

Chapter 1

Discovering a new class of languages

SEAN A. FULOP

ABSTRACT. This paper outlines a new approach to specifying formal constraints on grammars and their languages, in pursuit of more far-reaching statements about possible human languages than, e.g., “they are context-free.” We propose that possible human languages, conceived as term-labeled tree languages consisting of syntactic word trees annotated by semantic lambda terms, can be said to be *syntactically homogeneous* and *finitely illustratable*. These are new properties of term-labeled tree languages which, when used to specify a subset of all languages generated by a large class of multimodal type-logical grammars, yield a new class of languages which cross-cuts the traditional Chomsky hierarchy. A discovery procedure for type-logical lexicons is then outlined, whose range is proven to generate precisely these languages. Our new approach is actually quite old, having roots in ideas of the American Structuralist school.

1.1 Introduction

We set out to motivate and define a new class of formal languages that is not directly related to the Chomsky hierarchy, and which in fact completely cross-cuts it. This effort answers to a complaint that resonates through modern theoretical syntax, to the effect that the linguistic interest of the Chomsky hierarchy has long ago expired because the formal constraints on languages that it provides are not very useful for considering what possible human languages might be like. So when we consider, e.g., whether “human languages are context-free,” we really mean the upper bound on the language complexity. We never ask “are all context-free languages possibly human,” since we all know the answer: “of course not.”

Our new class consists not of string languages, but of *term-labeled tree* languages, where a term-labeled tree is an unlabeled syntactic structure paired with a

⁰Thanks to Jerry Sadock, Gerhard Jäger, and Jason Merchant for discussions that stimulated the development of this work. Thanks to the March 2002 colloquium audience at The University of Chicago for receiving the initial ideas. Thanks as always to Ed Stabler, for beginnings. This work was supported in part by a Research Incentive grant from the University of Chicago Dept. of Computer Science.

compositional semantic term (a lambda term). The class is defined with reference to a new formalization of the ideas of the American Structuralist school which permits the rigorous statement of conditions on the syntactic (and semantic) distribution of linguistic elements in a term-labeled tree (TLT) language. We call our new class of such languages *syntactically homogeneous*, and give a strict definition in terms of the structural conditions that must be met. Some of these languages have the further property of being *finitely illustratable*, forming an important subclass.

We next discuss a new algorithm which can discover lexicons for type-logical grammar (i.e. assignments of sets of syntactic types/categories to words) given a type logic (drawn from a broad but strictly defined class of “permitted” logics) and a sample of TLTs (or strings, increasing the computational complexity immensely) *which do not show their types*. The method is based on earlier work in this vein (Fulop, 2003) which discovered so-called *optimally unified* lexicons under the same learning conditions. Upon realizing that the class of optimally unified grammars that resulted from that work automatically adhered to certain Structuralist-style conditions on the distribution of linguistic elements, efforts were undertaken to improve the distributional analysis performed by the procedure beyond the naive one produced by optimally unifying the lexicon. The end product is a new, but similar, discovery procedure that outputs type-logical lexicons which we dub *structurally unified*. The TLT languages of such grammars, it turns out, are always syntactically homogeneous, and finitely illustratable. In fact, we demonstrate that the total range of the procedure, over all permitted type logics, is precisely a class of type-logical grammars which generate all and only the finitely illustratable syntactically homogeneous term-labeled tree languages generated by the permitted logics.

The mathematical interest of the preceding is, we hope, self-evident, but what about linguistics? It is conjectured that the condition of a TLT language’s being at once syntactically homogeneous and finitely illustratable is sufficiently strong to characterize possible human languages fairly tightly. That is to say, given a reasonable vocabulary of human language words and/or morphemes, any such language meeting a couple of other simple conditions “could reasonably be” human. Syntactic homogeneity of a language is thus offered as a precise property capturing the primary essence of a language’s “being human,” which is surely more than could ever be said of context-freeness and the like.

1.2 Term-labeled tree languages

A type-logical grammar $G = \langle V_G, I_G, R_G \rangle$ consists of a vocabulary V_G , a lexical function I_G assigning sets of categories/types to words, and a type logic R_G which treats the categories as formulae. Owing to space constraints, we forego the de-

tails of type logic (v. Fulop 2002b; Moortgat 1997). The type logics and grammars that concern us are all based upon the nonassociative system of Lambek (1961) called ‘NL,’ whose formulae involve the two logical operators $/, \backslash$ which we call “slashes.” The logics may be enriched with the addition of multiple indexed families of slashes, as well as unary modalities (Morrill, 1994) that license structural rules for rearranging the syntactic elements. We work with the logics in Gentzen’s sequent formulation (Gentzen, 1934).

Given such a general landscape, we do have to limit ourselves to certain regions. It is not possible to prove anything worthwhile about a class of “all modally enriched type logics,” and it is not even clear what this would mean because it imposes no restrictions at all on the nature of structural rules. It is clear that unrestricted logics of this kind can be created which imbue grammars based upon them with Turing-complete generating capacity (Carpenter, 1999), and this is not only undesirable, it makes the enterprise of proving a single sentence undecidable! On the other hand, many type logics have pleasant properties for sentence deduction; NL, for instance, enjoys Cut-elimination and has the subformula property, and its decidability is guaranteed as well.

We accordingly limit our further considerations to those classes \mathcal{L}_k of type logics R_G adhering to the following restrictions, which have been introduced and motivated elsewhere (Fulop, 2002b):

1. R_G possesses just a finite number of families of slashes, a finite number of families of unary modalities, and a finite number of structural rules which together preserve Cut-elimination and the subformula property of the base logic NL.
2. All structural rules are applicable only in the presence of certain unary modalities.
3. Each class \mathcal{L}_k of type logics consists of all logics satisfying the above, and whose well-formed formulae are restricted to those in which each subformula has at most k unary modal operators on its left edge.

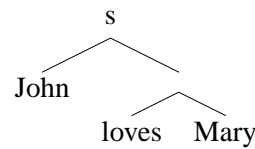
Suffice it to say that a class \mathcal{L}_k with k “sufficiently large” (e.g. $k = 2$) includes logics with sufficient generating capacity to handle the kind of context-sensitivity occasionally demanded by natural languages, but any logic in the class remains decidable. Our results in this paper will generally be about the union of all such classes, denoted $\bigcup_k \mathcal{L}_k$; this large class comprises all of the candidates for the Universal Type Logic in this setting, and we presume that some such logic could generate every possible human language, with only the lexicons varying. The precise weak generating capacity of grammars based on these logics in terms of the Chomsky hierarchy is in general not particularly well-understood. It has been established that the base logic NL yields grammars of context-free languages exactly, while the addition of modals together with certain sets of modally licensed structural rules (frequently called “interaction postulates”) yields some context-sensitivity. For current results along these lines, see Jäger (2003, 2002).

A key aspect of our learning algorithm is the connection between the syntactic structure of sentences, expressed as a tree of Lambek formulae, and the semantic structure, expressed as a lambda term. Any type logic $R_G \in \bigcup_k \mathcal{L}_k$ can be shown to induce a fragment Λ_{NL} of the simply typed lambda calculus such that for every sequent $\Gamma \Rightarrow s$ in R_G provable by some proof Π there will be a term $N^s \in \Lambda_{NL}$ such that N is a *homomorphic construction* of the sequent proof Π . Let Λ_{NL} denote the largest sublanguage of the simply typed lambda language each of whose terms N conforms to the following constraints: 1. Term N may not contain vacuous abstraction; 2. No subterms of N are without free variables or constants; 3. No subterm of N contains more than one free occurrence of the same variable; 4. Every prefix λx in N binds exactly one free variable occurrence, and that occurrence is either leftmost or rightmost in the subterm abstracted over by the prefix.

The facts mentioned above are proven elsewhere (Fulop, 2002a,b), and the idea of a homomorphic construction has been formalized in those references. Roughly, a lambda term is a homomorphic construction of a sequent proof just when the term can be used as a construction of the proof in the sense of Howard (1980), by invoking a generalized (homomorphic) version of the Curry-Howard formulae-as-types correspondence between lambda terms and proofs (see also Wansing 1992). The usual correspondence is generalized by interpreting any slash of the type logic as the functional type arrow ' \leftarrow ', and any modal operator is not interpreted at all.

A grammar G generates a language of word trees (from the antecedents of provable sentence sequents) labeled by these terms, called *term-labeled trees*. The lambda calculus is used in a standard fashion to model the compositional meaning structures of natural language; an example of a term-labeled tree that is *unsubtyped* (i.e. not showing subterm types) is:

(1.1) $((\mathbf{loves}(\mathbf{Mary}))(\mathbf{John}))^s$:



We will in the sequel use a standard compact representation of syntax trees with the aid of square brackets. The bold faced items are constants of the lambda language, and the bolding of e.g. **Mary** is usually taken to be the meaning of 'Mary.'

Let us say that a grammar G *generates* a subtyped term-labeled tree

N^s : **Struc**

if and only if: 1. The tree **Struc** is generated by G with some proof Π in R_G ; this means Π must be a proof of some particular sequent

Γ : **Struc** $\Rightarrow s$.

2. N^s is a homomorphic construction of Π . Now let us say that a grammar G generates an untyped term-labeled sentence tree $N_{\text{ust}}^s : \mathbf{Struc}$ iff there is some subtyped term-labeled tree $N^s : \mathbf{Struc}$ generated by G such that N_{ust}^s is just N^s stripped of the type labels on its subterms. The set of (subtyped) term-labeled sentence trees generated by G is called the (*subtyped*) *term-labeled tree language* of G , and is denoted $\text{ATL}(G)$.

We will be most interested in the languages of untyped term-labeled trees generated by various classes of type-logical grammars. We maintain that such TLT languages are the most relevant for representing natural languages, which we can all recognize as consisting of syntactically structured sentences with intended compositional meaning recipes. We assume here that meaning composition, at least in its skeletal form, is representable as a lambda term of the fragment defined above. It is important to note that this fragment is not in general sufficient for representing realistic semantic readings in many cases because of the strict limits placed on the locations of bound variables, but the composition of meaning from subterms can nonetheless be shown.

1.3 Distribution and syntactic homogeneity

We formalize the notions of syntactic distribution that were proposed by linguists of the American Structuralist tradition, in particular Bloomfield (1933) and Wells (1947). By the term *usage* it is meant a term-labeled tree $M[\mathbf{x}] : \mathbf{Struc}[x]$ having a placeholder variable \mathbf{x} in place of a subterm in the lambda term M and a corresponding placeholder x in place of the subtree in the tree \mathbf{Struc} whose meaning occurs at the position occupied by \mathbf{x} in the lambda term. The two placeholder positions as a pair will be termed the *focus* of a usage. For example, $(\mathbf{x}(\mathbf{John}))^s : [\mathbf{John}, x]$ is a usage. This definition makes a usage a modernization of Bloomfield's notion of the function of a linguistic form. Any $\langle \text{meaning}, \text{tree} \rangle$ pair which can fill the slots in the usage to make a term-labeled sentence tree in the TLT language at hand is a Bloomfieldian *form* which is said to *have the usage*, and to *instantiate the focus*. All the forms in a language which have a particular usage constitute a *form class* determined by that usage. Under our definitions, a form is then a pair consisting of a tree of words and a lambda term representing the meaning of the word tree.

Let us consider how usages can be considered equivalent in a term-labeled language. We begin by defining what it means for respective subterms of lambda terms M and N to function equivalently in their terms. The notion of two lambda-terms $M[\mathbf{x}]$ and $N[\mathbf{y}]$ having respective subterms \mathbf{x} and \mathbf{y} which *function equivalently* is defined inductively:

1. Any lambda terms M and N are subterms of themselves,

and they function equivalently as such. 2. Two application terms $(M_1(M_2))[\mathbf{x}]$ and $(N_1(N_2))[\mathbf{y}]$ have respective subterms \mathbf{x} and \mathbf{y} which function equivalently just when either $M_1[\mathbf{x}]$ and $N_1[\mathbf{y}]$ have \mathbf{x}, \mathbf{y} functioning equivalently, or $M_2[\mathbf{x}]$ and $N_2[\mathbf{y}]$ have \mathbf{x}, \mathbf{y} functioning equivalently. 3. Two abstraction terms $(\lambda z.M)[\mathbf{x}]$ and $(\lambda z.N)[\mathbf{y}]$ have subterms \mathbf{x}, \mathbf{y} which function equivalently just when $M[\mathbf{x}]$ and $N[\mathbf{y}]$ have \mathbf{x}, \mathbf{y} functioning equivalently.

We now consider a similar notion for syntax trees, and define inductively when two subtrees are *positioned equivalently* within their parent trees. 1. Any two syntax trees S and T are subtrees of themselves, and are positioned equivalently as such. 2. Any two binary-branching syntax trees (S_1, S_2) and (T_1, T_2) have respective subtrees γ and δ positioned equivalently if and only if for either $i = 1$ or $i = 2$ the two trees S_i and T_i have the respective subtrees γ and δ positioned equivalently.

We can put these two together to define when two usages are equivalent in shape. Two usages $M_1[\mathbf{x}] : \mathbf{Struc}_1[x]$ and $M_2[\mathbf{y}] : \mathbf{Struc}_2[y]$ are said to be *shape-equivalent* just when: 1. The lambda terms $M_1[\mathbf{x}]$ and $M_2[\mathbf{y}]$ have respective subterms \mathbf{x} and \mathbf{y} which function equivalently. 2. The syntactic trees $\mathbf{Struc}_1[x]$ and $\mathbf{Struc}_2[y]$ have respective subtrees x and y positioned equivalently. These definitions clearly make shape-equivalence of usages reflexive, symmetric, and transitive, and thus the relation is a formal equivalence.

Any set of shape-equivalent usages will be called a *usage class*; a usage class which contains all possible usages that can be formed from a sample \mathcal{D} of TLTs is termed a *complete usage class relative to \mathcal{D}* . The members of a usage class are each said to *illustrate* the class. When all the usages in a class determine the same form class, the usage class itself is held to also determine that form class; otherwise, the usage class does not determine a form class.

1 EXAMPLE Here is a term-labeled tree language with four elements:

$$\begin{array}{ll}
 (1.2) \quad (\mathbf{sings}(\mathbf{John}))^s : [\text{John, sings}] & (\mathbf{x}(\mathbf{John}))^s : [\text{John, x}] \\
 (\mathbf{sings}(\mathbf{Mary}))^s : [\text{Mary, sings}] & (\mathbf{x}(\mathbf{Mary}))^s : [\text{Mary, x}] \\
 (\mathbf{laughs}(\mathbf{Susan}))^s : [\text{Susan, laughs}] & (\mathbf{x}(\mathbf{Susan}))^s : [\text{Susan, x}] \\
 (\mathbf{laughs}(\mathbf{Tom}))^s : [\text{Tom, laughs}] & (\mathbf{x}(\mathbf{Tom}))^s : [\text{Tom, x}]
 \end{array}$$

The four usages on the right can be formed from the respective term-labeled trees in the sample on the left. They are all shape-equivalent, and they comprise a *complete* shape-equivalent set of usages which can be formed from the sample.

Any form class determined by a complete usage class relative to a particular language will be termed a *part of speech* in that language, in the spirit of Bloomfield. In other words, a part of speech in a term-labeled tree language L is an

equivalence class of trees of words (in fact, word occurrences) which is the form class determined by each of the usages in a usage class which is complete relative to L . The parts of speech of a term-labeled language are thus well-defined if and only if the members of each complete usage class are guaranteed to determine the same form class, throughout the language. In the example above, for instance, the four usages are a complete class relative to the four-sentence sample taken as a language. However, the first two usages form a usage class whose form class is different from that of the last two usages, as mentioned above. There is thus not a properly defined system of parts of speech in this four-sentence language. A term-labeled language L will be termed *syntactically homogeneous* if and only if it has well-defined parts of speech, meaning every complete usage class that can be formed from its trees determines a form class.

There is a second important property of TLT languages to be defined, which is entwined with the ways they are generated by their grammars. Let us say that a TLT $N_1: \Gamma_1$ is a *recursively redundant extension* of a second TLT $N_2: \Gamma_2$ just when: 1. they are shape-equivalent (i.e. instantiations of shape-equivalent usages); 2. there are corresponding foci within the two TLTs which are instantiated by respective subtrees $\nu_1: \gamma_1$ and $\nu_2: \gamma_2$ where the first is larger than the second; 3. the subtree $\nu_1: \gamma_1$ in turn contains a subtree $\nu'_1: \gamma'_1$ which is shape-equivalent to $\nu_2: \gamma_2$; 4. if a third TLT $N_3: \Gamma_3$ is created from $N_2: \Gamma_2$ by replacing $\nu_2: \gamma_2$ with the result of removing $\nu'_1: \gamma'_1$ from $\nu_1: \gamma_1$ (i.e. their difference), then $N_3: \Gamma_3$ is generated by a grammar for the other two by means of a proof precisely equivalent to one which generates $N_2: \Gamma_2$.

2 EXAMPLE The first TLT below is a recursively redundant extension of the second TLT, as can be seen by instantiating the subtrees of the definition as shown.

(1.3) **died(the((from(the((by(the(sea)))town)))man))):**

[[The [man [from [the [town [by [the sea]]]]]] died]

(1.4) **died(the((from(the(town)))man))):**

[[The [man [from [the town]]]] died]

$\nu_1: \gamma_1 = \mathbf{from(the((by(the(sea)))town))):$

[from [the [town [by [the sea]]]]]

$\nu_2: \gamma_2 = \mathbf{from(the(town))):$ [from [the town]]

$\nu'_1: \gamma'_1 = \mathbf{by(the(sea))):$ [by [the sea]]

Now, a (possibly infinite) TLT language L is said to be *finitely illustratable* if, and only if, some (nonempty) finite set \mathcal{C} of usages together illustrate the maximal set of usage classes which can be formed from trees of L so that no usage in \mathcal{C} is a recursively redundant extension of any other usage in \mathcal{C} . Such a set \mathcal{C} is called an *illustration set* for the language. If L is infinite, then the remaining infinite number of its usage classes not illustrated in \mathcal{C} must all consist of usages that are recursively redundant extensions of members of usage classes illustrated by \mathcal{C} .

1.4 Discovering structurally unified lexicons

Since we are willing to provide our human language learning model with a Universal Type Logic, what remains to be learned of a particular language is precisely its lexicon. This includes learning the syntactic and semantic categories that are at work in the language. We would like to induce this information from an American Structuralist-style analysis of the language sample to determine which items are distributionally equivalent, or intersubstitutable in some sense.

An algorithm is presented below, dubbed SUTL, which induces a set of lexicons from a sample of term-labeled trees. It is based on the GFTL procedure outlined in Fulop (2003, 2002b), which learns general form type-logical lexicons from such a sample, and also employs the optimal unification procedure of Buszkowski and Penn (1990).

A *general form* lexicon I_{gf} for a TLT sample is defined to be one in which distinct variable primitive types (drawn from a denumerable set Var) each occur only once as an atomic type (though they may occur additionally as subtypes of complex types), and which generates exactly the sample as its TLT language when used together with R_G . The notion of a general form lexicon and its role as an intermediate step in grammar discovery was elucidated by Buszkowski and Penn (1990) in their work on the discovery of classical categorial grammars from syntactic skeletons.

3 LEMMA *For any two shape-equivalent term-labeled trees $T_1 = M[\mathbf{x}] : Struc[x]$ and $T_2 = M'[\mathbf{y}] : Struc'[y]$ in the sample data, in which form occurrences $\langle \mathbf{x}, x \rangle$ and $\langle \mathbf{y}, y \rangle$ occur equivalently (they may be occurrences of any possibly distinct trees of words), at least one general form lexicon I will be discovered by GFTL which assigns unifiable types to the syntactic word occurrences x and y if indeed they are words. More generally if x or y or both are not words, the respective proofs Π and Π' of T_1 and T_2 using some such I will nonetheless be such that the word trees x in Π and y in Π' have types which are unifiable. Moreover, there will be some general form lexicon I_{gf} discovered by GFTL which has these properties*

for every such pair x, y of forms positioned equivalently in shape-equivalent trees.

Proof. This lemma follows from the definition of the GFTL algorithm of Fulop (2003), which is guaranteed to find all possible proofs Π and Π' which can be obtained by using the lambda terms M, M' as proof-building recipes, in whatever type logic R is being used, of the respective term-labeled trees. It is a property of the principal typing carried out in GFTL that \mathbf{x} and \mathbf{y} are assigned to semantic types which are alphabetic variants. Since the lambda terms M and M' are homomorphic constructions of the respective proofs Π and Π' of the syntax trees which they label, it must then be possible (though not necessary) to prove each of them in such a way that the types of x and y are the same in the two proofs, up to alphabetic variation, and are thus unifiable.

Turning to the final statement in the lemma, because GFTL is defined to output a lexicon from every possible combination of ways of proving each term-labeled tree in the sample, at least one such set of proofs for the sample will be such that every set of shape-equivalent trees will have unifiable types assigned to every set of respective subtrees which are positioned equivalently. \square

We now present the SUTL algorithm:

1. The input is a sample of term-labeled trees; some permitted type logic must also be chosen. Use GFTL to obtain the set $GF = \{I_1, \dots, I_m\}$ of m general form lexicons for the sample. These have the property that every occurrence of a primitive type other than s in their categories is a new variable—they are totally ununified.
2. Create a set U of all usages which can be formed from the term-labeled sentence trees in the sample by replacing single lexical items (words, in the simplest view) with placeholders $\langle \mathbf{x}, x \rangle$.
3. Create a set UC of all the complete usage classes in U . Let $n = |UC|$.
4. For each usage class \mathcal{C}_i in UC , form a set P_i of word instances which is the union of all the word instances in the form classes determined by the members of \mathcal{C}_i .
5. Select the first general form lexicon I_1 .
6. Do for all $1 \leq i \leq n$: Each of the words p in P_i will have a set T_p of syntactic types that are assigned to *its instances in proofs of those sample trees that are shape-equivalent to the usage class \mathcal{C}_i* , using I_1 . Create the set T_{P_i} which is the union $\bigcup_p T_p$.

7. Find all the optimal unifiers of the family $\mathcal{T} = \{T_{P_1}, \dots, T_{P_n}\}$ using the optimal unification procedure of Buszkowski and Penn (1990).
8. Discard any unifiers of the family which are incomplete, i.e. that fail to completely unify each set T_{P_i} . For a given general form lexicon I this may leave no unifiers, but we are guaranteed by Lemma 3 to have at least one lexicon that will produce a complete unifier, for any finite sample of TLTs.
9. A unifier of a lexicon formed in the above fashion relative to a set of term-labeled trees will be termed a *structural unifier*. Find all the unifications of the general form lexicon that result from applying each structural unifier in turn, if there are any. Add these unified lexicons to the set of lexicons to be output.
10. Select the next general form lexicon in the set GF and repeat from step 6, until there are no more lexicons in GF .
11. At last, output the complete set of *structurally unified* lexicons that has been built up through the iteration.

Structurally unified grammars have the interesting property that they generate their TLT languages so that distinct syntactic categories are assigned to syntactically distinct positions, while syntactically equivalent positions all have the same category. This property turns out to be equivalent to syntactic homogeneity of TLT languages.

Consider the class $\bigcup_R \Lambda TL(R)_{\text{homfin}}$ of finitely illustratable homogeneous TLT languages generated by permitted logics, obviously a proper subset of all the TLT languages of permitted logics. How does it relate to the class $\bigcup_R \Lambda TL(R)_{\text{sutl}}$ of all TLT languages of SUTL-range grammars under permitted logics?

4 PROPOSITION

$$\bigcup_R \Lambda TL(R)_{\text{sutl}} \subseteq \bigcup_R \Lambda TL(R)_{\text{homfin}}.$$

5 LEMMA *Any SUTL grammar G generates a TLT language whose usage classes are either illustrated in the learning sample or are recursively redundant extensions of those which are.*

6 LEMMA *For any SUTL grammar G , all lexical items in equivalently positioned foci in generated shape-equivalent TLTs will have the same type there.*

This lemma follows immediately from steps 6–8 of the SUTL algorithm together with the preceding lemma, whose proof we leave to the reader. Proposition 4 follows from these lemmas using an induction on the structure of the lambda term labels to establish the extension of the second lemma to all syntactically equivalent positions, no matter whether they are occupied by lexical items or larger trees.

7 PROPOSITION

$$\bigcup_R \Lambda TL_{homfin} \subseteq \bigcup_R \Lambda TL(R)_{sutl}.$$

Let us sketch a proof. Suppose a language $L \in \bigcup_R \Lambda TL(R)_{homfin}$. There is a finite set \mathcal{D} of trees drawn from L from which every usage in an illustration set \mathcal{C} can be formed, and containing every syntactically distinct word use in L at least once. There will then be some general form lexicon I_{gf} discovered by applying GFTL to \mathcal{D} using some logic R (viz. a logic suitable to generate L), such that I_{gf} can be structurally unified using SUTL to obtain a lexicon I_{su} that generates L .

8 COROLLARY

$$\bigcup_R \Lambda TL(R)_{homfin} = \bigcup_R \Lambda TL(R)_{sutl}.$$

It remains an empirical question, as Chomsky might put it, whether the class of possible human languages can in reality be construed to be either finitely illustratable or syntactically homogeneous. They are, at least, clear conditions which together substantially limit the class of languages, and we invite serious argument on the conjecture from an empirical linguistic perspective. Let us provisionally claim that no recognized syntactic phenomenon requires languages to be otherwise in any obvious way—suggesting that any human language can be generated by an SUTL grammar.

Bibliography

- Bloomfield, L. (1933). *Language*. Allen and Unwin, London.
- Buszkowski, W. and G. Penn (1990). Categorical grammars determined from linguistic data by unification. *Studia Logica*, **49**:431–454.
- Carpenter, B. (1999). The Turing-completeness of multimodal categorical grammars. In J. Gerbrandy, M. Marx, M. de Rijke, and Y. Venema, eds., *JFAK: Essays dedicated to Johan van Benthem on the occasion of his 50th birthday*. Institute for Logic, Language, and Computation, University of Amsterdam. Available on CD-ROM at <http://turing.wins.uva.nl>.

- Fulop, S. A. (2002a). On the logic and learning of language. Manuscript, University of Chicago.
- Fulop, S. A. (2002b). Semantic bootstrapping of type-logical grammars. Manuscript, University of Chicago.
- Fulop, S. A. (2003). Learnability of type-logical grammars. In L. Moss and G.-J. Kruijff, eds., *Proceedings of Formal Grammars / Mathematics of Language*, volume 53 of *Electronic Notes in Theoretical Computer Science*. Elsevier. To appear.
- Gentzen, G. (1934). Untersuchungen über das logische Schliessen. *Math. Zeitschrift*, **39**:176–210, 405–431. English translation in Szabo (1969).
- Howard, W. A. (1980). The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, eds., *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 479–490. Academic Press, New York.
- Jäger, G. (2002). Residuation, structural rules and context freeness. Manuscript, University of Potsdam.
- Jäger, G. (2003). On the generative capacity of multi-modal categorial grammars. *Journal of Language and Computation*. To appear.
- Lambek, J. (1961). On the calculus of syntactic types. In R. Jakobson, ed., *Structure of Language and its Mathematical Aspects*, Proceedings of Symposia in Applied Mathematics, pp. 166–178. American Mathematical Society, Providence, RI.
- Moortgat, M. (1997). Categorial type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*. Elsevier.
- Morrill, G. V. (1994). *Type Logical Grammar: Categorial Logic of Signs*. Kluwer, Dordrecht.
- Szabo, M. (1969). *The Collected Papers of Gerhard Gentzen*. North-Holland, Amsterdam.
- Wansing, H. (1992). Formulas-as-types for a hierarchy of sublogics of intuitionist propositional logic. In D. Pearce and H. Wansing, eds., *Non-classical Logics and Information Processing*, volume 619 of *Lecture Notes in Artificial Intelligence*, pp. 125–145. Springer-Verlag, Berlin.
- Wells, R. S. (1947). Immediate constituents. *Language*, **83**:81–117.