (\exists P_i)\eta^\dagger = \varphi^\dagger$, wie gewünscht. Diese Argumentation ist umkehrbar, und damit ist dieser Fall gezeigt. Analog für $\varphi = (\forall P_i)\eta$. Nun noch $\varphi = \langle m \rangle \eta$. Es sei $\langle \mathfrak{F}, \beta, w \rangle \models \varphi$. Dann existiert ein y mit $w\ r^m\ y$ und $\langle \mathfrak{F}, \beta, y \rangle \models \eta$. Sei δ'

so gewählt, daß $\delta'(x_0) = y$, und $\delta'(x_i) = \delta(x_i)$ für alle $i > 0$. Dann gilt nach Induktionsannahme $\langle \mathfrak{F}, \gamma, \delta' \rangle \models \eta^\dagger$. Ist x_i eine Variable, die in $\eta^\dagger$ nicht auftritt, so sei $\delta''(x_i) := \delta'(x_i)$, $\delta''(x_0) := w$ und $\delta''(x_j) := \delta'(x_j)$ für alle $j \notin \{0, i\}$. Dann ist $\langle \mathfrak{F}, \gamma, \delta'' \rangle \models r^m(x_0, x_i); [x_i/x_0]\eta^\dagger = \varphi^\dagger$. Also gilt $\langle \mathfrak{F}, \gamma, \delta'' \rangle \models \varphi^\dagger$. Es gilt nun $\delta''(x_0) = w = \delta(x_0)$, und x_i ist gebunden. Daher ist auch $\langle \mathfrak{F}, \gamma, \delta \rangle \models \varphi^\dagger$). Wiederum ist die Argumentation umkehrbar, und dieser Fall ist bewiesen. Ebenso für $\varphi = [m]\eta$. $\dashv$

Es seien $\mathfrak{F} = \langle F, R, K \rangle$ und $\mathfrak{F} = \langle F', R', K' \rangle$ Kripke–Strukturen. Dann sei

$$\begin{array}{lcl} F + F' & := & F \times \{0\} \cup F' \times \{1\} \\ (R \oplus R')(m) & := & \{\langle \langle x, 0\rangle, \langle y, 0\rangle \rangle : \langle x, y\rangle \in R(m)\} \\ & := & \cup\{\langle \langle x, 1\rangle, \langle y, 1\rangle \rangle : \langle x, y\rangle \in R'(m)\} \\ (K \oplus K')(c) & := & \{\langle x, 0\rangle : x \in K(c)\} \\ & := & \cup\{\langle x, 1\rangle : x \in K'(c)\} \\ \mathfrak{F} \oplus \mathfrak{F}' & := & \langle F + F', R \oplus F', K \oplus K' \rangle \end{array}$$

Der folgende Satz ist nicht schwer zu zeigen.

Proposition 5.1.3 *Es sei φ eine Formel, und $\mathfrak{F}$ und $\mathfrak{F}'$ Kripke–Strukturen. Dann ist $\mathfrak{F} \oplus \mathfrak{F}' \models \varphi$ genau dann, wenn $\mathfrak{F} \models \varphi$ und $\mathfrak{F}' \models \varphi$.*

Daraus folgt unmittelbar, daß $\mathfrak{F} \oplus \mathfrak{F} \models \varphi$ genau dann, wenn $\mathfrak{F} \models \varphi$. Man bemerke, daß $\mathfrak{F} \oplus \mathfrak{F}$ nicht zusammenhängend ist bezüglich der Vereinigung der in ihm definierten Relationen. Dies wird im Folgenden noch von Wichtigkeit sein.

Übung 113. Beweisen Sie Proposition 5.1.3.

Übung 114. Es seien 1, 2 und 3 Modalitäten. Zeigen Sie, daß eine Kripke–Struktur die folgende Formel genau dann erfüllt, wenn $R(3) = R(1) \cup R(2)$ ist.

$$\langle 3\rangle p \leftrightarrow \langle 1\rangle p \vee \langle 2\rangle p$$

Übung 115. Es seien 1, 2 und 3 Modalitäten. Zeigen Sie, daß eine Kripke–Struktur die folgende Formel genau dann erfüllt, wenn $R(3) = R(1) \circ R(2)$ ist.

$$\langle 3\rangle p \leftrightarrow \langle 1\rangle\langle 2\rangle p$$

Übung 116. Es sei r ein 2–stelliges Relationssymbol. Zeigen sie, daß in einem Modell der **MSO** gilt: genau dann ist $\langle \mathfrak{M}, \gamma, \beta \rangle \models Q(x, y)$, wenn x $(r^{\mathfrak{M}})^*$ y (das heißt, y ist von x aus in endlich vielen R–Schritten erreichbar).

$$Q(x, y) := (\forall P)(P(x) \wedge (\forall yz)(P(y) \wedge y\ r\ z. \rightarrow .P(z)). \rightarrow .P(y))$$

5.2 Axiomatische Klassen I: Zeichenketten

Für die Zwecke dieses Kapitels werden wir die Zeichenketten auf eine neue Art kodieren. Diese ist, nebenbei bemerkt, eine etwas andere Formalisierung, als die in Abschnitt 1.4 besprochene. Die Unterschiede sind aber unerheblich.

Definition 5.2.1 *Eine **Z–Struktur über dem Alphabet** A ist ein Tripel der Form $\mathfrak{L} = \langle L, \prec^{\mathfrak{L}}, \{Q_a^{\mathfrak{L}} : a \in A\}\rangle$, wobei L eine beliebige endliche Menge, $\{Q_a^{\mathfrak{L}} : a \in A\}$ ein System von paarweise disjunkten Teilmengen von L, deren Vereinigung L ist, und $\prec^{\mathfrak{L}} \subseteq L \times L$ eine zweistellige Relation, für die Folgendes gilt.*

1. *$\langle L, \prec^{\mathfrak{L}}\rangle$ ist zusammenhängend.*
2. *$\prec^{\mathfrak{L}}$ ist zykelfrei.*
3. *Aus $x \prec^{\mathfrak{L}} y$ und $x \prec^{\mathfrak{L}} z$ folgt $y = z$.*
4. *Aus $y \prec^{\mathfrak{L}} x$ und $z \prec^{\mathfrak{L}} x$ folgt $y = z$.*

Wir schreiben Q_a anstelle von $Q_a^{\mathfrak{L}}$ und $\prec$ anstelle von $\prec^{\mathfrak{L}}$, falls Verwechslungen ausgeschlossen sind. Z–Strukturen sind nicht Zeichenketten. Es ist allerdings nicht schwer, eine Abbildung zu definieren, welche jeder Z–Struktur eine Zeichenkette zuordnet. Allerdings gibt es (falls $L \neq \varnothing$) unendlich viele, und zwar *eine echte Klasse* von Z–Strukturen, welche dieselbe Zeichenkette haben.

Es bezeichne nun **MSO** die Menge der Sätze aus der monadischen Prädikatenlogik 2. Stufe, welche außer den logischen Symbolen (Quantoren, Junktoren, Gleichheit) noch das Symbol $\preceq$, sowie eine 1–stellige Prädikatkonstante $\underline{a}$ für jedes $a \in A$ besitzt.

Definition 5.2.2 *Es sei $\mathcal{K}$ eine Menge von Z–Strukturen über einem Alphabet A. Dann bezeichnet $\mathsf{Th}\,\mathcal{K}$ die Menge $\{\varphi \in \textbf{\textit{MSO}} : \text{ für alle } \mathfrak{L} \in \mathcal{K} : \mathfrak{L} \models \varphi\}$. Diese nennen wir die **MSO–Theorie** von $\mathcal{K}$. Ist umgekehrt Φ eine Menge von Sätzen aus **MSO**, so sei $\mathsf{Mod}\,\Phi$ die Menge aller $\mathfrak{L}$, die jeden Satz aus Φ erfüllen. Diese nennen wir die **Modellklasse von** Φ.*

Wir heben ein paar elementare Fakten heraus.

Lemma 5.2.3 *Es gilt für alle Klassen $\mathcal{K}$ von Z–Strukturen über A und alle Teilmengen von **MSO**:*

1. *Ist* $\mathcal{K} \subseteq \mathfrak{L}$, *so* $\mathsf{Th}\,\mathcal{K} \supseteq \mathsf{Th}\,\mathfrak{L}$.
2. *Ist* $\Phi \subseteq \Psi$, *so* $\mathsf{Mod}\,\Phi \supseteq \mathsf{Mod}\,\Psi$.
3. *Genau dann ist* $\Phi \subseteq \mathsf{Th}\,\mathcal{K}$, *wenn* $\mathsf{Mod}\,\Phi \supseteq \mathcal{K}$.
4. $\mathcal{K} \subseteq \mathsf{Mod}\,\mathsf{Th}\,\mathcal{K}$.
5. $\Phi \subseteq \mathsf{Th}\,\mathsf{Mod}\,\Phi$.
6. $\mathsf{Th}\,\mathcal{K} = \mathsf{Th}\,\mathsf{Mod}\,\mathsf{Th}\,\mathcal{K}$.
7. $\mathsf{Mod}\,\Phi = \mathsf{Mod}\,\mathsf{Th}\,\mathsf{Mod}\,\Phi$.

Beweis. Zu 1. Sei $\mathcal{K} \subseteq \mathfrak{L}$ und φ ein Satz, der in jeder Struktur aus $\mathfrak{L}$ gilt. Dann gilt φ auch in jeder Struktur aus $\mathcal{K}$. Zu 2. Analog. Zu 3. Sei $\Phi \subseteq \mathsf{Th}\,\mathcal{K}$. Sei ferner $\mathfrak{L} \in \mathcal{K}$. Dann gilt nach Voraussetzung jeder Satz aus Φ in $\mathfrak{L}$. Also ist $\mathfrak{L} \in \mathsf{Mod}\,\Phi$. Dies zeigt $\mathcal{K} \subseteq \mathsf{Mod}\,\Phi$. Die Argumentation ist umkehrbar. Zu 4. Aus $\mathsf{Th}\,\mathcal{K} \subseteq \mathsf{Th}\,\mathcal{K}$ mit Hilfe von 3. Zu 5. Aus $\mathsf{Mod}\,\Phi \supseteq \mathsf{Mod}\,\Phi$ mit Hilfe von 3. Zu 6. Es ist wegen 4. $\mathsf{Th}\,\mathcal{K} \subseteq \mathsf{Th}\,\mathsf{Mod}\,\mathsf{Th}\,\mathcal{K}$ (Ersetzen von $\mathsf{Th}\,\mathcal{K}$ für Φ). Ferner folgt aus $\mathcal{K} \subseteq \mathsf{Mod}\,\mathsf{Th}\,\mathcal{K}$ (welches nach 4. gilt) mit 1. die andere Inklusion. Zu 7. Analog zu 6. $\dashv$

Wir erhalten daraus unmittelbar den folgenden

Theorem 5.2.4 *Die Abbildung* $\rho : \mathcal{K} \mapsto \mathsf{Mod}\,\mathsf{Th}\,\mathcal{K}$ *ist ein Hüllenoperator auf der Klasse der Klassen von Z–Strukturen über A. Ebenso ist* $\lambda : \Phi \mapsto \mathsf{Th}\,\mathsf{Mod}\,\Phi$ *ein Hüllenoperator auf der Menge aller Teilmengen von* ***MSO****.*

(Der Leser möge sich nicht irritieren lassen von dem Unterschied zwischen Klassen und Mengen. In der gewöhnlichen Mengenlehre muß man zwischen Klassen und Mengen unterscheiden. Modellklassen sind immer Klassen, während Formelklassen immer Mengen sind, weil sie Teilklassen von gewissen Mengen sind, nämlich der Menge aller Formeln. Diesen Unterschied kann man im Folgenden getrost vernachlässigen.) Wir nennen nun die Mengen der Form $\lambda(\Phi)$ **Logiken** und die Klassen der Form $\rho(\mathcal{K})$ **axiomatische Klassen**. Eine Klasse heißt **endlich axiomatisierbar**, falls sie die Form $\mathsf{Mod}(\Phi)$ hat für ein endliches Φ. Wir nennen eine Klasse von Z–Strukturen über A **regulär**, falls sie die Klasse aller Z–Strukturen einer regulären Sprache ist. Die Logik der Klasse aller Strukturen wird auch schlicht mit der monadischen Prädikatenlogik 2. Stufe bezeichnet. Formeln heißen **allgemeingültig**, falls sie in allen Strukturen gelten; dies ist also gleichbedeutend damit, daß sie der monadischen Prädikatenlogik 2. Stufe angehören.

Ein wichtiges Ergebnis dieses Kapitels ist das folgende Ergebnis aus [4].

Theorem 5.2.5 (Büchi) *Genau dann ist eine Klasse von Z–Strukturen eine endlich axiomatisierbare axiomatische Klasse, wenn sie eine endliche reguläre Sprachen ist, die ε nicht enthält.*

Dieser Satz besagt, daß man mit Mitteln der monadischen Prädikatenlogik 2. Stufe im Wesentlichen nur reguläre Mengen von Zeichenketten definieren kann. Will man andere als reguläre Klassen beschreiben, muß man also stärkere Beschreibungslogiken heranziehen. Der Beweis dieses Satzes erfordert einigen Aufwand, und wir werden ihn erst im folgenden Abschnitt vollenden können. Bevor wir beginnen, einiges zu der Formulierung des Satzes. Der Definition nach sind Modelle immer auf nichtleeren Trägermengen definiert. Deswegen definiert die Modellmenge einer Formel immer eine Menge, welche ε nicht enthält. Dies kann man im Prinzip ändern, aber dann ist die Z–Struktur von ε Modell jeder Formel, und dann ist $\varnothing$ zwar regulär, aber nicht **MSO**–axiomatitiserbar. Man kann also nicht völlige Übereinstimmung erzielen.

Beginnen wir mit der einfachen Richtung. Diese ist die Behauptung, daß jede reguläre Klasse axiomatisch ist. Sei dazu $\mathcal{K}$ eine reguläre Klasse, und S die dazugehörende reguläre Sprache. Dann existiert ein endlicher Automat $\mathfrak{A} = \langle Q, i_0, F, \delta\rangle$ mit $L(\mathfrak{A}) = S$. Wir wählen $Q := n$ für eine natürliche Zahl n und $i_0 = 0$. Betrachte den folgenden Satz:

$$
\begin{array}{lll}
\delta(\mathfrak{A}) := & (\forall xyz)(x \underline{\preceq} y \wedge x \underline{\preceq} z. \to y. \doteq z) & \text{(a)}\\
\wedge & (\forall xyz)(y \underline{\preceq} x \wedge z \underline{\preceq} x. \to .y \doteq z) & \text{(b)}\\
\wedge & (\forall P)\{(\forall xy)(x \underline{\preceq} y \to (P(x) \to P(y))) \wedge (\exists x)P(x). & \\
 & \quad \to .(\exists x)(P(x) \wedge (\forall y)(x \underline{\preceq} y \to \neg P(y))\} & \text{(c)}\\
\wedge & (\forall P)\{(\forall xy)(x \underline{\preceq} y \to (P(y) \to P(x))) \wedge (\exists x)P(x). & \\
 & \quad \to .(\exists x)(P(x) \wedge (\forall y)(y \underline{\preceq} x \to \neg P(y))\} & \text{(d)}\\
\wedge & (\forall P)\{(\forall xy)(x \underline{\preceq} y \to (P(x) \leftrightarrow P(y))) \wedge (\exists x)P(x). & \\
 & \quad \to .(\forall x)P(x)\} & \text{(e)}\\
\wedge & (\forall x) \bigvee \langle \underline{a}(x) : a \in A\rangle & \text{(f)}\\
\wedge & (\forall x) \bigwedge \langle \underline{a}(x) \to \neg \underline{b}(x) : a \neq b\rangle & \text{(g)}\\
\wedge & (\exists P_0 P_1 \ldots P_{n-1}) & \\
 & \quad \{(\forall x)((\forall y)\neg(y \underline{\preceq} x). \to .P_0(x)) & \\
 & \quad \wedge(\forall x)((\forall y)\neg(x \underline{\preceq} y). \to . \bigvee \langle P_i(x) : i \in F\rangle) & \\
 & \quad \wedge(\forall xy)(x \underline{\preceq} y \to \bigwedge \langle \underline{a}(y) \wedge P_i(x). & \\
 & \quad\quad \to . \bigvee \langle P_j(y) : j \in \delta(i,a)\rangle : a \in A\rangle)\} & \text{(h)}
\end{array}
$$

Lemma 5.2.6 *Es sei $\mathfrak{L}$ eine **MSO**–Struktur. Dann gilt $\mathfrak{L} \models \delta(\mathfrak{A})$ genau dann, wenn die Zeichenkette von $\mathfrak{L}$ in $L(\mathfrak{A})$ liegt.*

Beweis. Es sei $\mathfrak{L} \models \delta(\mathfrak{A})$, und sei $\mathfrak{L}$ eine **MSO**–Struktur. Dann existiert eine 2–stellige Relation $\prec$ (die Interpretation von $\underline{\preceq}$) und für jedes $a \in A$ eine Teilmenge

$Q_a \subseteq L$. Aufgrund der Definition von $\delta(\mathfrak{A})$ ist dann $\prec$ eine Relation, für die gilt: ist $x \prec y$ und $x \prec z$, so $x \prec z$ (wegen (a)); und ist $y \prec x$ und $z \prec x$, so ist $y = z$ (wegen (b)). Jede nichtleere Teilmenge welche unter $\prec$–Nachfolgern abgeschlossen ist, enthält ein letztes Element (Formel (c)) und jede nichtleere Teilmenge, welche unter Vorgängern abgeschlossen ist, enthält ein erstes Element (Formel (d)). Schließlich ist jede zusammenhängende nichtleere Menge schon ganz L (Formel (e)). Dies bedeutet, wie man leicht beweist, daß L endlich und zusammenhängend ist. Denn zunächst enthält L, wenn es selbst nicht leer ist, ein erstes Element, x_0. Sei induktiv x_{i+1} der (eindeutig bestimmte) $\prec$–Nachfolger von x_i. Die Folge der x_i kann nicht unendlich sein, denn sonst wäre sie eine unter Nachfolgern abgeschlossene Menge ohne letztes Element. Also ist sie endlich. Daraus folgt schon, daß $\prec$ zykelfrei ist. Nun ist sie auch unter Vorgängern abgeschlossen, und daher zusammenhängend. Daher ist sie schon ganz L. Also ist L endlich und zusammenhängend unter $\prec$.

Ferner gilt: jedes $x \in L$ ist in mindestens einer Menge Q_a enthalten (Formel (f)), und es ist in höchstens einer Menge enthalten (Formel (g)). Also ist $\mathfrak{L}$ eine Z–Struktur. Wir haben nun zu zeigen, daß ihre Zeichenkette in $L(\mathfrak{A})$ liegt. Das letzte Konjunktionsglied, Formel (h), besagt, daß wir Mengen $H_i \subseteq L$ finden für $i < n$ derart, daß wenn x das bezüglich $\prec$ erste Element ist, so ist $x \in H_0$, wenn x das bezüglich $\prec$ letzte Element ist, so ist $x \in H_j$ für ein $j \in F$, und wenn $x \in H_i$, $y \in Q_b$, $x \prec y$, so ist $y \in H_j$ für ein $j \in \delta(i, a)$. Dies bedeutet nichts anderes, als daß die Zeichenkette in $L(\mathfrak{A})$ ist. (Es gibt nämlich eine eineindeutige Beziehung zwischen akzeptierenden Läufen des Automaten und möglichen Mengen H_i. $x \in H_i$ bedeutet nichts anderes, als daß der Automat im Zustand i ist, wenn er bei x steht.) Es sei nun $\mathfrak{L} \not\models \delta(\mathfrak{A})$. So ist entweder $\mathfrak{L}$ keine Z–Struktur, oder aber es existiert kein akzeptierender Lauf des Automaten $\mathfrak{A}$. Also ist die Zeichenkette nicht in $L(\mathfrak{A})$. $\dashv$

Es ist nützlich, sich klarzumachen, daß wir $x \leq y$ wie folgt definieren können:

$$x \leq y := (\forall P)(P(x) \wedge (\forall wz)(P(w) \wedge w \preceq z. \rightarrow .P(z)) \rightarrow P(y))$$

Ferner schreiben wir $x < y$ für $x \leq y \wedge x \neq y$, sowie $x \prec y$ anstelle von $x \preceq y$. Es sei nun für eine Z–Struktur $\mathfrak{L} = \langle L, \prec, \{Q_a : a \in A\}\rangle$ folgende Struktur definiert.

$$M(\mathfrak{L}) := \langle L, \prec, \succ, <, >, \{Q_a : a \in A\}\rangle$$

Eine Struktur der Form $M(\mathfrak{L})$ nennen wir **MZ–Struktur**.

Nun werden wir die Umkehrung im Satz von Büchi beweisen. Dazu werden wir einen kleinen Umweg machen. Wir definieren wir eine modale Sprache basiert auf vier Modalitäten. Wir setzen $M := \{+, -, \prec, \succ\}$. Die Menge der Konstanten sei $C := \{Q_a : a \in A\}$. Die damit definierten Sprache nennen wir **QML**. Jetzt setzen

wir

$$\begin{array}{lllll}
\sigma := & & \langle\prec\rangle p \to \langle+\rangle p & \wedge & \langle\succ\rangle p \to \langle-\rangle p \\
& \wedge & p \to [\succ]\langle\prec\rangle p & \wedge & p \to [\prec]\langle\succ\rangle p \\
& \wedge & \langle\prec\rangle p \to [\prec] p & \wedge & \langle\succ\rangle p \to [\succ] p \\
& \wedge & \langle+\rangle\langle+\rangle p \to \langle+\rangle p & \wedge & \langle-\rangle\langle-\rangle p \to \langle-\rangle p \\
& \wedge & [+]([+]p \to p) \to [+]p & \wedge & [-]([-]p \to p) \to [-]p \\
& \wedge & \bigwedge\langle Q^a \to \neg Q^b : a \neq b\rangle & & \\
& \wedge & \bigvee\langle Q^a : a \in A\rangle & &
\end{array}$$

Wir nennen eine Struktur **zusammenhängend**, falls sie nicht von der Form $\mathfrak{F} \oplus \mathfrak{G}$ mit nichtleeren $\mathfrak{F}$ und $\mathfrak{G}$ ist. Wir wir schon erwähnt haben, können die **QML**–Formeln nicht zwischen zusammenängenden und unzusammenhängenden Strukturen unterscheiden.

Theorem 5.2.7 *Es sei $\mathfrak{F}$ eine zusammenhängende Kripke–Struktur für **QML**. Setze $Z(\mathfrak{F}) := \langle F, R(\prec), R(\succ), R(+), R(-), \{K(Q^a) : a \in A\}\rangle$. Genau dann ist $\mathfrak{F} \models \sigma$, wenn $Z(\mathfrak{F})$ eine MZ–Struktur über A ist.*

Beweis. Der Beweis ist nicht schwer, aber langwierig. Wir heben ein paar Fakten heraus. (a) *Genau dann ist $\mathfrak{F} \models \langle\prec\rangle p \to \langle+\rangle p$, wenn $R(\prec) \subseteq R(+)$.* Sei dazu $\mathfrak{F} \not\models \langle\prec\rangle p \to \langle+\rangle p$. Dann existiert eine Menge $\beta(p) \subseteq F$ und ein $x \in F$ derart, daß $\langle\mathfrak{F}, \beta, x\rangle \models \langle\prec\rangle p; \neg\langle+\rangle p$. Das heißt, daß ein $y \in F$ existiert mit $y \in \beta(p)$ und $x\ R(\prec)\ y$. Es kann dann aber $x\ R(+)\ y$ nicht gelten. Dies zeigt $R(\prec) \not\subseteq R(+)$. Sie umgekehrt $R(\prec) \not\subseteq R(+)$. Dann existieren x und y mit $x\ R(\prec)\ y$ aber nicht $x\ R(+)\ y$. Setze nun $\beta(p) := \{y\}$. Dann gilt $\langle\mathfrak{F}, \beta, x\rangle \models \langle\prec\rangle p; \neg\langle+\rangle p$. (b) *Genau dann ist $\mathfrak{F} \models p \to [\succ]\langle\prec\rangle p$, wenn $R(\succ) \subseteq R(\prec)^\smile$.* Dies zeigt man im Wesentlichen wie (a). (c) *Genau dann ist $\mathfrak{F} \models \langle\prec\rangle p \to [\prec] p$, wenn jeder Punkt höchstens einen $R(\prec)$–Nachfolger hat.* Nachweis wie (a). (d) *Genau dann ist $\mathfrak{F} \models \langle+\rangle\langle+\rangle p \to \langle+\rangle p$, wenn $R(+)$ transitiv ist.* Auch dies zeigt man wie (a). (d) *Genau dann ist $\mathfrak{F} \models [+]([+]p \to p) \to [+]p$, wenn $R(+)$ transitiv und zykelfrei ist.* Wegen (d) beschränken wir uns darauf, dies für den Fall mit transitiven $R(+)$ nachzuweisen. Dazu sei $\mathfrak{F} \not\models [+]([+]p \to p) \to [+]p$. Dann existiert ein β und ein x_0 mit $\langle\mathfrak{F}, \beta, x\rangle \models [+](p \to [+]p); \langle+\rangle\neg p$. Es existiert dann ein x_1 mit $x_0\ R(+)\ x_1$ und $x_1 \notin \beta(p)$. Dann gilt $x_1 \models [+]p \to p$ und deswegen $x_1 \models \langle+\rangle\neg p$. Wegen der Transitivität von $R(+)$ gilt außerdem $x_1 \models [+]([+]p \to p)$. Wir finden also eine unendliche Kette $\langle x_i : i \in \omega\rangle$ derart, daß $x_i\ R(+)\ x_{i+1}$. Also ist $R(+)$ nicht zykelfrei, da F endlich. Sei umgekehrt $R(+)$ nicht zykelfrei. Dann existiert eine Menge $\langle x_i : i \in \omega\rangle$ mit $x_i\ R(+)\ x_{i+1}$ für alle $i \in \omega$. Setze $\beta(p) := \{y :$ es existiert $i \in \omega : y\ R(+)\ x_i\}$. Dann gilt $\langle\mathfrak{F}, \beta, x_0\rangle \models [+]([+]p \to p); \langle+\rangle\neg p$. Denn sei $x\ R(+)\ y$. Fall 1. Für alle z mit $y\ R(+)\ z$ gilt $z \models p$. Dann existiert kein i mit $y\ R(+)\ x_i$ ($R(+)$ ist ja transitiv). Also ist $y \models p$. Dies zeigt $y \models [+]p \to p$. Da y beliebig, ist jetzt $x \models [+][[+]p \to p)$. Nun ist $x_1 \not\models p$ und $x_0\ R(+)\ x_1$. Also $x_0 \models \langle+\rangle\neg p$, wie verlangt. Die restlichen Eigenschaften sind einfach zu überprüfen. $\dashv$

Nun definieren wir eine Einbettung von **MSO** nach **QML**. Dazu einige Vorbereitungen. Wir man sich leicht überzeugt, gelten folgende Gesetze.

1. $(\forall x)\varphi \leftrightarrow \neg(\exists x)(\neg\varphi)$, $\neg\neg\varphi \leftrightarrow \varphi$.
2. $(\forall x)(\varphi_1 \wedge \varphi_2) \leftrightarrow (\forall x)\varphi_1 \wedge (\forall x)\varphi_2$, $(\exists x)(\varphi_1 \vee \varphi_2) \leftrightarrow (\exists x)\varphi_1 \vee (\exists x)\varphi_2$.
3. $(\forall x)(\varphi_1 \vee \varphi_2) \leftrightarrow (\varphi_1 \vee (\forall x)\varphi_2)$, $(\exists x)(\varphi_1 \wedge \varphi_2) \leftrightarrow (\varphi_1 \wedge (\exists x)\varphi_2)$, falls x nicht frei in φ_1 auftritt.
4. $(\forall x)(\varphi_1 \vee \varphi_2) \leftrightarrow (\varphi_2 \vee (\forall x)\varphi_1)$, $(\exists x)(\varphi_1 \wedge \varphi_2) \leftrightarrow (\varphi_2 \wedge (\exists x)\varphi_1)$, falls x nicht frei in φ_2 auftritt.

Schließlich gilt für jede Variable $y \neq x$:

$$(\forall x)\varphi(x,y) \leftrightarrow (\forall x)(x < y \to \varphi(x,y)) \wedge (\forall x)(y < x \to \varphi(x,y)) \wedge \varphi(y,y)$$

Wir definieren nun folgende Quantoren.

$$\begin{array}{lcl} (\forall x < y)\varphi & := & (\forall x)(x < y \to \varphi) \\ (\forall x > y)\varphi & := & (\forall x)(x > y \to \varphi) \\ (\exists x < y)\varphi & := & (\exists x)(x < y \to \varphi) \\ (\exists x > y)\varphi & := & (\exists x)(x > y \to \varphi) \end{array}$$

Wir nennen diese Quantoren **beschränkt**. Offensichtlich können wir jeden inbeschränkten Quantor $(\forall x)\varphi$ durch die Konjunktion beschränkter Quantoren ersetzen, sofern φ eine von x verschiedene freie Variable enthält.

Lemma 5.2.8 *Zu jeder **MSO**–Formel φ mit mindestens einer freien Objektvariable existiert eine **MSO**–Formel φ^g mit beschränkten Quantoren, derart, daß auf allen Z–Strukturen $\mathfrak{L}$ gilt $\mathfrak{L} \models \varphi \leftrightarrow \varphi^g$.*

Wir definieren Funktionen f, f^+, g und g^+ auf einstelligen Prädikaten wie folgt.

$$\begin{array}{lcl} (f(\varphi))(x) & := & (\exists y \prec x)\varphi(y), \\ (g(\varphi))(x) & := & (\exists y \succ x)\varphi(y), \\ (f^+(\varphi))(x) & := & (\exists y < x)\varphi(y), \\ (g^+(\varphi))(x) & := & (\exists y > x)\varphi(y). \end{array}$$

Schließlich sei O definiert durch

$$O(\varphi) := (\forall x)\neg\varphi(x)$$

Also besagt $O(\varphi)$ nichts anderes, als daß $\varphi(x)$ nirgends erfüllt ist. Es sei P_x eine Prädikatvariable, die nicht in φ vorkommt. Definiere dann $\{P_x/x\}\varphi$ induktiv wie folgt.

$$\begin{array}{lcll}
\{P_x/x\}(x \doteq y) & := & P_x(y) & \\
\{P_x/x\}(y \doteq x) & := & P_x(y) & \\
\{P_x/x\}(v \doteq w) & := & v \doteq w, & \text{falls } x \notin \{v, w\} \\
\{P_x/x\}(x \prec y) & := & (g(P_x))(y) & \\
\{P_x/x\}(y \prec x) & := & (f(P_x))(y) & \\
\{P_x/x\}(v \prec w) & := & v \prec w, & \text{falls } x \notin \{v, w\} \\
\{P_x/x\}(\underline{a}(x)) & := & \underline{a}(x) & \\
\{P_x/x\}(\neg\varphi) & := & \neg\{P_x/x\}\varphi & \\
\{P_x/x\}(\varphi_1 \wedge \varphi_2) & := & \{P_x/x\}\varphi_1 \wedge \{P_x/x\}\varphi_2 & \\
\{P_x/x\}(\varphi_1 \vee \varphi_2) & := & \{P_x/x\}\varphi_1 \vee \{P_x/x\}\varphi_2 & \\
\{P_x/x\}((\exists y)\varphi) & := & (\exists y)\{P_x/x\}\varphi, & \text{falls } y \neq x \\
\{P_x/x\}((\exists x)\varphi) & := & (\exists x)\varphi & \\
\{P_x/x\}((\forall P)\varphi) & := & (\forall P)(\{P_x/x\}\varphi) & \\
\{P_x/x\}((\exists P)\varphi) & := & (\exists P)(\{P_x/x\}\varphi) &
\end{array}$$

Sei $\gamma(P_x) = \{\beta(x)\}$. Dann gilt

$$\langle \mathfrak{M}, \gamma, \beta\rangle \models \varphi \text{ gdw. } \langle \mathfrak{M}, \gamma, \beta\rangle \models \{P_x/x\}\varphi$$

Setze daher

$$(Ex)\varphi(x) := (\exists P_x)(\neg O(P_x) \wedge O(P_x \wedge f^+(P_x)) \wedge O(P_x \wedge g^+(P_x)). \rightarrow .\{P_x/x\}\varphi)$$

Dann ist für alle Z–Strukturen $\mathfrak{L}$

$$\langle \mathfrak{L}, \gamma, \beta\rangle \models (\exists x)\varphi \text{ gdw. } \langle \mathfrak{L}, \gamma, \beta\rangle \models (Ex)\varphi$$

Man beachte, daß $(\forall x)\varphi \leftrightarrow \neg(\exists x)\neg\varphi$; also ist die gesonderte Behandlung des Allquantors hier nicht nötig. Es ist also induktiv möglich, sämtliche erststufigen Quantoren unter Zuhilfenahme von f, f^+, g, g^+, O sowie (Ex) zu eliminieren. Jetzt sind wir soweit, eine Einbettung von **MSO** in **QML** definieren zu können. Es sei $h : P \rightarrow PV$ eine Bijektion von den Prädikatvariablen von **MSO** auf die Aussage-

variable von **QML**.

$$\begin{array}{lcl}
(\underline{a}(x))^\diamond & := & Q^a \\
(P(y))^\diamond & := & h(P) \\
(O(\varphi))^\diamond & := & \neg\varphi^\diamond \wedge [+]\neg\varphi^\diamond \wedge [-]\neg\varphi^\diamond \\
(f(\varphi))^\diamond & := & \langle\succ\rangle\varphi^\diamond \\
(g(\varphi))^\diamond & := & \langle\prec\rangle\varphi^\diamond \\
(f^+(\varphi))^\diamond & := & \langle -\rangle\varphi^\diamond \\
(g^+(\varphi))^\diamond & := & \langle +\rangle\varphi^\diamond \\
(\neg\varphi)^\diamond & := & \neg\varphi^\diamond \\
(\varphi_1 \wedge \varphi_2)^\diamond & := & \varphi_1^\diamond \wedge \varphi_2^\diamond \\
(\varphi_1 \vee \varphi_2)^\diamond & := & \varphi_1^\diamond \vee \varphi_2^\diamond \\
((\exists P)\varphi)^\diamond & := & (\exists P)\varphi^\diamond \\
((\forall P)\varphi)^\diamond & := & (\forall P)\varphi^\diamond
\end{array}$$

Man beachte, daß die Übersetzung von $(Ex)\varphi$ damit auch festgelegt ist.

Theorem 5.2.9 *Es sei φ eine **MSO**–Formel mit höchstens einer freien Variablen, die Objektvariable x_0. Dann existiert eine **QML**–Formel φ^M dergestalt, daß für alle Z–Strukturen $\mathfrak{L}$ gilt:*

$$\langle \mathfrak{L}, \beta\rangle \models \varphi(x_0) \text{ gdw. } \langle M(\mathfrak{Z}), \beta(x_0)\rangle \models \varphi(x_0)^M$$

Korollar 5.2.10 *Modulo der Identifikation $\mathfrak{L} \mapsto M(\mathfrak{L})$ definieren **MSO** und **QML** dieselben Modellklassen von zusammenhängenden, nichtleeren, endlichen Z–Strukturen. Ferner: genau dann ist $\mathcal{K}$ eine endlich axiomatisierbare **MSO**–Klasse von Z–Strukturen, wenn $M(\mathcal{K})$ eine endlich axiomatisierbare **QML**–Klasse von MZ–Strukturen ist.*

Dies beweist nun, daß es genügen würde, wenn wir von endlich axiomatisierbaren **QML**–Klassen von MZ–Strukturen nachweisen können, daß sie reguläre Sprachen definieren. Dies werden wir jetzt tun. Man beachte, daß wir zum Beweis lediglich Grammatiken betrachten müssen, welche Regeln der Form $X \to a \mid aY$ haben, und keine Regeln der Form $X \to \varepsilon$. Ferner können wir anstelle von regulären Grammatiken mit Grammatiken* arbeiten, wo wir also eine Menge von Startsymbolen anstelle eines einzigen haben.

Sei $G = \langle \Sigma, N, A, R\rangle$ eine reguläre Grammatik* und $\vec{x}$ eine Zeichenkette. Für eine Ableitung von $\vec{x}$ definieren wir eine Z–Struktur über $A \times N$ (!) wie folgt. Wir betrachten die Grammatik $G^\times := \langle \Sigma, N, A \times N, R^\times\rangle$, welche aus folgenden Regeln besteht.

$$R^\times := \{X \to \langle a, X\rangle\, Y : X \to aY \in R\}$$

Die Abbildung $h : A \times N \to A : \langle a, X\rangle \mapsto a$ definiert einen Homomorphismus von $(A \times M)^*$ nach A^*, den wir auch mit h bezeichnen. Ebenso liefert sie uns

eine Abbildung von Z–Strukturen über $A \times N$ nach Z–Strukturen über A. Jede G–Ableitung von $\vec{x}$ definiert in eindeutiger Weise eine $G^\times$–Ableitung einer Zeichenkette $\vec{x}^\times$ mit $h(\vec{x}^\times) = \vec{x}$ und diese also eine Z–Struktur

$$\mathfrak{M} = \langle L, \prec, \{Q^{\mathfrak{M}}_{\langle a,X\rangle} : \langle a, X\rangle \in A \times N\}\rangle \ .$$

Zu diesem Modell definieren wir jetzt ein Modell über dem Alphabet $A \cup N$ wie folgt.

$$\langle L, \prec^{\mathfrak{L}}, \{Q^{\mathfrak{L}}_a : a \in A\}, \{Q^{\mathfrak{L}}_X : X \in N\}\rangle$$

Dabei ist $w \in Q^{\mathfrak{L}}_a$ genau dann, wenn $w \in Q^{\mathfrak{M}}_{\langle a,X\rangle}$ für ein X, und $w \in Q^{\mathfrak{L}}_X$ genau dann, wenn $w \in Q^{\mathfrak{M}}_{\langle a,X\rangle}$ für ein $a \in A$. Wir bezeichnen dieses Modell ebenfalls mit $\vec{x}^\times$.

Definition 5.2.11 *Es sei $G = \langle \Sigma, N, A, R\rangle$ eine reguläre Grammatik* und $\varphi \in$ **QML** eine konstante Formel (mit Konstanten in A). Wir sagen, G sei **treu für** φ, falls es eine Teilmenge $H_\varphi \subseteq N$ gibt derart, daß für jede Zeichenkette $\vec{x}$ und jedes $\vec{x}^\times$ gilt: genau dann ist $\langle \vec{x}^\times, w\rangle \models \varphi$, wenn ein $X \in H_\varphi$ existiert mit $w \in Q_X$. Wir sagen, H_φ **codiert** φ **bezüglich** G.*

Die Bedeutung dieser Definition ist die Folgende. Gegeben eine Menge H und eine Formel φ. H codiert φ bezüglich G, falls in jeder Ableitung einer Zeichenkette $\vec{x}$ φ genau dort wahr wird, wo ein Nichtterminalzeichen aus H auftritt. Der Leser mag sich von folgenden einfachen Fakten überzeugen.

Proposition 5.2.12 *Es sei G eine reguläre Grammatik*, und es codiere H φ bezüglich G, und K χ. Dann gilt:*

- *$N - H$ codiert $\neg\varphi$ bezüglich G.*
- *$H \cap K$ codiert $\varphi \wedge \chi$ bezüglich G.*
- *$H \cup K$ codiert $\varphi \vee \chi$ bezüglich G.*

Wir werden induktiv zeigen, daß man jede **QML**–Formel in eine reguläre Grammatik codieren kann, wobei man allerdings die Ausgangsgrammatik etwas erweitern muß.

Definition 5.2.13 *Es seien $G_1 = \langle \Sigma_1, N_1, A, R_1\rangle$ und $G_2 = \langle \Sigma_2, N_2, A, R_2\rangle$ reguläre Grammatiken*. Dann sei*

$$G_1 \times G_2 := \langle \Sigma_1 \times \Sigma_2, N_1 \times N_2, A, R_1 \times R_2\rangle$$

wo

$$\begin{aligned} R_1 \times R_2 := \ & \{\langle X_1, X_2\rangle \to a \quad \langle Y_1, Y_2\rangle : X_1 \to aY_1 \in R_1, X_2 \to aY_2 \in R_2\} \\ & \cup \{\langle X_1, X_2\rangle \to a : X_1 \to a \in R_1, X_2 \to a \in R_2\} \end{aligned}$$

*Dies nennen wir das **Produkt** der Grammatiken* G_1 und G_2.*

Es gilt

Proposition 5.2.14 *Es seien G_1 und G_2 Grammatiken* über A. $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$.*

Der folgende Satz ist nicht schwer zu zeigen und daher als Übung überlassen.

Lemma 5.2.15 *Es sei φ in G_2 durch H codiert. Dann ist φ in $G_1 \times G_2$ durch $N_1 \times H$ codiert und in $G_2 \times G_1$ durch $H \times N_2$.*

Definition 5.2.16 *Es sei φ eine **QML**–Formel. Ein **Code** für φ ist ein Paar $\langle G, H\rangle$, wo $L(G) = A^*$ und H φ in G codiert. φ heißt **codierbar**, falls es einen Code für φ gibt.*

Angenommen, φ besitzt einen Code $\langle G, H\rangle$, und es sei G' gegeben. Dann gilt $L(G' \times G) = L(G')$, und φ ist in $G' \times G$ durch $N' \times H$ codiert. Offensichtlich genügt es daher, für jede Formel einen Code zu finden. Wir werden allerdings noch rasch eine Folgerung ziehen, die das Leben erleichtern wird.

Lemma 5.2.17 *Es sei Δ eine endliche Menge von codierbaren Formeln. Dann existiert eine Grammatik* G und Mengen H_φ, $\varphi \in \Delta$, derart, daß $\langle G, H_\varphi\rangle$ Code von φ ist.*

Beweis. Sei $\Delta := \{\delta_i : i < n\}$. Sei $\langle G_i, M_i\rangle$ Code von δ_i, $i < n$. Setze $G := \mathsf{X}_{i<n} G_i$. Ferner setze $H_i := \mathsf{X}_{j<i} N_i \times H_i \times \mathsf{X}_{i<j<n} N_j$. Iterierte Anwendung von Lemma 5.2.15 liefert die Behauptung. $\dashv$

Theorem 5.2.18 *Jede konstante **QML**–Formel ist codierbar.*

Beweis. Der Beweis erfolgt durch Induktion über den Aufbau von φ. Wir beginnen mit dem Code von $\underline{a}$, $a \in A$. Dieser ist die folgende Grammatik* G_a. Wir haben $N := \{X, Y\}$, wir erlauben sowohl X als auch Y als Startsymbol. Die Regeln sind

$$X \to a \mid aX \mid aY, \quad Y \to b \mid bX \mid bY$$

wobei b alle Elemente aus $A - \{a\}$ durchläuft. Der Code ist jetzt $\langle G_a, \{X\}\rangle$, wie man leicht nachrechnet. Die Induktionsbehauptung ist für $\neg$, $\wedge$, $\vee$ und $\to$ durch Proposition 5.2.12 gezeigt. Nun zu dem Fall $\varphi = \langle \prec\rangle \eta$. Wir nehmen an, η sei codierbar

und der Code sei $C_\eta = \langle G_\eta, H_\eta\rangle$. Jetzt definieren wir G_φ. Es sei $N_\varphi := N_\eta \times \{0,1\}$ und $\Sigma_\varphi := \Sigma_\eta \times \{0,1\}$. Schließlich seien die Regeln von der Form

$$\langle X,1\rangle \to a\,\langle Y,0\rangle, \quad \langle X,1\rangle \to a\,\langle Y,1\rangle\ ,$$

wo $X \to aY \in R_\eta$ und $Y \in H_\eta$ und

$$\langle X,0\rangle \to a\,\langle Y,0\rangle, \quad \langle X,0\rangle \to a\,\langle Y,1\rangle\ ,$$

wo $X \to aY \in R_\eta$ und $Y \notin H_\eta$. Ferner haben wir noch Regeln der Form

$$\langle X,0\rangle \to a, \quad \langle X,1\rangle \to a\ ,$$

für alle $X \to a \in R_\eta$. Man überlegt sich leicht, daß zu jeder Regel $X \to aY$ oder $X \to a$ eine Regel in G_φ existiert. Der Code von φ ist dann $\langle G_\varphi, N_\eta \times \{1\}\rangle$. Nun zu dem Fall $\varphi = \langle\succ\rangle\eta$. Wir nehmen wieder an, η sei codierbar und der Code sei $C_\eta = \langle G_\eta, H_\eta\rangle$. Jetzt definieren wir G_φ. Es sei $N_\varphi := N_\eta \times \{0,1\}$, $\Sigma_\varphi := \Sigma_\eta \times \{0\}$. Schließlich sei R_φ die Menge der Regeln von der Form

$$\langle X,0\rangle \to a\,\langle Y,1\rangle, \quad \langle X,1\rangle \to a\,\langle Y,1\rangle\ ,$$

wo $X \to aY \in R_\eta$ und $X \in H_\eta$ und

$$\langle X,0\rangle \to a\,\langle Y,0\rangle, \quad \langle X,1\rangle \to a\,\langle Y,1\rangle\ ,$$

wo $X \to aY \in R_\eta$ und $X \notin H_\eta$; und schließlich nehmen wir für jede Regel $X \to a$ die Regeln

$$\langle X,0\rangle \to a, \quad \langle X,1\rangle \to a$$

auf. Der Code von φ ist dann $\langle G_\varphi, N_\eta \times \{1\}\rangle$. Nun zu $\varphi = \langle+\rangle\eta$. Wiederum ist $N_\varphi := N_\eta \times \{0,1\}$, sowie $\Sigma_\varphi := \Sigma_\eta \times \{0,1\}$ die Menge der Startsymbole. Die Regeln sind nun von der Form

$$\langle X,0\rangle \to a\,\langle Y,0\rangle\ ,$$

falls $X \to aY \in R_\eta$. Ferner haben sie die Form

$$\langle X,1\rangle \to a\,\langle Y,1\rangle\ ,$$

falls $X \to aY \in R_\eta$ und $Y \notin H_\eta$. Ferner nehmen wir die Regeln

$$\langle X,1\rangle \to a\,\langle Y,0\rangle\ ,$$

auf, wenn $X \to aY \in R_\eta$ und $Y \in H_\eta$, sowie schließlich alle Regeln der Form

$$\langle X,i\rangle \to a\ ,$$

wo $X \to a \in R_\eta$ und $i \in \{0,1\}$. Der Code von φ ist damit $\langle G_\varphi, N_\eta \times \{1\}\rangle$. Jetzt zu $\varphi = \langle-\rangle\eta$. Es sei $N_\varphi := N_\eta \times \{0,1\}$, sowie $\Sigma_\varphi := \Sigma_\eta \times \{0\}$ die Menge der Startsymbole. Die Regeln sind nun von der Form

$$\langle X,1\rangle \to a\,\langle Y,1\rangle\ ,$$

falls $X \to aY \in R_\eta$. Ferner haben sie die Form

$$\langle X, 0\rangle \to a\ \langle Y, 1\rangle\ ,$$

falls $X \to aY \in R_\eta$ und $X \in H_\eta$. Ferner nehmen wir die Regeln

$$\langle X, 0\rangle \to a\ \langle Y, 0\rangle\ ,$$

auf, wenn $X \to aY \in R_\eta$ und $Y \notin H_\eta$, sowie schließlich alle Regeln der Form

$$\langle X, i\rangle \to a\ ,$$

wo $X \to a \in R_\eta$ und $i \in \{0, 1\}$. Der Code von φ ist damit $\langle G_\varphi, N_\eta \times \{1\}\rangle$. Den größten Aufwand macht der letzte Fall, $\varphi = (\forall p_i)\eta$. Zunächst führen wir ein ein neues Alphabet ein, nämlich $A \times \{0, 1\}$, und eine Konstante $\underline{c}$. Es sei $\underline{a} := \underline{\langle a, 0\rangle} \vee \underline{\langle a, 1\rangle}$. Ferner sei $\underline{c} \leftrightarrow \bigvee_{a\in A} \underline{\langle a, 1\rangle}$. Dann ist $\underline{\langle a, 1\rangle} \leftrightarrow \underline{a} \wedge \underline{c}$ und $\underline{\langle a, 0\rangle} \leftrightarrow \underline{a} \wedge \neg\underline{c}$. Dann sei $\eta' := \eta[\underline{c}/p_i]$. Auf diese Formel können wir die Induktionsbehauptung anwenden. Es sei Δ die Menge der Teilformeln von η'. Für eine Teilmenge $\Sigma \subseteq \Delta$ sei

$$L_\Sigma := \bigwedge_{\delta\in\Sigma} \delta \wedge \bigwedge_{\delta\notin\Sigma} \neg\delta$$

Zunächst einmal finden wir eine Grammatik G und Mengen H_δ, $\delta \in \Delta$, derart, daß $\langle G, H_\delta\rangle$ δ codiert. Also existieren H_Σ derart, daß $\langle G, H_\Sigma\rangle$ L_Σ codiert. Jetzt bilden wir als erstes die Grammatik G^1 mit den Nichtterminalsymbolen $N \times \wp(\Delta)$ und dem Alphabet $A \times \{0, 1\}$. Die Regelmenge ist die Menge aller

$$\langle X, \Sigma\rangle \to \langle a, i\rangle \quad \langle X', \Sigma'\rangle\ ,$$

wo $X \to aX' \in R$ ist, $X \in H_\Sigma$ und $X' \in H_{\Sigma'}$; ferner alle Regeln der Form

$$\langle X, \Sigma\rangle \to \langle a, i\rangle\ ,$$

wo $X \to a \in R$ und $X \in H_\Sigma$. Setze $H^1_\Sigma := H_\Sigma \times \{\Sigma\}$. Wiederum sieht man leicht, daß $\langle G^1, H^1_\Sigma\rangle$ ein Code für Σ ist für jedes $\Sigma \subseteq \Delta$. Wir gehen jetzt zu der Grammatik G^2 über, mit $N^2 := N \times \{0, 1\} \times \wp(\Delta)$ und $A^2 := A$, sowie den Regeln

$$\langle X, i, \Sigma\rangle \to a \quad \langle X', i', \Sigma'\rangle$$

wo $\langle X, \Sigma\rangle \to \langle a, i\rangle \quad \langle X', \Sigma'\rangle \in R^1$ und

$$\langle X, i, \Sigma\rangle \to a$$

wo $\langle X, \Sigma\rangle \to \langle a, i\rangle \in R^1$. Als Letztes definieren wir G^3. $N^3 := N \times \wp(\wp(\Delta))$, $A^3 := A$. Ferner sei

$$\langle X, \mathbb{A}\rangle \to a \quad \langle Y, \mathbb{B}\rangle$$

genau dann eine Regel, wenn $\mathbb{B}$ die Menge aller Σ' ist, für die $\Sigma \in \mathbb{A}$ sowie $i, i' \in \{0,1\}$ existieren mit

$$\langle X, i, \Sigma\rangle \to a \quad \langle Y, i', \Sigma'\rangle \in R^2 \ .$$

Ebenso ist genau dann

$$\langle X, \mathbb{A}\rangle \to a \in R^3 \ ,$$

wenn ein $\Sigma \in \mathbb{A}$ existiert und ein $i \in \{0,1\}$ mit

$$\langle X, i, \Sigma\rangle \to a \in R^2 \ .$$

Setze $H_\varphi := \{\Sigma : \eta' \in \Sigma\}$. Wir behaupten: $\langle G^3, H_\varphi\rangle$ ist ein Code für φ. Zum Beweis sei $\vec{x}$ eine Zeichenkette, und es sei eine beliebige G^3–Ableitung von $\vec{x}$ gegeben. Wir konstruieren daraus eine G^1–Ableitung. Es sei $\vec{x} = \prod_{i<n} x_i$. Nach Voraussetzung haben also wir eine Ableitung

$$\langle X_i, \mathbb{A}_i\rangle \to x_i \quad \langle X_{i+1}, \mathbb{A}_{i+1}\rangle$$

für $i < n-1$ und

$$\langle X_{n-1}, \mathbb{A}_{n-1}\rangle \to x_{n-1} \ .$$

Nach Konstruktion existiert ein j_{n-1} sowie ein $\Sigma_{n-1} \in \mathbb{A}_{n-1}$ mit

$$\langle X_{n-1}, j_{n-1}, \Sigma_{n-1}\rangle \to a \in R^2 \ .$$

Absteigend bekommen wir jetzt für jedes $i < n-1$ ein j_i sowie ein Σ_i mit

$$\langle X_i, j_i, \Sigma_i\rangle \to a \quad \langle X_{i+1}, j_{i+1}, \Sigma_{i+1}\rangle \in R^2 \ .$$

Wir haben also eine G^2–Ableitung von $\vec{x}$. Daraus bekommen wir sofort eine G^1–Ableitung. Diese ist über dem Alphabet $A \times \{0,1\}$. Nach Voraussetzung ist in G^1 η' durch $H_{\eta'}$ codiert. Dann gilt η' an allen Knoten i mit $X_i \in H_{\eta'}$. Dies ist die Menge aller i mit $X_i \in H_\Sigma$ für ein $\Sigma \subseteq \Delta$ mit $\eta' \subseteq \Delta$. Dies ist genau die Menge aller i mit $\langle X_i, \mathbb{A}_i\rangle \in H_\varphi$. Also gilt $\langle X_i, \mathbb{A}_i\rangle \in H_\varphi$ genau dann, wenn die Z–Struktur von $\vec{x}$ bezüglich der gegebenen G^3–Ableitung φ bei i erfüllt. Dies war aber zu zeigen. Ebenso kann man aus einer G^1–Ableitung kann man auch eine G^3–Ableitung konstruieren, wie man leicht sieht. $\dashv$

Wir sind nun fast am Ziel. Als letztes müssen wir noch einsehen, wie wir aus der Tatsache der Codierbarkeit einer Formel auch eine Grammatik bekommen, die nur Zeichenketten erzeugt, die diese Formel erfüllen. Sei also Φ eine endliche Menge von **MSO**–Formeln. Wir dürfen sofort annehmen, daß diese Sätze sind (wenn nicht, quantifizieren wir die freien Variablen universell ab). Nach Korollar 5.2.10 können wir annehmen, daß wir es statt mit **MSO**–Sätzen mit **QML**–Formeln zu tun haben. Ferner ist eine endliche Konjunktion von **QML**–Formeln wieder eine **QML**–Formel, sodaß wir uns auf eine einzige **QML**–Formel φ zurückziehen

können. Nach Satz 5.2.18 existiert für φ ein Code $\langle G, H\rangle$, $G = \langle \Sigma, N, A, R\rangle$. Setze $G^\varphi := \langle \Sigma \cap H, N \cap H, A, R_H\rangle$, wo

$$R_H := \{X \to aY \in R : X, Y \in H\} \cup \{X \to a \in R : X \in H\}$$

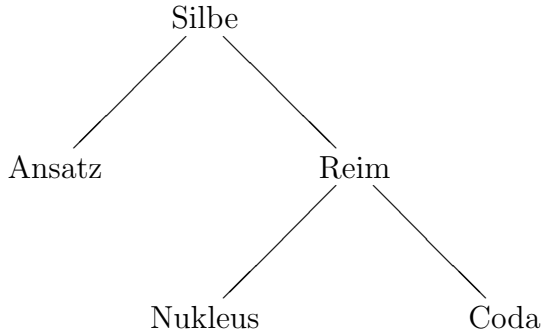
Nun existiert eine G^φ–Ableitung von $\vec{x}$ offensichtlich genau dann, wenn $\vec{x}^\times \models \varphi$.

Übung 117. Zeigen Sie, daß jede (!) Sprache der Schnitt von regulären Sprachen ist. Dieser Schnitt ist möglicherweise unendlich. (Dies heißt also, daß wir auf den Zusatz der endlichen Axiomatisierbarkeit im Satz von Büchi nicht verzichten können.)

5.3 Einiges zu Phonologie und Silbenstruktur

In diesem Abschnitt werden wir auf Silbenstruktur und phonologische Regeln eingehen. Dies ist nur zum Teil als Anwendung der Ergebnisse des vorigen Abschnitts zu verstehen. Vieles von dem, was wir hier sagen werden, ist ohne die Theorie von Büchi mindestens ebenso leicht zu verstehen. Trotzdem ist es für das Verständnis von Sprachstruktur unerläßlich, die hier angesprochenen Themen wenigstens ansatzweise würdigen zu können. Der Leser ist alledings aufgefordert zu verifizieren, daß die phonologischen Prinzipien, die wir hier ansprechen, in **MSO** formulierbar sind, sodaß man, wenn dies erforderlich ist, einen endlichen Automaten bauen kann, der sie verifiziert. Außerdem folgt daraus sofort, daß die phonologische Wohlgeformtheit einer Lautfolge in linearer (also Echt)Zeit überprüft werden kann.

Der bei weitem wichtigste Anwendungsbereich für reguläre Sprachen ist die Phonologie. Denn nach allem, was man weiß, sind die phonologischen Regeln einer Sprache stets so geartet, daß sich eine reguläre Sprache ergibt. Dabei sind die Strukturen, welche man annimmt, keineswegs identisch mit denen einer Typ 3 Grammatik. Man nimmt nämlich an, daß sich Phoneme in einer Silbe organisieren, welche ihrerseits folgende Struktur hat.



Eine Silbe besteht also aus einem **Ansatz** und einem **Reim**, welcher seinerseits aus einem **Nukleus** und einer **Coda** besteht. Der Nukleus besteht hierbei in der Regel aus den in der Silbe auftretenden Vokalen, der Ansatz enthält die dem Nukleus vorausgehenden Konsonanten, die Coda die den Vokalen folgenden Konsonanten. Zum Beispiel besteht die Silbe ['ʃtraik] (`Streik`) aus dem Nukleus [ai], dem Ansatz [ʃtr] und der Coda [k]. Es gibt in jeder Sprache eine Höchstgrenze für die Anzahl der Phoneme, die in jedem der drei Teilen auftretenden Phoneme. So erscheinen im Deutschen nie mehr als 3 Konsonanten im Ansatz, höchstens 2 Vokale im Nukleus, und höchstens 5 (!) Konsonanten in der Coda. (Siehe [34], Seite 774.) Das Maximum wird einerseits durch das Wort `Streik` im Ansatz realisiert, andererseits durch das einsilbige Wort ['ʃɪmpfst] (`schimpfst`). Ferner gibt es gewisse teilweise universelle Gesetze über die Verteilung von Konsonanten in der Silbe. In Ansatz ist zum Beispiel ein liquider Laut ([l] oder [r]) stets an letzter Stelle. Deswegen ist *[ls] kein möglicher Ansatz. Ferner kann es weder in dem Ansatz noch in der Coda mehr als einen Plosivlaut ([b], [p], [d], [t], [g] oder [k]) geben. Im Deutschen folgt [l] nicht auf [t] im Ansatz, aber es gibt Sprachen, in denen dies erlaubt ist. Eine allgemeine Theorie der Silbenstruktur auszuarbeiten, ist an dieser Stelle nicht möglich. Es gilt jeodch grob gesagt Folgendes. Die Laute werden in einer Sonoritätshierarchie angeordnet. Diese ist wie folgt.

tiefe Vokale	> mittlere Vokale	> hohe Vokale	> r–Laute
[a], [o]	[æ], [œ]	[i], [y]	[r]
> Nasale; Laterale	> sth. Frikative	> sth. Plosive	> stl. Frikative
[s], [ʃ]	[m], [n]; [l]	[z], [ʒ]	[b], [d]
> stl. Plosive			
[p], [t]			

Eine Silbe ist nun so organisiert.

> *Silbenstruktur.* In einer Silbe steigt die Sonorität monoton bis zum Nukleus, und fällt anschließend monoton. Ferner enthält der Nukleus mindestens einen Sonoritätsgipfel, das heißt, einen Laut mit maximaler Sonorität.

Dies bedeutet, daß es in einer Silbe einen Laut geben muß, der in der Hierarchie mindestens so hoch ist wie die anderen Laute. Dieser ist dann im Nukleus. Dieser heißt **Sonoritätsgipfel**. Von diesem Gipfel aus gesehen fällt die Sonorität monoton nach links wie nach rechts gehend. Dies erklärt, warum wir im Anlaut [tr] (['tro:st], `Trost`) aber nicht [rt] finden, umgekehrt aber im Auslaut [rt] (['pfeʁt], `Pferd`) haben aber nicht [tr]. Ebenso ist [pfr] ein wohlgeformter Ansatz. Allerdings macht das [ʃ]

im Anlaut Kummer. So haben wir ['ʃtra:sə], `Straße`, obwohl [ʃ] in der Hierarchie höher liegt als [t]. Es wird deswegen [ʃ] als nicht zum Ansatz gehörig betrachtet, was dann aber die Silbenstruktur weiter verkompliziert. Es ist nicht nötig, daß eine Silbe einen Vokal enthält. Als Beispiel diene das Wort `prügeln`. Es wird — unabhängig von seiner Schreibung — ['pry: gln] gesprochen. Diese enthält zwei Silben: [pry:] und [gln]. Die zweite Silbe hat im Nukleus [l]. Sehen wir, wie dies folgt. Wir notieren die Laute in Folge und schreiben darüber eine Ziffer, die Hierarchiestufe in der Sonoritätshierarchie.

1	6	7	3	5	5
p	r	y:	g	l	n

Wir müssen also mindestens zwei Silben haben. Der Leser möge prüfen, daß, falls wir zwei Silben annehmen, die folgenden Gliederungen mit dem obenstehenden Gesetz kompatibel sind.

$$\text{pry:} \cdot \text{gln}, \text{pry: g} \cdot \text{ln}$$

Ferner könnten sowohl [l] alleine wie auch [ln] den Nukleus der zweiten Silbe bilden.

Die Phonologie war im Übrigen auch die erste formal ausgearbeitet Theorie, in der Merkmale eine Rolle spielten. Es wurde allgemein davon ausgegangen, daß ein Phonem wie [b] keine unanlysierbare Einheit bildet, sondern aus mehreren Merkmalen zusammengesetzt ist, welche einen ganz realen, das heißt, artikulatorischen Hintergrund haben. Wir können an [b] mehrere Merkmale isolieren:

1. [b] wird artikuliert durch den Lippenverschluß. Man nennt es daher auch einen **Bilabiallaut**. Im Gegensatz dazu wird [d] an den Zähnen mit der Zungenspitze artikuliert (und heißt deswegen **Apikodentallaut**).
2. Bei der Artikulation von [b] bewegen sich die Stimmbänder. Daher heißt [b] **stimmhaft**. Im Gegensatz dazu ist [p] stimmlos.
3. [b] ist ein plötzlicher Laut, er kann nicht beliebig lange artikuliert werden. Daher heißt er auch ein **Plosivlaut**. Im Gegensatz dazu ist [v] ein Laut, welchen man so lange artiluieren kann, wie man möchte. Er heißt deswegen auch **kontinuierlich**.

Diese drei Merkmale definieren den Artikulationsort (1.) und die Artikulationsart (2. und 3.). Die Merkmale besitzen einige Abhängigkeiten untereinander. So sind Vokale stets stimmhaft und kontinuierlich. Im Deutschen sind stimmlose Plosivlaute behaucht; wir sprechen also [t] mit einem nachfolgenden Hauchlaut, weswegen in

alten Büchern auch oft `Theil` anstelle von `Teil` geschrieben wurde. Diese sogenannte Aspiration entfällt allerdings dann, wenn davor ein [ʃ] erscheint. In [ˈʃtʊmpf] ist der Hauchlaut nicht existent. Vokale sind im Deutschen nicht alleine lang oder kurz. Auch die Vokalqualität ändert sich mit der Länge. Lange Vokale sind gespannt. So wird `i` als [ɪ] gesprochen, wenn es kurz ist, und als [i] wenn es lang ist (man probiere `Sinn` [ˈzɪn]) gegenüber `Tief` [ˈti:f]). Ebenso für die anderen Vokale. Hier ist eine Korrespondenztabelle für lange und kurze Vokale.

i:	ɪ	y:	ʏ
a:	ʌ	ø:	œ
o:	ɔ	æ:	æ
u:	ʊ, ɷ	ɛ:	æ
e:	ə		

Lediglich das æ klingt lang genauso wie kurz. Es ist daher im allgemeinen nicht leicht zu sagen, welches Merkmal distinktiv wirkt: ist es die Länge oder die Spannung? Das ist insbesondere dann heikel, wenn Deutsche eine Sprache verstehen muß, in der Länge unabhängig ist von der Spannung (zum Beispiel das Finnische). Dann können sie sich nicht auf die Spannung als zusätzliche Hilfe verlassen.

Die Attribut–Wert Strukturen stellen einen Schreibweise dar, in welcher man Merkmalsstrukturen aufschreiben kann.

$$\begin{bmatrix} \text{ORT} & : & \textit{bilabial} \\ \text{STIMMHAFT} & : & + \\ \text{PLOSIV} & : & + \end{bmatrix}$$

Genauso können wir uns der monadische Prädikatenlogik oder der (quantifizierten) Modallogik bedienen. Die Wahl ist hier nur eine der Bequemlichkeit. Das Interessante an diesen Analysen ist, daß sie eine konzise Darstellung von phonologischen Regeln erlaubt. So können wir folgende Prinzip im Deutschen formulieren.

Auslautbedingung. Ein Konsonsant in einer Coda ist niemals stimmhaft.

Nehmen wir das Verb `geben`. Der Imperativ lautet `gib`. Die Coda ist [p] und nicht [b]. Dies ist im Einklang mit den Erfordernissen. Warum aber steht [p] anstelle von [b] und nicht [t]? Die Antwort ist einfach. Es gibt einen Prozeß, welcher in der Coda den Wert des Merkmals STIMMHAFT auf + setzt. Dementsprechend heißt es [ˈli:s] (`lies`) und nicht [ˈli:z], [ˈle:k] (`leg`) und nicht [ˈle:g] (aber [ˈle:gə], `lege`). Dies ist ein Beweis für die Richtigkeit dieser Analyse. In der Phonologie formuliert man dies also als Regel.

Auslautverhärtung. Ein Konsonsant wird in einer Coda stimmlos.

Dies bedeutet: in der Coda operiert die folgende Ersetzung

$$[\textsc{stimmhaft} : +] \Longrightarrow [\textsc{stimmhaft} : -]$$

Solche Ersetzungen haben wir bisher nicht betrachtet. Wodurch kommen sie zustande, und wie kann man sie bschreiben? Die erste Frage ist in diesem Fall schnell beantwortet. Die Silbenstruktur ist in der lexikalischen Repräsentation überwiegend nicht festgelegt. Erst wenn nach Anwendung morphologischer Prozesse eine Wortform entstanden ist, wird sie syllabifiziert, das heißt, in eine Silbenstruktur analysiert. Dann greifen aber die Prinzipien, wie etwa die Auslautbedingung. Um sie zu erfüllen, werden die Konsonanten im Auslaut minimal verändert; es wird also das Merkmal der Stimmhaftigkeit auf $-$ gesetzt, wenn dies erforderlich ist. Dies legt also nahe, phonologische Regeln erst nach morphologischen Regeln anzuwenden. Die Morphologie arbeitet daher zwar mit Phonemen, kann aber deren tatsächliche Artikulation nur begrenzt bestimmen. Ob ein [b] in der morphologischen Repräsentation tatsächlich als [b] ausgesprochen wird oder als [p], die entscheidet sich erst nach erfolgter Syllabifikation und Anwendung phonologischer Regeln.

Die zweite Frage erfordert einen neue Technik, den *Transducer*. Ein Transducer ist eine Maschine, welche Zeichenketten übersetzt. Wir beschreiben hier nur den einfachsten Transducer, den sogenannten **endlichen Transducer**.

Definition 5.3.1 *Seien A und B Alphabete. Ein **(partieller) endlicher Transducer von** A nach B ist ein Quadrupel $\mathfrak{T} = \langle Q, i_0, F, \delta\rangle$, sodaß $i_0 \in Q$, $F \subseteq Q$ und $\delta : Q \times A_\varepsilon \to \wp(Q \times B^*)$, wobei $\delta(q, \vec{x})$ stets endlich ist für jedes $\vec{x} \in A_\varepsilon$. Q heißt die Menge der **Zustände**, i_0 heißt der **Anfangszustand**, F die Menge der **Endzustände** und δ die Übergangsfunktion. $\mathfrak{T}$ heißt **deterministisch**, falls $\delta(q, a)$ genau ein Element hat für jedes $q \in Q$ und $a \in A$.*

Es heißt A das **Eingabealphabet** und B das **Ausgabealphabet**. Der Transducer unterscheidet sich von einem endlichen Automaten dadurch, daß er nicht nur eine Übergangsfunktion besitzt, welche festlegt, welcher Zustand als nächstes angesteuert werden kann, sondern die auch noch bestimmt, daß eine gewisse Zeichenkette ausgegeben wird. Man beachte, daß der Transducer auch leere Übergänge erlaubt, was interessant ist, weil in diesem Falle trotzdem der Automat eine Ausgabe machen kann. Man schreibt

$$q \xrightarrow{\vec{x}:\vec{y}} q' ,$$

wenn der Transducer vom Zustand q aus unter Eingabe der Zeichenkette $\vec{x}$ ($\in A^*$) in

den Zustand q' übergeht, und die Kette $\vec{y}$ ($\in B^*$) ausgibt. Dies ist wie folgt definiert.

$$q \xrightarrow{\vec{x}:\vec{y}} q', \quad \text{falls} \begin{cases} & (q', \vec{y}) \in \delta(q, \vec{x}) \\ \text{oder} & \text{für gewisse } q'', \vec{u}, \vec{u}', \vec{v}, \vec{v}' : \\ & q \xrightarrow{\vec{u}:\vec{v}} q'' \xrightarrow{\vec{u}':\vec{v}'} q' \\ & \text{und } \vec{x} = \vec{u} \cdot \vec{u}', \vec{y} = \vec{v} \cdot \vec{v}' \end{cases}$$

Schließlich definiert man

$$L(\mathfrak{T}) := \{\langle \vec{x}, \vec{y} \rangle : \text{ existiert } q \in F \text{ mit } i_0 \xrightarrow{\vec{x}:\vec{y}} q\} .$$

Transducer lassen sich benutzen, um den Effekt von Regeln zu beschreiben. So kann man einen Syllabifizierungstransducer $\mathfrak{Syl}$ schreiben, der Folgendes tut. Das Eingabealphabet ist $A \cup \{\Box, \sharp\}$, wo A das Phonemalphabet ist, $\Box$ die Wortgrenze und $\sharp$ die Silbengrenze. Das Ausgabealphabet ist $A \times \{a, n, c\} \cup \{\Box, \sharp\}$. Dabei steht a für *Ansatz*, n für *Nukleus* und c für *Coda*. Somit annotiert die Maschine jedes Phonem dahingehend, ob es im Ansatz, im Nukleus oder in der Coda auftritt. Zusätzlich werden Silbengrenzen eingetragen, wo es notwendig ist. (In der Eingabe werden nur solche Silbengrenzen eingetragen, welche nicht vorhersagbar sind, wie etwa die Trennung `Atmo-sphäre`.) Jetzt schreiben wir eine Maschine $\mathfrak{AWh}$, welche den Effekt der Auslautverhärtung simuliert. Sie hat einen Zustand, i_0, ist deterministisch und die Übergänge sind beschrieben durch $\langle [\mathrm{b}], c\rangle : \langle [\mathrm{p}], c\rangle$, $\langle [\mathrm{d}], c\rangle : \langle [\mathrm{t}], c\rangle$, $\langle [\mathrm{g}], c\rangle : \langle [\mathrm{k}], c\rangle$ sowie $\langle [\mathrm{z}], c\rangle : \langle [\mathrm{s}], c\rangle$ und $\langle [\mathrm{v}], c\rangle : \langle [\mathrm{f}], c\rangle$ (das Symbol [v] entspricht unserem Deutschen `w`). (Überall sonst haben wir $\langle P, \alpha\rangle : \langle P, \alpha\rangle$, P eine Phonem, $\alpha \in \{a, c, n\}$.

Wir wollen nun ein allgemeines Theorem beweisen, welches als *Transducertheorem* bekannt ist. Es besagt, daß das Bild unter einem Transducer von einer regulären Sprache wieder eine reguläre Sprache ist. Der Beweis ist nicht schwer. Zunächst einmal kann man die Funktion $\delta : Q \times A_\varepsilon \to \wp(Q \times B^*)$ durch eine Funktion $\delta^\diamond : Q^\diamond \times A_\varepsilon \to \wp(Q^\diamond \times B_\varepsilon)$ ersetzen, indem man entsprechend mehr Zustände hinzufügt. Die Details dieser Konstruktion überlassen wir dem Leser. Nun ersetzen wir diese durch eine Funktion $\delta^2 : Q \times A_\varepsilon \times B_\varepsilon \to \wp(Q)$. Was wir jetzt bekommen, ist ein Automat über dem Alphabet $A \times A$. Wir übernehmen die Notation aus dem Abschnitt 4.3 und notieren Paare als $\vec{x} \otimes \vec{y}$. Einziger Unterschied: wir definieren

$$(\vec{u} \otimes \vec{v}) \cdot (\vec{w} \otimes \vec{x}) := (\vec{u} \cdot \vec{w}) \otimes (\vec{v} \cdot \vec{x})$$

Definition 5.3.2 *Es sei R ein regulärer Term. Wir definieren $L^2(R)$ wie folgt.*

$$\begin{array}{lcll} L^2(0) & := & \varnothing & \\ L^2(\vec{x} \otimes \vec{y}) & := & \{\vec{x} \otimes \vec{y}\} & \vec{x} \otimes \vec{y} \in A_\varepsilon \times B_\varepsilon \\ L^2(R \cdot S) & := & \{\mathfrak{x} \cdot \mathfrak{y} : \mathfrak{x} \in L^2(R), \mathfrak{y} \in L^2(S)\} & \\ L^2(R \cup S) & := & L^2(R) \cup L^2(S) & \\ L^2(R^*) & := & (L^2(R))^* & \end{array}$$

*Eine **reguläre Relation auf** A ist eine Relation der Form $L^2(R)$ für einen regulären Term R.*

Theorem 5.3.3 *Genau dann ist $Z \subseteq A^* \times B^*$ eine reguläre Relation, wenn es einen endlichen Transducer $\mathfrak{T}$ gibt mit $L(\mathfrak{T}) = Z$.*

Dies ist im Wesentlichen eine Konsequenz aus dem Satz von Kleene. Anstelle des Alphabets A wurde hier das Alphabet $A \times A$ gewählt. Man muß sich nur überlegen, daß die Transitionen $\varepsilon : \varepsilon$ nichts hinzufügen. Wir können daraus eine Menge Folgerungen ziehen.

Korollar 5.3.4 *Es gilt*

- *Reguläre Relationen sind unter Schnitt abgeschlossen.*
- *Ist $H \subseteq A^*$ regulär, so auch $H \times B^*$. Ist $K \subseteq B^*$ regulär, so auch $A^* \times K$.*
- *Ist $Z \subseteq A^* \times B^*$ eine reguläre Relation, so auch die Projektionen*
  - $\pi_1[Z] := \{\vec{x} :$ *es existiert* $\vec{y}$ *mit* $\langle \vec{x}, \vec{y} \rangle \in Z\}$
  - $\pi_2[Z] := \{\vec{y} :$ *es existiert* $\vec{x}$ *mit* $\langle \vec{x}, \vec{y} \rangle \in Z\}$
- *Ist Z eine reguläre Relation und $H \subseteq A^*$ eine reguläre Menge, so ist $Z[H] := \{\vec{y} :$ es existiert $\vec{x} \in H$ mit $\langle \vec{x}, \vec{y} \rangle \in Z\}$ eine reguläre Menge.*

Man unterscheidet bei dem Transducer zwei mögliche Gebrauchsweisen. Die erste ist die einer Maschine, die bei gegebenem Paar von Eingabeketten prüft, ob sie sich in Relation befinden. Die zweite, ob sich zu einer gegebenen Zeichenkette eine andere Zeichenkette finden läßt, die mit dieser in Relation steht. Im ersten Fall kann man stets eine deterministische Maschine haben, im zweiten in der Regel nicht. Die Relation $\{\langle \mathtt{a}, \mathtt{a}^n \rangle : n \in \omega\}$ ist regulär, aber es gibt keinen deterministischen Übersetzungsalgorithmus. Man findet leicht auch eine Sprache, in der es in keiner Richtung, links nach rechts wie rechts nach links, einen deterministischen Übersetzungsalgorithmus gibt.

Als Anwendung für Transducer und das Transducertheorem nehmen wir die historische Sprachwissenschaft. In der historischen Sprachwissenschaft formuliert man sogenannte Lautkorrespondenzen. Im einfachsten Fall modellieren sie die Entwicklung von einer Sprache in eine andere. Wir nehmen als Beispiel das Indogermanische. Dies ist eine rekonstruierte Sprache, deren Eigenschaften uns nicht direkt bekannt sind, sondern aus den Sprachen erschlossen wurden, in die sie sich — so glaubt man — entwickelt hat. Je zuverlässiger die Lautgesetze, desto besser bestätigt ist

natürlich die Existenz und die Eigenarten des Indogermanischen. Wir geben hier eine Korrespondenztabelle.

Indogermanisch	Sanskrit	Griechisch	Latein	(Bedeutung)
g^{u}hermós	gharmah	thermós	formus	*warm*
o̯uis	avih	óis	ovis	*Schaf*
s̯uos	svah	hos	suus	*sein*
septḿ	saptá	heptá	septem	*sieben*
dék^{u}m̥	dáśa	déka	decem	*zehn*
néu̯os	návah	néos	novus	*neu*
ĝénos	janaḥ	génos	genus	*Geschlecht*
s̯uepnos	svapnah	hypnos	somnus	*Schlaf*

Einige Laute in der einen Sprache haben exakte Enstprechungen in der nächsten. Zum Beispiel entspricht [p] in allen Sprachen wieder [p]. Bei anderen ist die Entsprechung nicht eineindeutig. Die indogermanischen Laute [e] und [a] werden im Sanskrit zu [a]. Also hat das Sanskrit [a] in den anderen Sprachen mehrere Entsprechungen. Schließlich ist es noch so, daß Laute sich verschieden entwickeln je nach der Umgebung. Im Ansatz wird indogermanisches [s] im Sanskrit zu [s], aber am Wortende zu [h]. Die Einzelheiten mögen hier nun nicht weiter interessieren. Es sei aber gesagt, daß man für die Lautkorrespondenzen tatsächlich einen Transducer schreiben kann. Dem Leser sei zur Übung empfohlen, einen solchen zu entwickeln.

Kommen wir zum Schluß noch einmal auf den Satz von Büchi zurück. Wir wollen eine Anwendung bringen, die nicht ganz banal ist. Im Finnischen gibt es sogenannte *Vokalharmonie*. Es gibt drei Arten von Vokalen: dunkle ([a], [o], [u], geschrieben `a`, `o` und `u`), helle ([æ, œ, y], geschrieben `ä`, `ö` und `y`) und neutrale ([e], [i], geschrieben `e` und `i`). Viele Suffixe besitzen zwei verschiedene Formen, eine dunkle und eine helle. Falls das Wort, an das sie angehängt werden, einen dunklen Vokal enthält, so wird die dunkle Variante gewählt, ansonsten die helle. Finnische Worte haben in der Regel nie zugleich einen dunklen und einen hellen Vokal, sodaß auch nach Anhängen des Suffixes diese Eigenschaft erfüllt ist.

> *Vokalharmonie (Finnisch).* Ein Wort enthält nicht zugleich einen dunklen und einen hellen Vokal.

Wir haben zum Beispiel die Nomina `talo` (*Haus*), `urheilu` (*Sport*), `hissi` (*Aufzug*), sowie `tyttö` (*Tochter*). Das Suffix des Adessivs ist `lla` (dunkel) oder `llä` (hell). Wendet man das Prinzip der Vokalharmonie an, so bekommt man

`talolla`, `urheilulla`, `hissillä`, `tyttöllä`

Die Vokalharmonie geht nur bis zur Wortgrenze. Es können sehr wohl zwei Worte unterschiedlicher Harmoniewerte miteinander kombiniert werden. Man hat zum Beispiel `osakeyhtiö` (*Aktiengesellschaft*). Dies besteht aus den zwei Worten `osake` (*Aktie*) sowie `yhtiö` (*Gesellschaft*). Das Suffix richtet sich nach dem näher liegenden Wort. Wir haben also

`osakeyhtiöllä`

Aufgrund der hier angegebenen Daten läßt sich relativ leicht ein **MSO**–Satz angeben, der die Vokalharmonie im Finnischen kodifiziert. Daraus folgt dann, daß es einen Automaten gibt, der sie überprüfen kann.

Das Interessante an diesem Beispiel ist hierbei auch noch Folgendes. Im Prinzip geht man davon aus, daß phonologische Prozesse Kontaktphänomene sind. Ein Beispiel dafür ist die Einfügung von [p] zwischen [m] und [f], welches nur der besseren Sprechbarkeit dient; oder der Ausschluß von [s] zu [ʃ] vor Plosiven im Deutschen. Auch die Auslautverhärtung gehört hierher. Man nimmt einfach an, daß ein Konsonant stimmlos wird vor einem stimmlosen Konsonant. Das Silbenende wird als stimmlos definiert. Ausgehend von dem Silbenende werden alle Konsonanten der Coda rückwärts gehend erfaßt und stimmlos. Wichtig hierbei ist also, daß es jeweils die aneinandergrenzenden Laute sind, die einander beeinflussen; eine Fernwirkung ist hier ausgeschlossen. Die Vokalharmonie im Finnischen bildet nun eine Ausnahme. Es gibt zunächst keinen Grund anzunehmen, daß es sich hier um ein Kontaktphänomen handelt. Denn der Suffixvokal im Adessivsuffix ist durch das doppelte [l] von dem Wortstamm getrennt. Sofern wir nicht annehmen wollen, daß es zwei oder gar drei Sorten von [l] gibt, ein helles, neutrales und ein dunkles, kann die Vokalqualität des Suffixvokals nicht von den Wurzelvokalen bestimmt werden. In der autosegmentalen Phonologie ist man allerdings einen Weg gegangen, der beide Intuitionen wieder vereinen kann. Hier nimmt man an, daß gewisse Merkmale sich auf einem eigenen Strang, genannt **Tier** (das ist Englisch, spricht sich aber fast genauso wie deutsch `Tier`) befinden. Nicht alle Laute haben Merkmale in jedem Tier. So spielt zum Beispiel bei Vokalen der Artikulationsort keine Rolle (er ist eigentlich immer gleich), sodaß sie nicht auf dem dafür vorgesehenen Tier auftreten. Nehmen wir nun an, daß es einen eigenen Tier für das Harmoniemerkmal gibt, so ist es leichter, Harmonie als Kontaktphänomen zu interpretieren. Denn auf dem Harmonie–Tier sind der letzte Wurzelvokal und der Suffixvokal adjazent. Allerdings — und das ist das Unglück — ist der letzte Vokal nicht unbedingt entscheidend. Denn falls er neutral ist, so ist eine Entscheidung nicht möglich, ohne die davor liegenden Vokale zu kennen. Auch hier kann man Abhilfe schaffen. Man nimmt an, daß neutrale Vokale auch nicht auf dem Harmonietier vertreten sind. Dann ist Harmonie in der Tat ein Kontaktphänomen geworden.

5.4 Axiomatische Klassen II: Erschöpfend geordnete Bäume

Der Beweis von Büchi über axiomatische Klassen von Zeichenketten hat ein sehr interessantes Analogon für erschöpfend geordnete Bäume. Wir werden ihn auch beweisen, wobei wir allerdings nur diejenigen Fakten zeigen werden, die nicht unmittelbar übetragen werden können. Anschließend werden wir auf die Bedeutung dieses Satzes für die Syntaxtheorie eingehen. Der Leser möge sich noch einmal mit den Begriffen aus Abschnitt 1.4 vertraut machen. Geordnete Bäume sind Strukturen über einer Sprache, welche 2 zweistellige Relationssymbole besitzt, $\sqsubset$ und $<$. Wir nehmen auch noch Marken aus A und N (!) hinzu und bekommen so die Sprache $\mathbf{MSO}^b$. In ihr ist die Menge der erschöpfend geordneten Bäume eine endlich axiomatisierbare Klasse. Wir betrachten zunächst die Postulate. $<$ ist transitiv und irreflexiv, $\uparrow x$ ist linear für jedes x, und es gibt ein größtes Element, und jede Teilmenge hat ein bezüglich $<$ größtes und ein bezüglich $<$ kleinstes Element. Daraus folgt insbesondere, daß unterhalb eines beliebigen Elements ein Blatt liegt. Hier sind nun in derselben Reihenfolge die Axiome, die dies beschreiben.

$$
\begin{array}{l}
(\forall xyz)(x < y \wedge y < z. \rightarrow .x < z) \\
(\forall x)\neg(x < x) \\
(\forall xyz)(x < y \wedge x < z. \rightarrow .y < z \vee y \doteq z \vee y > z) \\
(\exists x)(\forall y)(y < x \vee y \doteq x) \\
(\forall P)(\exists x)(\forall y)(P(x) \wedge P(y). \rightarrow .x > y \vee x \doteq y) \\
(\forall P)(\exists x)(\forall y)(P(x) \wedge P(y). \rightarrow .x < y \vee x \doteq y)
\end{array}
$$

Im Folgenden benutzen wir die Abkürzung $x \leq y := x < y \vee x \doteq y$. Nun legen wir Ordnungsaxiome nieder. $\sqsubset$ ist transitiv und irreflexiv, es ist linear auf den Blättern, und es gilt $x \sqsubset y$ genau dann, wenn für alle Blätter u unterhalb von x und alle Blätter $v \leq y$ gilt $u \sqsubset v$. Schließlich gibt es nur endlich viele Blätter, was wir dadurch ausdrücken können, daß wir verlangen, daß jede Menge von Knoten ein bezüglich $\sqsubset$ kleinstes und ein bezüglich $\sqsubset$ größtes Element hat. Wir setzen $b(x) := \neg(\exists y)(y < x)$.

$$
\begin{array}{l}
(\forall xyz)(x \sqsubset y \wedge y \sqsubset z. \rightarrow .x \sqsubset z) \\
(\forall x)\neg(x \sqsubset x) \\
(\forall xy)(b(x) \wedge b(y). \rightarrow .x \sqsubset y \vee x \doteq y \vee y \sqsubset x) \\
(\forall xy)(x \sqsubset y \leftrightarrow (\forall uv)(b(u) \wedge u \leq x \wedge b(v) \wedge v \leq y. \rightarrow .u \sqsubset v)) \\
(\forall P)\{(\forall x)(P(x) \rightarrow b(x)) \rightarrow (\exists y)(P(y) \wedge (\forall z)(P(z) \rightarrow \neg(z \sqsubset y)))\} \\
(\forall P)\{(\forall x)(P(x) \rightarrow b(x)) \rightarrow (\exists y)(P(y) \wedge (\forall z)(P(z) \rightarrow \neg(z \sqsupset y)))\}
\end{array}
$$

Als Drittes müssen wir die Verteilung der Marken regulieren.

$$
\begin{array}{l}
(\forall x)(b(x) \leftrightarrow \bigvee\langle \underline{a}(x) : a \in A\rangle) \\
(\forall x)(b(x) \rightarrow \bigwedge\langle \underline{a}(x) \rightarrow \neg\underline{b}(x) : a \neq b\rangle) \\
(\forall x)(\neg b(x) \rightarrow \bigvee\langle \underline{A}(x) : A \in N\rangle) \\
(\forall x)(\neg b(x) \rightarrow \bigwedge\langle \underline{A}(x) \rightarrow \neg\underline{B}(x) : A \neq B\rangle)
\end{array}
$$

Die Tatsache, daß ein Baum erschöpfend geordnet ist, wird durch folgende Formel beschrieben:

$$(\forall xy)(\neg(x \leq y \vee y \leq x). \rightarrow .x \sqsubset y \vee y \sqsubset x)$$

Proposition 5.4.1 *Folgendes sind endlich* $\boldsymbol{MSO}^b$*–axiomatisierbare Klassen.*

1. *Die Klasse der geordneten Bäume.*
2. *Die Klasse der erschöpfend geordneten Bäume.*

Genauso können wir eine quantifizierte modale Sprache definieren. Wir ändern jedoch hier die Basis an Relationen. Grundlage ist die Übung 1.4. Wir haben jetzt 8 Operatoren, $M_8 := \{\diamondsuit, \diamondsuit^+, \diamondsuit, \diamondsuit^+, \diamondsuit, \diamondsuit^+, \diamondsuit, \diamondsuit^+\}$, welche mit den folgenden Relationen korrespondieren: $\prec$, $<$, $\succ$, $>$, *unmittelbar linke Schwester von*, *linke Schwester von*, *unimttelbare rechte Schwester von*, sowie *linke Schwester von*. Diese sind $\mathbf{MSO}^b$–definierbar. Es sei $\mathfrak{B} = \langle B, <, \sqsubset\rangle$ ein erschöpfend geordneter Baum. Dann definieren wir $R : M_8 \to B \times B$ wie folgt.

$$\begin{array}{lcl}
x\ R(\diamondsuit^+)\ y & := & x \sqsubset y \wedge (\exists z)(x \prec z \wedge y \prec z) \\
x\ R(\diamondsuit)\ y & := & x\ R(\diamondsuit^+)\ y \wedge \neg(\exists z)(x\ R(\diamondsuit^+)\ z \wedge z\ R(\diamondsuit^+)\ y) \\
x\ R(\diamondsuit^+)\ y & := & x \sqsupset y \wedge (\exists z)(x \succ z \wedge y \succ z) \\
x\ R(\diamondsuit)\ y & := & x\ R(\diamondsuit^+)\ y \wedge \neg(\exists z)(x\ R(\diamondsuit^+)\ z \wedge z\ R(\diamondsuit^+)\ y) \\
x\ R(\diamondsuit^+)\ y & := & x > y \\
x\ R(\diamondsuit)\ y & := & x \succ y \\
x\ R(\diamondsuit^+)\ y & := & x < y \\
x\ R(\diamondsuit)\ y & := & x \prec y
\end{array}$$

Die resultierende Struktur nennen wir $M(\mathfrak{B})$. Ist nun B sowie R gegeben, so sind die Relationen $\prec$, $\succ$, $<$, $>$, $\sqsubset$, sowie $\sqsupset$ aus diesen Relationen definierbar. Zunächst definieren wir $\diamondsuit^*\varphi := \varphi \wedge \diamondsuit^+\varphi$, und ebenso für die anderen Relationen. Es ist dann $R(\diamondsuit^*) = \Delta \cup R(\diamondsuit^+)$.

$$\begin{array}{lcl}
\prec & = & R(\diamondsuit) \\
\succ & = & R(\diamondsuit) \\
< & = & R(\diamondsuit^+) \\
> & = & R(\diamondsuit^+) \\
\sqsubset & = & R(\diamondsuit^*) \circ R(\diamondsuit^+) \circ R(\diamondsuit^*) \\
\sqsupset & = & R(\diamondsuit^*) \circ R(\diamondsuit^+) \circ R(\diamondsuit^*)
\end{array}$$

Analog wie bei den Zeichenketten kann man zeigen, daß folgende Eigenschaften axiomatisierbar sind: $R(\diamondsuit^+)$ ist transitiv und irreflexiv mit konverser Relation $R(\diamondsuit^+)$;

$R(\diamond\!\!\!\!\cdot^+)$ ist die transitive Hülle von $R(\diamond\!\!\!\!\cdot)$ und $R(\diamond\!\!\!\!\cdot^+)$ die transitive Hülle von $R(\diamond\!\!\!\!\cdot)$. Ebenso für $R(\diamond\!\!\!\!\cdot^+)$ und $R(\diamond\!\!\!\!\cdot)$, $R(\diamond\!\!\!\!\cdot^+)$ und $R(\diamond\!\!\!\!\cdot)$. Wir können auch mittels des folgenden Axioms axiomatisch erfassen, daß $\uparrow x$ stets linear sein muß:

$$\diamond\!\!\!\!\cdot^+ p \wedge \diamond\!\!\!\!\cdot^+ q. \rightarrow .\diamond\!\!\!\!\cdot^+(p \wedge q) \vee \diamond\!\!\!\!\cdot^+(p \wedge \diamond\!\!\!\!\cdot^+ q) \vee \diamond\!\!\!\!\cdot^+(q \wedge \diamond\!\!\!\!\cdot^+ p)$$

Die anderen Axiome machen etwas mehr Mühe. Man beachte zunächst Folgendes.

Lemma 5.4.2 *Es sei $\langle B, <, \sqsubset\rangle$ ein erschöpfend geordneter Baum und $x, y \in B$. Dann gilt $x \neq y$ genau dann, wenn (a) $x < y$ oder (b) $x > y$ oder (c) $x \sqsubset y$ oder (d) $x \sqsupset y$.*

Daher folgende Definitionen.

$$\begin{array}{lcl} \langle\neq\rangle\varphi & := & \diamond\!\!\!\!\cdot^+\varphi \wedge \diamond\!\!\!\!\cdot^+\varphi \vee \diamond\!\!\!\!\cdot^*\diamond\!\!\!\!\cdot^+\diamond\!\!\!\!\cdot^*\varphi \vee \diamond\!\!\!\!\cdot^*\diamond\!\!\!\!\cdot^+\diamond\!\!\!\!\cdot^*\varphi \\ \boxtimes\varphi & := & \varphi \wedge [\neq]\varphi \end{array}$$

Wir nehmen jetzt als zusätzliche Axiome:

$$\begin{array}{ll} \boxtimes\varphi \rightarrow \boxminus^+\varphi, & \boxtimes\varphi \rightarrow \boxminus^+\varphi, \\ \boxtimes\varphi \rightarrow \boxplus^+\varphi, & \boxtimes\varphi \rightarrow \boxplus^+\varphi, \\ \boxtimes\varphi \rightarrow \boxtimes\boxtimes\varphi, & \boxtimes\varphi \rightarrow \varphi, \\ \varphi \rightarrow \boxtimes\neg\boxtimes\neg\varphi\,. & \end{array}$$

(Die meisten sind allerdings bereits ableitbar. Das Axiomensystem ist also nicht minimal.) Diese Axiome sorgen dafür, daß in einer Struktur jeder Knoten, der irgendwie mittels einer der acht Basisrelationen erreichbar ist, in einem Schritt mit $R(\boxtimes)$ erreichbar ist. Es ist

$$R(\boxtimes) = \{\langle x, y\rangle : x = y \textit{ oder es existiert } z : x < z > y\}\,.$$

Man mache sich klar, daß dies in Bäumen immer gilt, und daß umgekehrt aus den obigen Axiomen folgt, daß $R(\diamond\!\!\!\!\cdot^+)$ ein größtes Element besitzen muß.

Nun setzen wir:

$$b(\varphi) := \varphi \wedge \boxplus\bot \wedge [\neq]\neg\varphi$$

$b(\varphi)$ ist genau dann an einem Knoten x erfüllt, wenn x ein Blatt ist und φ genau bei x gilt. Nun können wir axiomatisch erfassen, daß $R(\diamond\!\!\!\!\cdot^+)$ auf Blättern linear sein muß.

$$\boxplus\bot \wedge \langle\neq\rangle b(q). \rightarrow .p \vee \diamond\!\!\!\!\cdot^+ p \vee \diamond\!\!\!\!\cdot^+ p$$

Schließlich muß man noch Axiome aufnehmen, die die Verteilung der Marken regulieren. Dies überlassen wir dem Leser.

Proposition 5.4.3 *Die Klasse der erschöpfend geordneten Bäume ist endlich* $\boldsymbol{QML}^b$*–axiomatisierbar.*

Wir wissen schon, daß wir $\mathbf{QML}^b$ in $\mathbf{MSO}^b$ einbetten können. Die Umkehrung ist wie immer die Schwierigkeit. Dazu gehen wir wie im Zeichenkettenfall vor. Wir führen das Analogon der beschränkten Quantoren ein. Wir definieren Funktionen $\Diamond$, $\Diamond^+$, $\Diamond$, $\Diamond^+$, $\Diamond$, $\Diamond^+$, $\Diamond$, $\Diamond^+$, sowie $\langle\neq\rangle$ auf einstelligen Prädikaten, deren Bedeutung selbsterklärend sein dürfte. Zum Beispiel ist

$$\begin{array}{lcl}(\Diamond\varphi)(x) & := & (\exists y \succ x)\varphi(y)\ ,\\ (\Diamond^+\varphi)(x) & := & (\exists y > x)\varphi(y)\ ,\end{array}$$

wobei $y \notin fv(\varphi)$. Schließlich sei O definiert durch

$$O(\varphi) := (\forall x)\neg\varphi(x)\ .$$

Also besagt $O(\varphi)$ nichts anderes, als daß $\varphi(x)$ nirgends erfüllt ist. Es sei P_x eine Prädikatvariable, die nicht in φ vorkommt. Definiere dann $\{P_x/x\}\varphi$ induktiv wie in Abschnitt 5.2 beschrieben. Sei $\gamma(P_x) = \{\beta(x)\}$. Dann gilt

$$\langle\mathfrak{M},\gamma,\beta\rangle \models \varphi \text{ gdw. } \langle\mathfrak{M},\gamma,\beta\rangle \models \{P_x/x\}\varphi$$

Setze daher

$$(Ex)\varphi(x) := (\exists P_x)(\neg O(P_x) \wedge O(P_x \wedge \langle\neq\rangle P_x).\ \rightarrow .\{P_x/x\}\varphi)$$

Zu dieser Definition gibt es lediglich Folgendes zu bemerken. Aufgrund dieses Sachverhalts ist für alle erschöpfend geordneten Bäume $\mathfrak{B}$

$$\langle\mathfrak{B},\gamma,\beta\rangle \models (\exists x)\varphi \text{ gdw. } \langle\mathfrak{B},\gamma,\beta\rangle \models (Ex)\varphi$$

Es sei wieder $h : P \to PV$ eine Bijektion von den Prädikatvariablen von $\mathbf{MSO}^b$ auf die Aussagevariable von $\mathbf{QML}^b$.

$$\begin{array}{llll}(\underline{a}(x))^\diamond & := & Q^a & (P(y))^\diamond := h(P)\\ (\Diamond(\varphi))^\diamond & := & \Diamond\varphi^\diamond & (\Diamond\varphi)^\diamond := \Diamond\varphi^\diamond\\ (\Diamond\varphi)^\diamond & := & \Diamond\varphi^\diamond & (\Diamond\varphi)^\diamond := \Diamond\varphi^\diamond\\ (\Diamond^+\varphi)^\diamond & := & \Diamond^+\varphi^\diamond & (\Diamond^+\varphi)^\diamond := \Diamond^+\varphi^\diamond\\ (\Diamond^+\varphi)^\diamond & := & \Diamond^+\varphi^\diamond & (\Diamond^+\varphi))^\diamond := \Diamond^+\varphi^\diamond\\ (\neg\varphi)^\diamond & := & \neg\varphi^\diamond & (O(\varphi))^\diamond := \boxtimes\neg\varphi^\diamond\\ (\varphi_1 \wedge \varphi_2)^\diamond & := & \varphi_1^\diamond \wedge \varphi_2^\diamond & (\varphi_1 \vee \varphi_2)^\diamond := \varphi_1^\diamond \vee \varphi_2^\diamond\\ ((\exists P)\varphi)^\diamond & := & (\exists h(P))\varphi^\diamond & ((\forall P)\varphi)^\diamond := (\forall h(P))\varphi^\diamond\end{array}$$

Damit ist nun die angestrebte Einbettung von $\mathbf{MSO}^b$ nach $\mathbf{QML}^b$ gezeigt.

Theorem 5.4.4 *Es sei φ eine **MSO**^{b}–Formel mit höchstens einer freien Variablen, die Objektvariable x_0. Dann existiert eine **QML**^{b}–Formel φ^M dergestalt, daß für alle erschöpfend geordneten Bäume $\mathfrak{B}$ gilt:*

$$\langle \mathfrak{B}, \beta \rangle \models \varphi(x_0) \text{ gdw. } \langle M(\mathfrak{B}), \beta(x_0) \rangle \models \varphi(x_0)^M$$

Korollar 5.4.5 *Modulo der Identifikation $\mathfrak{B} \mapsto M(\mathfrak{B})$ definieren **MSO**^{b} und **QML**^{b} dieselben Modellklassen von erschöpfend geordneten endlichen Bäumen. Ferner: genau dann ist $\mathcal{K}$ eine endlich axiomatisierbare Klasse von **MSO**^{b}–Strukturen, wenn $M(\mathcal{K})$ eine endlich axiomatisierbare Klasse von **QML**^{b}–Strukturen ist.*

Für den Zweck der Definition von Code heben wir den Unterschied zwischen terminalen und nichtterminalen Zeichen auf.

Definition 5.4.6 *Es sei $G = \langle \Sigma, N, A, R \rangle$ eine kontextfreie Grammatik* und $\varphi \in$ **QML**^{b} eine konstante Formel (mit Konstanten in A). Wir sagen, G sei **treu für** φ, falls es eine Teilmenge $H_\varphi \subseteq N$ derart gibt, daß für jede jeden Baum $\mathfrak{B}$ und jeden Knoten $w \in B$ gilt: genau dann ist $\langle \mathfrak{B}, w \rangle \models \varphi$, wenn $\ell(w) \in H_\varphi$. Wir sagen auch, H_φ **codiert** φ **bezüglich** G. Es sei φ eine **QML**^{b}–Formel und n eine natürliche Zahl. Ein n–**Code** für φ ist ein Paar $\langle G, H \rangle$, wo $L_B(G)$ die Menge aller höchstens n–fach verzweigenden, endlichen, erschöpfend geordneten Bäume über $A \cup N$ ist und H φ in G codiert. φ heißt n–**codierbar**, falls es einen n–Code für φ gibt. φ heißt **codierbar**, falls es für jedes n einen n–Code für φ gibt.*

Man beachte, daß wir uns aus technischen Gründen auf höchstens n–verzweigende Bäume beschränken müssen, da wir ja sonst gar keine kontextfreie Grammatik* als Code hinschreiben können. Es seien $G = \langle \Sigma, N, A, R \rangle$ und $G' = \langle \Sigma', N', A, R' \rangle$ Grammatiken* über A. Der Einfachheit halber nehmen wir an, sie seien in Standardform. Das **Produkt** ist dann definiert als die Grammatik*

$$G \times G' = \langle \Sigma \times \Sigma', N \times N', A, R \times R' \rangle$$

wo

$$\begin{aligned} R \times R' := \quad & \{\langle X, X' \rangle \to \langle Y_0, Y_0' \rangle \dots \langle Y_{n-1}, Y_{n-1}' \rangle : \\ & \qquad X \to Y_0 \dots Y_{n-1} \in R, X' \to Y_0' \dots Y_{n-1}' \in R'\} \\ \cup \quad & \{\langle X, X' \rangle \to a : X \to a \in R, X' \to a \in R'\} \end{aligned}$$

Um das Analogon des Codierungstheorems für Zeichenketten zu beweisen, müssen wir noch einen Kunstgriff vornehmen. Wie man leicht nachweisen kann, ist die direkte Erweiterung auf Bäume deshalb falsch, weil wir jetzt auch die Nichtterminalsymbole explizit aufgenommen haben. Deswegen machen wir es wie folgt. Es sei $h : N \to N'$ eine Abbildung und $\mathbb{G} = \langle B, <, \sqsubset, \ell \rangle$ ein Baum mit Marken in $B \cup N$. Dann sei $h[\mathfrak{B}] := \langle B, <, \sqsubset, h_A \circ \ell \rangle$, wobei $h_A \restriction N = h$ und $h_A(a) = a$ für alle $a \in A$. Dann heißt $h[\mathfrak{B}]$ eine **Projektion** von $\mathfrak{B}$. Ist $\mathcal{K}$ eine Klasse von Bäumen, so sei $h[\mathcal{K}] := \{h[\mathfrak{B}] : \mathfrak{B} \in \mathcal{K}\}$. Das Theorem lautet zuerst einmal so.

Theorem 5.4.7 (Thatcher & Wright, Doner) *Es sei $n \in \omega$, A ein Terminalalphabet und N ein Nichtterminalalphabet. Genau dann ist eine Klasse von erschöpfend geordneten, höchstens n–verzweigenden, endlichen Bäumen über $A \cup N$ endlich axiomatisierbar in $\mathbf{MSO}^b$, wenn sie die Projektion auf $A \cup N$ einer kontextfreien* Klasse von Bäumen ist.*

Hierbei heißt eine Klasse **kontextfrei***, wenn sie die Klasse von Bäumen einer kontextfreien Grammatik* ist. Man beachte, daß das Symbol ε nicht wie bei den regulären Sprachen Probleme macht. Es ist nunmehr ein eigenständiges Symbol, welches als Marke auf einem Blatt auftauchen kann. Insofern muß hier zwischen dem Alphabet A und dem Alphabet A_ε unterscheden werden. Man beachte, daß eine Vereinigung zweier kontextfreier Klassen nicht notwendig wieder kontextfrei ist. (Bei regulären Sprachen war es anders, weil wir dort nur von Terminalzeichenketten geredet haben, nicht von Ableitungen.)

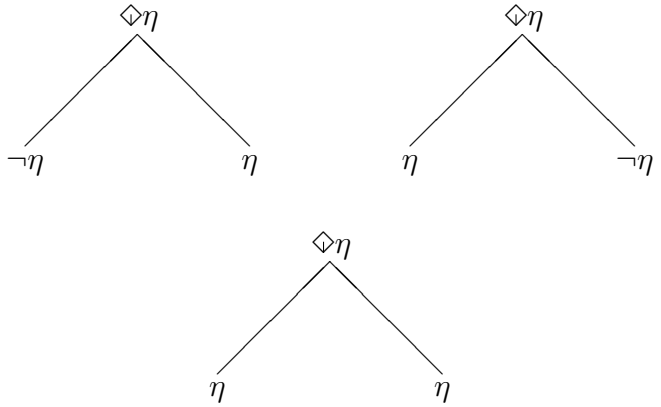
Von nun ab läuft das Verfahren wieder analog. Zunächst beweist man die Codierbarkeit von $\mathbf{QML}^b$–Formeln. Anschließend überlegt man wie folgt. Es sei $\langle G, H\rangle$ der Code einer Formel φ. Wir schränken die Menge der Symbole (also sowohl N als auch A) auf H ein. Dadurch bekommen wir eine Grammatik*, welche nur Bäume erzeugt, welche φ erfüllen. Schließlich definieren wir die Projektion $h : H \to A \cup N$ wie folgt. Ist $h(a) := a$, $a \in A$, und $h(Y) = X$, falls gilt $L_B(G) \models (\forall x)(\underline{Y}(x) \to \underline{X}(x))$. Damit dies wohldefiniert ist, muß folglich für alle $Y \in H$ ein $X \in N$ existieren mit dieser Eigenschaft. Wir nennen den Code in diesem Fall **uniform**. Die uniforme Codierbarkeit wird ganz leicht aus der Codierbarkeit folgen, da wir stets Produkte $G \times G'$ von Grammatiken* bekommen, sodaß $G = \langle \Sigma, N, A, R\rangle$ und $L_B(G \times G') \models \underline{X}(\langle x, y\rangle)$ genau dann, wenn $L_B(G) \models \underline{X}(x)$. Die Abbildung h ist dann nichts weiter als die Projektion auf die erste Komponente.

Theorem 5.4.8 *Jede konstante $\mathbf{QML}^b$–Formel ist uniform codierbar.*

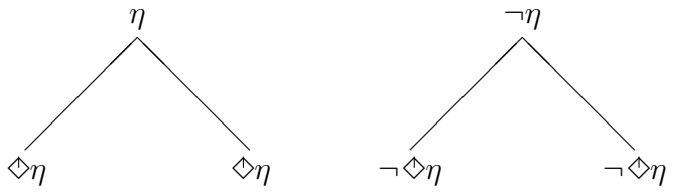
Beweis. Wir liefern hier nur noch eine Skizze ab. Wir wählen zunächst ein n und zeigen von nun ab die uniforme n–Codierbarkeit. Der Bequemlicheit halber illustrieren wir alles für $n = 2$. Für die Formeln $\underline{a}(x)$, $a \in A$, oder $\underline{Y}(x)$, $Y \in N$, ist nichts Besonderes zu tun. Ebenso leicht sind wiederum die Fälle der booleschen Junktoren. Es bleiben wiederum die acht Modaloperatoren sowie die Quantoren. Bevor wir damit beginnen, wollen wir eine etwas bequemere Notation einführen. Wie immer gehen wir davon aus, daß wir eine Grammatik* $G = \langle \Sigma, N, A, R\rangle$ vorliegen haben sowie gewisse Mengen H_η für gewissen Formeln. Nun nehmen wir das Produkt mit einer neuen Grammatik* und definieren H_φ. Anstelle von expliziten Marken verwenden wir nun die Formeln selbst, wobei η für eine beliebige Marke aus H_η steht.

Die einfachen Modalitäten sind wie folgt. Wir nehmen $\mathbf{2} = \langle \{0, 1\}, \{0, 1\}, A, R_2\rangle$, wobei R_2 aus allen möglichen höchstens n–verzweigenden Regeln einer Grammatik

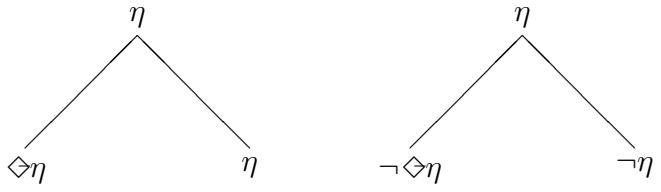
in Standardform besteht. Jetzt bilden wir das Produkt dieser zwei Grammatiken*; allerdings wählen wir eine Teilmenge der Regeln und der Startsymbole. $\Sigma' := \Sigma \times \{0,1\}$. Es sei $H'_\eta := H_\eta \times \{0,1\}$, $H'_{\Diamond\eta} := N \times \{1\}$. Die Regeln sind nun alle von der Form



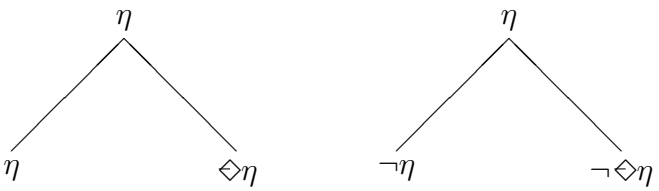
Nun gehen wir zu $\Diamond\eta$ über. Hier ist $\Sigma'_{\Diamond\eta} := N \times \{0\}$.



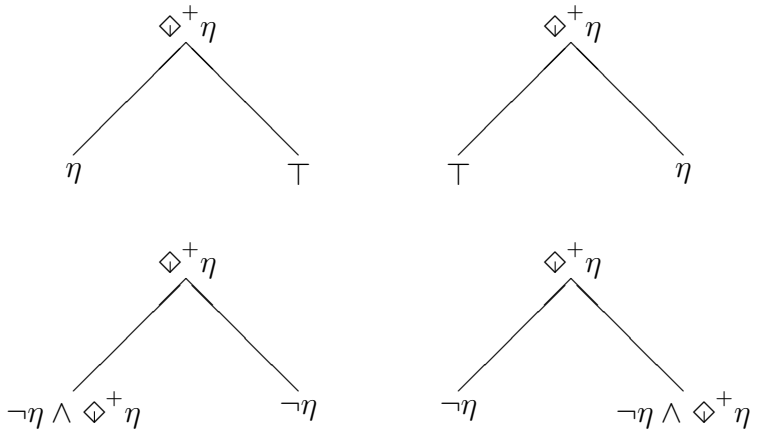
Bei $\Diamond\eta$ ist $\Sigma'_{\Diamond\eta} = \Sigma \times \{0\}$.



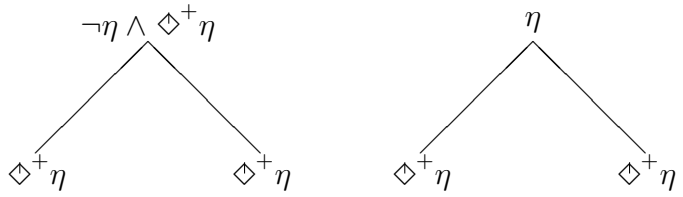
Ebenso ist $\Sigma'_{\Diamond\eta}$ das Startsymbol von G' im Falle von $\Diamond\eta$.



Betrachten wir nun die Relation $\Diamond^+\eta$.



Die Startsymbolmenge ist $\Sigma \times \{0, 1\}$.



Die Startsymbolmenge ist $\Sigma' := \Sigma \times \{0\}$.

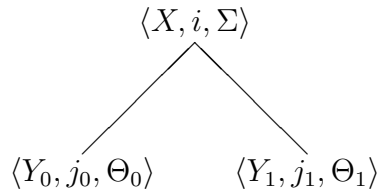
Als Letztes studieren wir den Quantor $(\exists p)\eta$. Es sei $\eta' := \eta[\underline{c}/p]$, wo $\underline{c}$ eine neue Konstante ist. Unser Terminalalphabet sei also $A \times \{0, 1\}$, das Nichtterminalalphabet $N \times \{0, 1\}$. Wir nehmen an, $\langle G^1, H^1_\theta \rangle$ sei ein uniformer Code für θ, θ eine beliebige Teilformel von η'. Für jede Teilmenge Σ der Menge Δ aller Teilformeln von η' setzen wir

$$L_\Sigma := \bigwedge_{\theta \in \Sigma} \theta \wedge \bigwedge_{\theta \notin \Sigma} \neg\theta$$

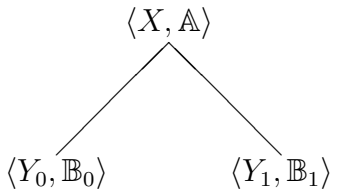
Dann ist $\langle G^1, H^1_\Sigma\rangle$ ein Code für L_Σ, wenn

$$H^1_\Sigma := \bigcap_{\theta\in\Sigma} H^1_\theta \cap \bigcap_{\theta\notin\Sigma}(N - H^1_\theta)$$

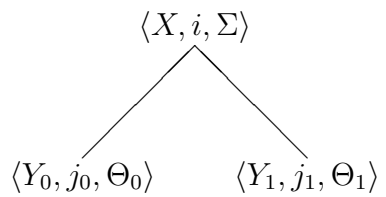
Nun wählen wir eine neue Grammatik, G^2. Dazu sei $N^2 := N \times \{0,1\} \times \wp(N^1)$. Als Regeln nehmen wir alle Regeln der Form



wo $\langle X, i\rangle \in H^1_\Sigma$, $\langle Y_0, j_0\rangle \in H^1_{\Theta_0}$, $\langle Y_1, j_1\rangle \in H^1_{\Theta_1}$ und $\Sigma \to \Theta_0\Theta_1$ eine Regel von G^1 ist. (Dies ist wiederum der Fall, wenn es X, Y_0 und Y_1 sowie i, j_0 und j_1 gibt derart, daß $\langle X, i\rangle \to \langle Y_0, j_0\rangle\ \langle Y_1, j_1\rangle \in R$.) Ebenso für einstellige Regeln. Jetzt gehen wir als Letztes zu G^3 über, mit $N^3 := N \times \wp(\wp(N^1))$. Hier nehmen wir alle Regeln der Form



wo $\mathbb{A}$ die Menge aller Σ ist, für die Θ_0, Θ_1 existieren und i, j_0, j_1 derart, daß



eine Regel von G^2 ist. $\dashv$

Übung 118. Zeigen Sie: $<$ ist aus $\prec$ definierbar, ebenso $>$. Man kann also Bäume alternativ auch mittels $\prec$ (oder $\succ$) axiomatisieren. Zeigen Sie ferner: in geordneten Bäumen ist $\prec$ eindeutig durch $<$ bestimmt. Geben Sie eine explizite Definition an.

Übung 119. Es sei $x\,L\,y$, falls x und y Schwestern sind und $x \sqsubset y$. Zeigen Sie: in geordneten Bäumen ist L mittels $\sqsubset$ definierbar und umgekehrt.

5.5 Eine Detailstudie

Wir wollen in diesem Anschnitt einiges zu verschiedenen syntaktischen Theorien und die Art und Weise sagen, wie sie gewisse sprachliche Phänomene behandeln. Schauen wir uns folgendes einfaches Phänomen an, welches **Topikalisierung** heißt.

(5.1) `Harry` `likes` `trains.`
Harry mag-3.Sg Zug-Pl
Harry mag Züge.

(5.2) `Trains,` `Harry` `likes.`
Zug-Pl Harry mag-3.Sg
Züge mag Harry.

Wir haben zwei verschiedene englische Sätze, von denen der erste in der Normalreihenfolge, SVO, der andere in der Reihenfolge OSV steht. Man sagt, das Objekt wurde im zeiten Satz topikalisiert. Die beiden Sätze sind wenigstens pragmatisch verschieden, und aller Wahrscheinlichkeit nach auch semantisch. Mit unseren bisherigen Instrumenten können wir den Unterschied zwischen ihnen allerdings nicht greifen, und es wird für die Diskussion dieses Abschnitts unerheblich sein, ob es ihn gibt oder nicht.

Die Transformationsgrammatik nimmt an, (5.2) ist aus (5.1) entstanden. Verantwortlich dafür ist eine Transformation, welche das Objekt eines Satzes an das linke Ende verschiebt. Mit der Erfindung der Spuren wurder ferner angenommen, daß als Ergebnis dieser Transformation an der ursprünglichen Stelle nunmehr eine sogenannte Spur steht, welche man t schreibt, und diese einen Index trägt, den gleichen, wie das Element, das bewegt worden ist. Eigentlich sieht (5.2) deswegen so aus.

(5.3) `Trains`$_1$`, Harry likes` t_1.

Wir haben hier den Index 1 gewählt, aber jeder andere wäre auch möglich gewesen. Die Indizes wie auch das Element t sind nicht hörbar, und werden auch nicht geschrieben. Man beachte, daß die Transformationsgrammatik also einen mehrschichtigen Erzeugungsprozeß für Sätze annimmt. Zunächst stellt sie sogenannte Tiefenstrukturen bereit, welche durch eine kontextfreie Grammatik erzeugt werden, und anschließend kreiert sie mittels Transformationen die Oberflächenstrukturen, welche allerdings noch leere Symbole enthalten, die anschließend getilgt werden müssen.

Das Äquivalent der Transformationen in GPSG sind die Metaregeln, auf die wir weiter unten eingehen werden. Dies sind Regeln, welche aus einer (kontextfreien) Regel wieder eine (kontextfreie) Regel machen. Es gibt nun eine Regel **TOP**, welche

den Satzanfang einer topikalisierten Struktur beschreibt.

$$\left[\begin{array}{lcl} \text{KAT} & : & np \end{array}\right] \rightarrow \left[\begin{array}{lcl} \text{KAT} & : & s \\ \text{SLASH} & : & np \end{array}\right]$$

Der spezielle Trick in GPSG ist die Einführung des Merkmals SLASH. Es ist ein Buchführungsinstrument, welches uns erlaubt, die Position zu verfolgen, von welcher dieses Element herkommt, um zu wissen, wo etwas ausgelassen werden muß. In der GPSG Terminologie spricht man von der topikalisierten Nominalphrase als dem **Füller**, und von der Stelle, wo sie herkommt, als **Lücke**. Nun fügen wir noch zwei Regeln hinzu. Die erste ist die Folgende.

$$\left[\begin{array}{lcl} \text{KAT} & : & s \\ \text{SLASH} & : & np \end{array}\right] \rightarrow \left[\begin{array}{lcl} \text{KAT} & : & np \end{array}\right] \qquad \left[\begin{array}{lcl} \text{KAT} & : & vp \\ \text{SLASH} & : & np \end{array}\right]$$

Sie sorgt dafür, daß das Subjekt nicht als Lücke genommen wird sondern daß die Lücke im Verb gesucht wird. Die zweite Regel ist diese.

$$\left[\begin{array}{lcl} \text{KAT} & : & vp \\ \text{sc slash} & : & np \end{array}\right] \rightarrow \left[\begin{array}{lcl} \text{KAT} & : & v \end{array}\right]$$

Da wir Verben im Moment grundsätzlich als transitive Verben behandeln, sagt diese Regel nun aus, daß die Phrase eines transitiven Verbs, welche eine NP–Lücke trägt, durch diese Verb ohne sein Objekt realisiert wird. Der Leser kann sich leicht davon überzeugen, daß dies genau den Satz (5.2) erzeugt.

Im Grunde genommen kann man es dabei bewenden lassen, die Regeln konkret hinzuschreiben, aber GPSG geht hier einen Schritt weiter. Es wird eine Reihe von Metaregel postuliert, welche aus einer kontextfreien Regelmenge eine größere Menge machen. Zum Beispiel kann man eine Reihe von Metaregeln schreiben, welche die Einführung und Verwaltung von SLASH–Merkmalen regulieren. Zum Beispiel die Folgende Metaregel, die SLASH–Terminierungsregel.

$$X \rightarrow YZ \Longrightarrow X[\text{SLASH} : Z] \rightarrow Y$$

Diese besagt das Folgende. Ist $X \rightarrow YZ$ eine Regel, so ist auch $X[\text{SLASH} : Z] \rightarrow Y$ eine Regel, wo $X[\text{SLASH} : Z]$ die Kategorie X vermehrt um das Merkmal [SLASH : Z] ist. Damit dies nicht zu unerwünschten Resultaten führt, muß man bei dieser Metaregel voraussetzen, daß X auf der linken Seite das Merkmal $\neg[\text{SLASH} : \top]$ trägt.

5.6 Multiple Dominanz

Literaturverzeichnis

[1] Alexander Aigner. *Zahlentheorie*. de Gruyter, Berlin.

[2] Martin Aigner. *Diskrete Mathematik*. Vieweg Verlag, Braunschweig/Wiesbaden, 1993.

[3] Patrick Blackburn. Modal logic and attribute value structures. In Maarten de Rijke, editor, *Diamonds and Defaults*, Synthese No. 229, pages 19 –65. Kluwer, 1993.

[4] J. Büchi. Weak second–order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66 – 92, 1960.

[5] Hadumod Bußmann. *Lexikon der Sprachwissenschaft*. Number 452 in Kröners Taschenausgabe. Kröner, Stuttgart, 2 edition, 1990.

[6] Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science 32. Cambridge University Press, 1992.

[7] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28:114 – 133, 1981.

[8] Noam Chomsky. Context–free grammars and pushdown storage. *MIT Research Laboratory of Electronics Quarterly Progress Report*, 65, 1962.

[9] J. E. Doner. Tree acceptors and some of their applications. *Journal of Computer and Systems Sciences*, 4:406 – 451, 1970.

[10] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics*. Number 11 in Synthese Library. Reidel, Dordrecht, 1981.

[11] Kit Fine. Transparency, Part I: Reduction. Unveröffentlichtes Manuskript, UCLA, 1992.

[12] L. T. F. Gamut. *Logic, Language and Meaning*, volume 2: Intensional Logic and Logical Grammar. University of Chicago Press, Chicago, 1991.

[13] L. T. F. Gamut. *Logic, Language and Meaning*, volume 1: Introduction to Logic. University of Chicago Press, Chicago, 1991.

[14] G. Gazdar, G. Pullum, R. Carpenter, T. Hukari, and R. Levine. Category structures. *Computational Linguistics*, 14:1 – 19, 1988.

[15] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Blackwell, 1985, 1985.

[16] Peter Geach. A Program for Syntax. In Donald Davidson and Gilbert Harman, editors, *Semantics for Natural Language*, number 40 in Synthese Library. Reidel, Dordrecht, 1972.

[17] M. M. Geller and M. A. Harrison. On $LR(k)$ grammars and languages. *Theoretical Computer Science*, 4:245 – 276, 1977.

[18] Seymour Ginsburg. *Algebraic and Automata–Theoretic Properties of Formal Languages*. North–Holland, Amsterdam, 1975.

[19] S. A. Greibach. A new normal–formal theroem for context–free phrase structure grammars. *Journal of the Association for Computing Machinery*, 13:42 – 52, 1967.

[20] Günter Grewendorf, Fritz Hamm, and Wolfgang Sternefeld. *Sprachliches Wissen. Eine Einführung in moderne Theorien der grammatischen Beschreibung*. Number 695 in suhrkamp taschenbuch wissenschaft. Suhrkamp Verlag, 1987.

[21] Annius Groenink. *Surface without Structure. Word Order and Tractability Issues in Natural Language Analysis*. PhD thesis, University of Utrecht, 1997.

[22] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, Reading (Mass.), 1978.

[23] Harro Heuser. *Lehrbuch der Analysis. Teil 1*. Teubner Verlag, Stuttgart, 11 edition, 1994.

[24] J. R. Hindley, B. Lercher, and J. P. Seldin. *Introduction to Combinatory Logic*. Number 7 in London Mathematical Society Lecture Notes. Oxford University Press, Oxford, 1972.

[25] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata*. Addison Wesley, Reading (Mass.), 1969.

[26] R. Huybregts. Overlapping Dependencies in Dutch. *Utrecht Working Papers in Linguistics*, 1:3 – 40, 1984.

[27] Mark Johnson. *Attribute-Value Logic and the Theory of Grammar*, volume 16 of *CSLI Lecture Notes*. CSLI, 1988.

[28] Aravind K. Joshi. How much context–sensitivity is needed for characterizing structural descriptions: Tree adjoining grammars. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*. Cambdrige University Press, Cambridge, 1985.

[29] T. Kasami. An efficient recognition and syntax–analysis algorithm for context–free languages. Technical report, Air Force Cambridge Research Laboratory, Bedford, Mass., 1965. Science Report AFCRL–65–758.

[30] Steven C. Kleene. Representation of events in nerve nets. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3 – 40. Princeton University Press, 1956.

[31] Donald Knuth. On the translation of languages from left to right. *Information and Control*, 8:607 – 639, 1965.

[32] Marcus Kracht. Is there a genuine modal perspective on feature structures? *Linguistics and Philosophy*, 18:401 – 458, 1995.

[33] Joachim Lambek. The Mathematics of Sentence Structure. *The American Mathematical Monthly*, 65:154 – 169, 1958.

[34] Th. Lewandowski. *Linguistisches Wörterbuch*. Number 200, 201, 300 in UTB. Quelle & Meyer Verlag, Heidelberg und Wiesbaden, 4 edition, 1985.

[35] John Lyons. *Einführung in die moderne Linguistik*. Beck'sche Elementarbücher. Beck Verlag, München, 1968.

[36] Alexis Manaster-Rames and Michael B. Kac. The Concept of Phrase Structure. *Kinguistics and Philosophy*, 13:325 – 362, 1990.

[37] Rohit Parikh. Language generating devices. *MIT Research Laboratory of Electronics Quarterly Progress Report*, 60:199 – 212, 1961.

[38] Mati Pentus. Models for the Lambek calculus. *Annals of Pure and Applied Logic*, 75:179 – 213, 1995.

[39] Mati Pentus. Product–Free Lambek–Calculus and Context–Free Grammars. *Journal of Symbolic Logic*, 62:648 – 660, 1997.

[40] Carl Pollard and Ivan Sag. *Head–Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, 1994.

[41] Carl J. Pollard. *Generalized Phrase Structure Grammar, Head Grammars and Natural Language*. PhD thesis, Stanford University, 1984.

[42] Daniel Radzinski. Unbounded Syntactic Copying in Mandarin Chinese. *Linguistics and Philosophy*, 13:113 – 127, 1990.

[43] William C. Rounds. LFP: A Logic for Linguistic Description and an Analysis of its Complexity. *Computational Linguistics*, 14:1 – 9, 1988.

[44] Arto K. Salomaa. *Formale Sprachen*. Springer Verlag, Berlin, 1978.

[45] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context–free grammars. *Theoretical Computer Science*, 88:191 – 229, 1991.

[46] A. Sestier. Contributions à une théorie ensembliste des classifications linguistiques. In *Actes du Ier Congrès de l'AFCAL*, pages 293 – 305, Grenoble, 1960.

[47] Stuart Shieber. Evidence against the Context–Freeness of Natural Languages. *Linguistics and Philosophy*, 8:333 – 343, 1985.

[48] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second–order logic. *Mathematical Systems Theory*, 2:57 – 81, 1968.

[49] K. Vijay-Shanker, David J. Weir, and Aravind Joshi. Characterizing structural descriptions produced by various grammar formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics*, pages 104 – 111, 1987.

[50] Arnim von Stechow and Wolfgang Sternefeld. *Bausteine syntaktischen Wissens. Ein Lehrbuch der generativen Grammatik*. Westdeutscher Verlag, Opladen, 1987.

[51] Christian Wartena. *Storage Structures and Conditions on Movement in Natural Language Syntax*. PhD thesis, Institut für allgemeine Sprachwissenschaft, Universität Potsdam, 1999.

[52] D. H. Younger. Recognition and parsing of context–free languages in time n^3. *Information and Control*, 10:189 – 208, 1967.

[53] Wlodek Zadrozny. From Compositional Semantics to Systematic Semantics. *Linguistics and Philosophy*, 17:329 – 342, 1994.