Kracht/Sportiche 214, Winter 2003: Combinatory Categorial Grammar

Basic Asssumptions

A **syntactic object** (alias **sign**) is a triple $E :: C :: M$, where

1. E is its phonetic/phonological form.
2. C is its syntactic category.
3. M is its meaning.

A syntactic theory has to specify (a) the basic entities (**lexicon**), and (b) the **modes of combination**.

Combinatory Categorial Grammar (**CCG**) is a particular theory inside categorial grammar that assumes **surface compositionality**. This is to say that on the side of phonetic form, the mechanism for combination is *concatenation*. Moreover, it inherits from categorial grammar a tight coupling between categories and meanings. Once the rule of combination is specified in the syntactic categories, its semantics is completely determined. The reason that this theory is called *combinatorial* CG is that the particular syntactic modes of combination correspond to certain combinators on the semantic side.

Categories and Meanings

Definition 1 (Types) *Let B be a set of **basic types**. We denote by $\mathrm{Typ}(B)$ the smallest set which contains B and $\alpha \to \beta$ for every $\alpha, \beta \in \mathrm{Typ}(B)$.*

We assume the basic types e (entities) and t (truth values). Although there may be more (say, worlds and time points), we shall not need them here.

Definition 2 (Interpretation) *Fix for every $\alpha \in B$ a set $\ulcorner\alpha\urcorner$. Then for every nonbasic type*

$$\ulcorner\alpha \to \beta\urcorner := \ulcorner\alpha\urcorner \to \ulcorner\beta\urcorner := \{f : dom(f) = \ulcorner\alpha\urcorner, rg(f) \subseteq \ulcorner\beta\urcorner\}$$

Definition 3 (Categories) *Let C be a set of **basic categories**. We denote by $\mathrm{Cat}(C)$ the smallest set which contains C and α/β as well as $\alpha\backslash\beta$ for every $\alpha, \beta \in \mathrm{Typ}(C)$.*

Steedman uses in particular V, VP, NP, S as basic categories (this list is not exhaustive). The map from categories to types is a homomorphism. This is to say, one needs to fix a map $v : C \to \mathrm{Typ}(B)$. Then, for complex categories:

$$v(\alpha/\beta) := v(\alpha\backslash\beta) := v(\beta) \to v(\alpha)$$

Warning: There are two conventions in CG. One is to write $\beta\backslash\alpha$ when α is the resulting category, and β the argument (which is used by the majority) the others (including Mark Steedman) write $\alpha\backslash\beta$ in this case, so that the resulting category is always to the left. We follow his practice here for compatibility. Neither notation is actually always superior.

We remark here that we do not require C and B to be equal. This allows to introduce separate syntactic categories for the same semantic type. It also allows to assign a complex type to a basic category. This is, although not always acknowledged, the standard practice. Notice that when an expression has category α, its type is $v(\alpha)$, and therefore its meaning should be in $\ulcorner v(\alpha) \urcorner$.

Basic Categorial Grammar

Assume the following are entries of the lexicon.

`walks`	:: S\NP	:: $\lambda x.\mathsf{walk}'(x)$
`watches`	:: (S\NP)/NP	:: $\lambda x.\lambda y.\mathsf{watch}'(x)(y)$
`gives`	:: ((S\NP)/NP)/NP	:: $\lambda x.\lambda y.\lambda z.\mathsf{give}'(x)(y)(z)$
`Dominique`	:: NP	:: d'
`Paul`	:: NP	:: p'
`the fish`	:: (S\NP)\((S\NP)/NP)	:: $\lambda\Omega.\Omega(\iota x.\mathsf{fish}'(x))$

Here, x, y, and z are variables of type e, Ω a variable of type $e \to (e \to t)$.

Notice that $\lambda x.\mathsf{walk}'(x)$ is the same as walk'. The introduction of the λ–operator is of no real service here, except to please the eye. Standard categorial grammar (the so–called Ajdukiewicz–Bar Hillel Calculus) has the following rules:

$$\begin{aligned} A_>(E_1 :: \alpha/\beta :: M_1, E_2 :: \beta :: M_2) &= E_1^\frown E_2 :: \alpha :: M_1(M_2) \\ A_<(E_1 :: \beta :: M_1, E_2 :: \alpha\backslash\beta :: M_2) &= E_1^\frown E_2 :: \alpha :: M_2(M_1) \end{aligned}$$

Here, $E^\frown F$ is the concatenation of E with F (with an interspersed blank). $A_>$ is called **forward application**, $A_<$ **backward application**. Given a

lexicon, the language generated is simply the set of all syntactic objects that can be created from the lexicon using these two rules. In particular, it contains the object

$$\texttt{Paul walks} :: \mathrm{S} :: \mathsf{walk}'(\mathsf{p}')$$

In categorial grammar, one typically uses this representation:

$$\begin{array}{cc} \texttt{Paul} & \texttt{walks} \\ \mathrm{NP} & \mathrm{S}\backslash\mathrm{NP} \\ \hline & \mathrm{S} \end{array}$$

Montague basically used the AB–Calculus, although he added special phonetic functions (such as transforming the pronouns $\texttt{he}_n$ into something else). By and large, however, the function used is concatenation. Steedman follows Montague in paying only minimal attention to agreement.

Combinators

The central idea of CCG is the insight that $A_>$ and $A_<$ can be given a precise semantics as well. Their interpretation is that of a **combinator**, by which we may simply understand a closed λ–term. All combinators can be built from these two basic ones: S, and K. These symbols are treated as proxy for the following terms:

$$\begin{array}{rcl} \mathsf{K} & = & \lambda x.\lambda y.x \\ \mathsf{S} & = & \lambda x.\lambda y.\lambda z.(xz)(yz) \end{array}$$

Warning: While particular lexical entries have particular types, these combinators are polymorphic. They exist in all consistent types. For example, if x has type α and y has type β, the type of K must be $\alpha \to (\beta \to \alpha)$ ("weakening" in logic). In natural language, typically only the coordinators are polymorphic. For example, `and` and `or` have any type of the form $(\alpha\backslash\alpha)/\alpha$, while `not` has any type of the form α/α. The semantics needs to match the type in each case. We ignore the issue of typing of combinators henceforth.

The basic idea is that language(s) employ combinators to serve their basic need to build denotations for complex expressions. Since combinators are actually immensely powerful, the work consist in finding the right base of combinators.

> *The Principle of Combinatory Type Transparency.* All syntactic rules are type–transparent versions of one of a small number of simple semantic operations over functions. ((Steedman, 2000), Page 37.)

Steedman basically uses the following combinators. (Notice the absence of K — since it is destructive!)

$$\begin{array}{lcl} \mathsf{B} & := & \lambda x.\lambda y.\lambda z.x(yz) \\ \mathsf{S} & := & \lambda x.\lambda y.\lambda z.(xz)(yz) \\ \mathsf{T} & := & \lambda x.\lambda y.yx \end{array}$$

B (bluebird, see (Smullyan, 1985)) is used to compose functions: $((\mathsf{B}M)N) = \lambda z.M(N(z))$ is otherwise written $M \circ N$. This is the most frequently used combinator. The syntactic rule based on this combinator has effectively been proposed by Geach in order to create some flexibility in using the categories. S (starling), is needed in parasitic gap constructions, and T is nothing but type raising (when applied to one argument only).

The combinator is a semantic operation, and its syntactic counterpart is actually not unique. We have seen this with function application: a functor may look for its argument to its right or to its left. Steedman assumes that a syntactic operation is compatible with the combinator only if (a) the type assignment matches it, (b) the argument is found to the right of the functor if the functor contains $/\alpha$, and to the left otherwise and (c) the principle of directional inheritance is adhered to: it says that the directionality of the argument is specified by the functor. Directionality is not free as long as the argument has not been cancelled in the syntax. For example: B allows the following syntactic counterparts.

$$\begin{array}{llllll} (+\,+\,+) & \alpha/\beta & \beta/\gamma & \rightsquigarrow & \alpha/\gamma \\ (+\,+\,-) & \alpha/\beta & \beta/\gamma & \rightsquigarrow & \alpha\backslash\gamma \\ (+\,-\,+) & \alpha/\beta & \beta\backslash\gamma & \rightsquigarrow & \alpha/\gamma \\ (+\,-\,-) & \alpha/\beta & \beta\backslash\gamma & \rightsquigarrow & \alpha\backslash\gamma \\ (-\,+\,+) & \alpha\backslash\beta & \beta/\gamma & \rightsquigarrow & \alpha/\gamma \\ (-\,+\,-) & \alpha\backslash\beta & \beta/\gamma & \rightsquigarrow & \alpha\backslash\gamma \\ (-\,-\,+) & \alpha\backslash\beta & \beta\backslash\gamma & \rightsquigarrow & \alpha/\gamma \\ (-\,-\,-) & \alpha\backslash\beta & \beta\backslash\gamma & \rightsquigarrow & \alpha\backslash\gamma \end{array}$$

Of these, $(+\,+\,-)$, $(+\,-\,+)$, $(-\,+\,+)$, $(-\,+\,-)$, $(-\,-\,+)$ and $(-\,-\,-)$ are ruled out. (b) rules out $(-\,+\,+)$, $(-\,+\,-)$, $(-\,-\,+)$ and $(-\,-\,-)$, (c) rules

out $(++-)$, $(-+-)$, $(+-+)$, and $(--+)$. Likewise, when we interchange the order of the categories above, six out of eight combinations are ruled out. Thus, B adds the following four rules to the syntax:

$$\begin{array}{lcl} B_{>}(E_1 :: \alpha/\beta :: M_1, E_2 :: \beta/\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha/\gamma :: \mathsf{B}M_1M_2 \\ B_{<}(E_1 :: \beta\backslash\gamma :: M_1, E_2 :: \alpha\backslash\beta :: M_2) & = & E_1^\frown E_2 :: \alpha\backslash\gamma :: \mathsf{B}M_2M_1 \\ Bx_{>}(E_1 :: \alpha/\beta :: M_1, E_2 :: \beta\backslash\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha\backslash\gamma :: \mathsf{B}M_1M_2 \\ Bx_{<}(E_1 :: \beta/\gamma :: M_1, E_2 :: \alpha\backslash\beta :: M_2) & = & E_1^\frown E_2 :: \alpha/\gamma :: \mathsf{B}M_2M_1 \end{array}$$

$B_{>}$ and $B_{>}x$ are called **forward composition** and **mixed forward composition**, $B_{<}$ and $B_{<}x$ **backward composition** and **mixed backward composition**. To see that this strengthens the language generated from the lexicon, let us give this example (see (Geach, 1972)). Assume that nonrelational nouns (`elephant`, `car`) have category N, relational nouns (`father`) category N/PP, and adjectives (`tall`, `heavy`) have category N/N. Then `tall elephant` is a phrase in AB, but not `tall father`. However, adding composition works:

$$\begin{array}{rl} & B_{>}(\texttt{tall} :: \mathrm{N/N} :: \lambda\mathcal{Q}.\lambda x.\mathsf{tall}'(x) \wedge \mathcal{Q}(x), \\ & \quad \texttt{father} :: \mathrm{N/PP} :: \lambda y.\lambda x.\mathsf{father\text{-}of}'(y)(x)) \\ = & \texttt{tall father} :: \mathrm{N/PP} :: \lambda y.\lambda x.(\mathsf{tall}'(x) \wedge \mathsf{father\text{-}of}'(y)(x)) \end{array}$$

The combinator B allows us to delay feeding an argument; we put it 'on hold' so to speak.

We remark here that to make this theory fully satisfactory, a parametrized family of combinators is needed, where basically any number of arguments can be put on hold. Syntactically, the generalized forward composition rule is as follows:

$$\alpha/\beta \quad (\beta/\gamma)/\$_1 \rightsquigarrow (\alpha/\gamma)/\$_1$$

(Analogously, generalized backward composition and the mixed composition rules are defined.) Here the following convention is at work ((Steedman, 2000), Page 42):

> *The $–convention.* For a category α, $\{\alpha\$\}$ (respectively $\{\alpha/\$\}$, $\{\alpha\backslash\$\}$) denotes the set containing α and all functions (respectively, leftward functions, rightward functions) into a category in $\{\alpha\$\}$ (respectively, $\{\alpha/\$\}$, $\{\alpha\backslash\$\}$).

This is not the best of all definitions. What it means typographically is that $\$_1$ above abbreviates a sequence of category symbols separated by slashes

(left associativity assumed) (see (Kracht, 2003)). The semantics is defined accordingly.

T gives rise to the following rules otherwise known as **type raising**.

$$\begin{array}{lcl} T^{\beta}_{>}(E :: \alpha :: M) & := & (E :: \beta/(\beta\backslash\alpha) :: \mathsf{T}^{v(\beta)}M) \\ T^{\beta}_{<}(E :: \alpha :: M) & := & (E :: \beta\backslash(\beta/\alpha) :: \mathsf{T}^{v(\beta)}M) \end{array}$$

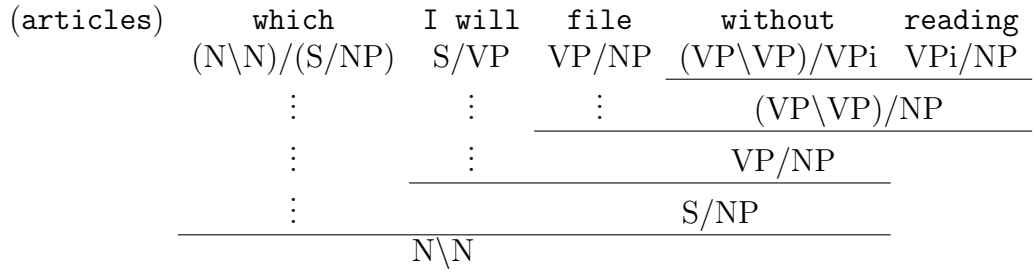
Notice that the resulting category contains β, so we cannot not simply write $T_>$ or $T_<$, for the result is then not unique. The type of $\mathsf{T}^{v(\beta)}M$ is that $v(\beta) \to v(\alpha)$. This precaution is actually necessary: otherwise there is no guarantee that the syntax and semantics match as required. It is assumed that type raising is restricted to prevent infinite loops. Moreover, language particular restrictions apply. Type raising gives us that the following sentences are grammatical.

`Fred`	`likes`	`and`	`I`	`eat`	`beans`
NP	$\vdots$	$\vdots$	NP	$\vdots$	$\vdots$
S/(S\NP)	(S\NP)/NP	$\vdots$	S/(S\NP)	(S\NP)/NP	$\vdots$
S/NP		$(\alpha\backslash\alpha)/\alpha$	S/NP		$\vdots$
$\vdots$		(S/NP)\(S/NP)			NP
S/NP					S\(S/NP)
S					

In this derivation, α must be set to S/NP. (Steedman, 2000), Page 45, assumes that case markers are responsible for type raising. If case markers actually did the type raising, then the rules $T_>$ and $T_<$ above would actually not be necessary. However, the identity of the category over which we raise differs from construction to construction, so case markers would actually not have a unique type. In the example above, the object raises over transitive verbs that already have a subject: observe, namely, that S/NP is *not* the category of intransitive verbs (which is S\NP!

Finally we look at S. Again, there are four possibilities:

$$\begin{array}{lcl} S_{>}(E_1 :: (\alpha/\beta)/\gamma :: M_1, E_2 :: \beta/\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha/\gamma :: \mathsf{S}M_1M_2 \\ Sx_{>}(E_1 :: (\alpha/\beta)\backslash\gamma :: M_1, E_2 :: \beta\backslash\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha\backslash\gamma :: \mathsf{S}M_2M_1 \\ S_{<}(E_1 :: \beta/\gamma :: M_1, E_2 :: (\alpha\backslash\beta)\backslash\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha\backslash\gamma :: \mathsf{S}M_1M_2 \\ Sx_{<}(E_1 :: \beta/\gamma :: M_1, E_2 :: (\alpha\backslash\beta)/\gamma :: M_2) & = & E_1^\frown E_2 :: \alpha/\gamma :: \mathsf{S}M_2M_1 \end{array}$$



To create the constituent `file without reading` we need the rule of backward crossed substitution ($Sx_<$).

Gapping and Coordination

The particular strength of CCG lies in the fact that for a given sentence one may have several constituent analyses. However, while Lambek Calculus allows *any* constituent structure for a given sentence if it has at least one, CCG may reject some. The particular claim of CCG is that anything that is a constituent can be coordinated, so that if something cannot be coordinated it ought not to be a constituent and vice versa. Consider the sentence (1). (1a) – (1f) suggest that the only subsequence that is not a constituent is `ate the` and `Fred ate the`.

(1) `Fred ate the beans.`
(1a) `Fred ate and Harry liked the beans.`
(1b) *`Fred ate the and Harry had some beans.`
(1c) *`Fread ate the and liked some bananas.`
(1d) `Fred ate the beans and Harry liked the beans.`
(1e) `Fred ate the beans and liked some bananas.`
(1f) `Fred ate the beans and some bananas.`

Warning: Some of the constructions above might not be called coordination in other theories. Here, the distinction between, say, coordination and right node raising is not made. Hence, the following structures (2a,b) coexist, while (2c,d,e) do not.

(2a) `Fred (ate (the beans))`
(2b) `(Fred ate) (the beans)`
(2c) *`((Fred ate) the) beans`
(2d) *`(Fred (ate the)) beans`
(2e) *`Fred ((ate the) beans)`.

CCG achieves this goal as follows. Noun phrases are systematically type raised before they combine with the verb. (We will not display the step of typeraising to avoid showing too many steps.) This means that in English, both SV and VO can both be constituents, though not in one and the same structure. S(VO) constituent structure is derived as follows.

`Fred`	`saw`	`Paul`
S/(S\NP)	(S\NP)/NP	(S\NP)\((S\NP)/NP)
⋮	S\NP	
S		

And (SV)O constituent structure is derived thus.

`Fred`	`saw`	`Paul`
S/(S\NP)	(S\NP)/NP	⋮
S/NP		S\(S/NP)
S		

Likewise, for other basic constituent orders different constituent structures are available. For example, Dutch and German are SOV in subordinate clauses, so the verb has category (S\NP)\NP, and by type raising into a suitable category, we get either S(OV) or (SO)V. However, the type raising does not depend on the structure. Call $V_0 := S$ and $V_{n+1} := V_n\backslash NP$. Then transitive verbs are V_2, subjects will be V_0/V_1 and objects V_1/V_2. Now SO is a constituent of category V_0/V_2, by forward composition. By application, *OV* is a constituent. So, both S(OV) and (SO)V are constituents of category S. Finally, in Irish (which is VSO), we find that both (VS)O and V(SO) are available. However, notice the different types for subject and object:

`Chonaic`	`Eoghan`	`Siobhán`
saw	*Eoghan*	*Siobhán*
(S/NP)/NP	(S/NP)\((S/NP)/NP)	S\(S/NP)
⋮	S\((S/NP)/NP)	
S		

This predicts the following gapping patterns:

SOV : *SOV and SO	SO and SOV
VSO : VSO and SO	*SO and VSO

This is basically what has been observed for these languages. Steedman claims that to the extent that they can be violated this reflects alternative serializations in the claus, thus do not counterexemplify the scheme above.

The troublemaker is the SVO type (English!). In English, SO can be a constituent. However, the principle of adjacency forbids the formation of a SO constituent in an SVO structure. Steedman therefore assumes an *ex post* reanalysis of the SVO clause as a VSO clause:

> *Virtual conjunct revealing rule.*
>
> $$\alpha \quad \rightsquigarrow \quad \beta \quad \alpha\backslash\beta$$
>
> where $\beta = \mathrm{S}/\$$.

With this rule, the gapping pattern for SVO languages become that of an VSO language.

SVO : SVO and SO *SO and SOV

Language Particulars

CCG not only assumes that the lexicon varies from language to language. It is also the rules that change. Moreover, rules may be restricted to particular instance of categories. For example, forward composition ($B_>$) is restricted to two cases: (a) $\beta = \mathrm{S}_{-sub}/\$$, where S_{-sub} is the category of a tensed main clause, or (b) $\beta = \mathrm{S}$ (tensed clause or IP), and $\gamma = (\beta\backslash\delta)$. Similarly for the other rules.

The contrast between subjects and objects in extraction in English is attributed to the unavailability of the rule $B_>x$ in English, which is needed to generate (b). For (a) only $B_>$ is required.

(a) (`a man who(m)` [`I think that`]$_{\mathrm{S/S}}$ [`Dexter likes`]$_{\mathrm{S/NP}}$

(b) *(`a man who(m)` [`I think that`]$_{\mathrm{S/S}}$ [`likes Dexter`]$_{\mathrm{S\backslash NP}}$

On the other hand, $B_<x$ *is* available in English, though only with the β restricted to $S\$$.

Theme–Rheme Articulation and Prosody

Consider the sentence (1) again.

(1) `Fred ate the beans.`

We have seen that it may have several constituent analyses. Once we add the intonational contour, however, the ambiguity disappears (at least for this sentence). Moreover, different contours indicate different theme–rheme articulation. Steedman assumes an autosegmental approach (so that contours can be added as if they are segmental markers but then spread over the structure) and additionally the theory by (Pierrehumbert, 1980). Of the six pitch accent tones, only two will be studied: H^* and $\mathrm{L} + \mathrm{H}^*$. Although it is the intention to keep tones separate from the lexicon, the theory at present compiles them into the lexicon, so that we get three (!) lexical entries for a given word:

$$\begin{array}{l} \texttt{ate} \quad :: (\mathrm{S}_\theta \backslash \mathrm{NP}_\theta)/\mathrm{NP}_\theta :: *\mathsf{ate}' \\ \mathrm{L} + \mathrm{H}^* \end{array}$$

$$\begin{array}{l} \texttt{ate} \quad :: (\mathrm{S}_\rho \backslash \mathrm{NP}_\rho)/\mathrm{NP}_\rho :: *\mathsf{ate}' \\ \mathrm{H}^* \end{array}$$

$$\texttt{ate} \quad :: (\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP} \quad :: \mathsf{ate}'$$

This requires comment. First, the tone is annotated below the word; there are the two tones mentioned above plus a null tone. The semantics of the tones is as follows. The pitch accent tones identify whether the intonational phrase is a theme or a rheme. Correspondingly, each basic category b is split into two variants: b_θ and b_ρ, denoting the theme part and the rheme part, respectively. Categories are now formed as usual, but now they carry the additional features. The instantiation of the feature depends on the identity of the pitch accent: H^* for rheme, $\mathrm{L} + \mathrm{H}^*$ for theme. Within theme and rheme, the word carrying the accent tone is the *focus*, the other words being the *background*. Focus marking is done by an asterisk, for want of a better notation (the semantics of theme and rheme as well as topic and focus is not worked out in CCG). A word that has zero tone can be either thematic or rhematic. (Steedman uses techniques from unification categorial grammar: no tone means the symbol for theme/rheme is unspecified and unifies with either.)

Finally, there are three boundary tones. They constitute the following

signs.

$$\begin{array}{ll} \text{L} & :: \text{S\$}_\varphi\backslash\text{S\$}_\eta :: \lambda x.\eta' x \\ \text{LL\%} & :: \text{S\$}_\varphi\backslash\text{S\$}_\eta :: \lambda x.\eta' x \\ \text{LH\%} & :: \text{S\$}_\varphi\backslash\text{S\$}_\eta :: \lambda x.\eta' x \end{array}$$

Here, the \$–convention is used again. Furthermore, η is a variable ranging over the set $\{\theta, \rho\}$. φ is a new symbol. (So, we should actually have three basic categories for each b ... It seems to me [MK] that this can be avoided by viewing φ as a variables just like η.) Notice that $\$_\varphi$ results from $\$_\eta$ by systematically changing b_η to b_φ, $b \in C$.

$$\dfrac{\dfrac{\dfrac{\begin{array}{cc}\texttt{FRED} & \texttt{ate} \\ \text{L+H}^* & \text{LH\%} \\ \text{S}_\theta/(\text{S}_\theta\backslash\text{NP}_\theta) & (\text{S}\backslash\text{NP})/\text{NP} \end{array}}{\begin{array}{cc} \text{S}_\theta/\text{NP}_\theta & \text{S\$}_\varphi\backslash\text{S\$}_\eta \end{array}}}{\text{S}_\varphi/\text{NP}_\varphi} \quad \dfrac{\dfrac{\begin{array}{cc}\texttt{the} & \texttt{BEANS} \\ & \text{H}^* \\ \text{NP/N} & \text{N}_\rho \end{array}}{\text{NP}_\rho}}{\dfrac{\text{S}_\rho\backslash(\text{S}_\rho/\text{NP}_\rho)}{\text{S\$}_\varphi/\text{S\$}_\rho}} \quad \text{LL\%}}{\text{S}_\varphi\backslash(\text{S}_\varphi\backslash\text{NP}_\varphi)}}{\text{S}_\varphi}$$

This sentence has a unique analysis — by virtue of its intonation contour. Here are two more examples.

(3) Q: What about FRED? What did HE do to the beans?
A: (`FRED`) (`ATE the beans.`)
L+H*LH% H* LH%
Theme *Rheme*

(4) Q: I know who COOKED the beans. But then, who ATE them?
A: (`FRED`) (`ATE the beans.`)
H*L L+H* LH%
Rheme *Theme*

It is not always the case that intonation contours disambiguate the phrases. Here is an example:

(`the green`) BEANS
H* LL%
`the` (`green` BEANS)
H* LL%

Discussion

CCG uses a unification categorial backbone, with language specific rules drawn from a small set of universally available rules. CCG has the following features:

- It shares with CG the tight connection between syntax and (type theoretic) semantics.
- It uses a small number of rules to account for the syntactic behaviour of languages.
- In attributing several structures for sentences it can basically unify many construction types (gapping, right node raising) as coordination of constituents. Basic gapping patterns are predicted solely on the basis of the surface order in the main clause.
- Intonation contours can be given precise categories as well as semantics. In addition to contributing to the theme–rheme and the focus–background articulation of the sentence they serve to narrow down the choice of analyses.
- Parsing can be done 'lazily', assuming left bracketing whenever possible. There exist polynomial algorithms ((Vijay-Shanker and Weir, 1990)) and CCGs have been shown to be weakly equivalent to TAGs.

On the other hand, there are several problematic aspects.

- While the verb still encodes the basic sentence order, it is the arguments that have to also encode it through type raising. This is a problematic aspect of all categorial grammars: the organisation of the structure is basically encoded several times.
- The reanalysis rule for English is problematic semantically. If the resulting meaning is known, it is not always clear that we can extract the verb from it (such a function is anyway not given by the λ–calculus). Thus, if `Fred sold the car to Mary.` means the same as `Mary bought the car from Fred.` then (a) may mean either (b) or (c). (The surface realisation of the argument relations are not visible in the semantics.)

(a) `Fred sold the car to Mary and the bicycle to John.`
(b) `Fred sold the car to Mary and Fred sold the bicycle to John.`
(c) `Fred sold the car to Mary and Mary bought the bicycle from John.`

References

(Geach, 1972) 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.

(Kracht, 2003) Marcus Kracht. *Mathematics of Language*. Mouton de Gruyter, Berlin, 2003?

(Pierrehumbert, 1980) Janet Pierrehumbert. *The Phonology and Phonetics of English Intonation*. PhD thesis, MIT, 1980.

(Smullyan, 1985) Raymond Smullyan. *To Mock a Mocking Bird*. Knopf, New York, 1985.

(Steedman, 2000) Mark Steedman. *The Syntactic Process*. MIT Press, Cambridge (Mass.), 2000.

(Vijay-Shanker and Weir, 1990) K. Vijay-Shanker and David J. Weir. Polynomial time parsing of combinatory categorial grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh*, 1990.