

Integrating typed feature structures into Combinatory Categorical Grammar

Mark McConville*
University of Edinburgh

Inheritance-driven CCG encapsulates a uniform approach to the elimination of redundancy in CCG lexicons, where grammars incorporate inheritance hierarchies of lexical types, defined over a simple, feature-based category description language. The resulting formalism is partially ‘model-theoretic’, in that the category notation is interpreted against an underlying set of tree-like typed feature structures. This extension of CCG subsumes a number of other proposed category notations devised to allow for the construction of more efficient lexicons.

1 The CCG formalism

In its most basic conception, a Combinatory Categorical Grammar (henceforth CCG) over alphabet Σ of terminal symbols is an ordered triple $\langle A, S, L \rangle$, where A is an alphabet of saturated category symbols, S is a distinguished element of A , and L is a lexicon, i.e. a mapping from Σ to categories over A . The set of categories over alphabet A is the closure of A under the binary infix connectives $/$ and \backslash and the associated ‘modalities’ of (Bal02). For example, assuming the saturated category symbols ‘S’ and ‘NP’, here is a simple CCG lexicon (with modalities omitted for convenience):

- (1) John \vdash NP
- Mary \vdash NP
- loves \vdash (S\NP)/NP

The combinatory projection of a CCG lexicon is its closure under a finite set of resource-sensitive combinatory operations such as forward application (2), backward application (3), forward type raising (4), and forward composition (5):

- (2) $X/Y \ Y \Rightarrow X$
- (3) $Y \ X \backslash Y \Rightarrow X$
- (4) $X \Rightarrow Y/(Y \backslash X)$
- (5) $X/Y \ Y/Z \Rightarrow X/Z$

CCG $\langle A, S, L \rangle$ over alphabet Σ generates string $s \in \Sigma^*$ just in case $\langle s, S \rangle$ is in the combinatory projection of lexicon L . The derivation in Figure 1 shows that CCG

* Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh, EH8 9LW, Scotland. Email: Mark.McConville@ed.ac.uk

(1) generates the sentence *John loves Mary*, assuming that ‘S’ is the distinguished symbol, and where $>\mathbf{T}$, $>\mathbf{B}$ and $>$ denote instances of forward raising, forward composition and forward application respectively.

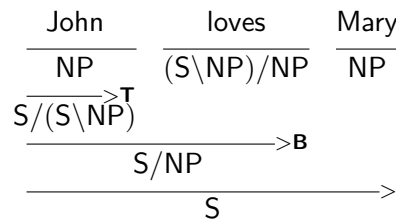


Figure 1
A CCG derivation

A complete introduction to the CCG formalism is found in (Ste00).

2 Lexical redundancy in CCG

CCG is a particularly interesting and effective theory of human linguistic competence, and forms the basis for efficient practical tools for natural language processing. However, in a number of cases, work has been hindered by the lack of an agreed approach to eliminating redundancy in CCG lexicons. This deficit constitutes a particular problem for radically lexicalised formalisms like CCG, where the customary way of treating bounded dependency constructions such as case, agreement and binding is by means of multiple lexical category assignments. For example, the fragment of English schematised in Table 1, though small, exemplifies non-trivial aspects of case and number agreement.

John		John
he	loves	me
the girl		you
girls		him
I		us
you	love	them
we		the girl
they		girls
girls		girls

Table 1
A fragment of English

Table 2 presents the simplest CCG lexicon for this language. This lexicon contains a number of multiple category assignments: (a) the proper noun *John* and the second person pronoun *you* are each assigned to *two* categories, one for each case distinction; (b) the plural suffix *-s* is assigned to *three* categories, depending on both the case and ‘bar level’ of the resulting nominal; and (c) the definite article *the* is assigned to *four* categories, one for each combination

$$\begin{array}{l}
\text{John} \vdash \text{NP}_{\text{subj}}^{\text{sg}}, \text{NP}_{\text{obj}} \\
\text{girl} \vdash \text{N}_{\text{sg}} \\
\text{s} \vdash \text{N}_{\text{pl}} \setminus \text{N}_{\text{sg}}, \text{NP}_{\text{subj}}^{\text{pl}} \setminus \text{N}_{\text{sg}}, \text{NP}_{\text{obj}} \setminus \text{N}_{\text{sg}} \\
\text{the} \vdash \text{NP}_{\text{subj}}^{\text{sg}} / \text{N}_{\text{sg}}, \text{NP}_{\text{obj}} / \text{N}_{\text{sg}}, \\
\quad \text{NP}_{\text{subj}}^{\text{pl}} / \text{N}_{\text{pl}}, \text{NP}_{\text{obj}} / \text{N}_{\text{pl}} \\
\text{l, we, they} \vdash \text{NP}_{\text{subj}}^{\text{pl}} \\
\text{me, us, them, him} \vdash \text{NP}_{\text{obj}} \\
\text{you} \vdash \text{NP}_{\text{subj}}^{\text{pl}}, \text{NP}_{\text{obj}} \\
\text{he} \vdash \text{NP}_{\text{subj}}^{\text{sg}} \\
\text{love} \vdash (\text{S} \setminus \text{NP}_{\text{subj}}^{\text{pl}}) / \text{NP}_{\text{obj}} \\
\text{s} \vdash ((\text{S} \setminus \text{NP}_{\text{subj}}^{\text{sg}}) / \text{NP}_{\text{obj}}) \setminus ((\text{S} \setminus \text{NP}_{\text{subj}}^{\text{pl}}) / \text{NP}_{\text{obj}})
\end{array}$$
Table 2

A CCG lexicon

of case and number agreement distinctions. In each of these three cases, there appears to be no pre-theoretically obvious ambiguity involved. Thus, this lexicon violates the following efficiency-motivated ideal on human language lexicons, in the Chomskyan sense of locus of non-systematic information:

ideal of functionality a lexicon is ideally a *function* from morphemes to category labels, modulo genuine ambiguity

Another efficiency-motivated ideal which the CCG lexicon in Table 2 violates is the following:

ideal of atomicity a lexicon is a mapping from morphemes ideally to *atomic* category labels

In the lexicon in Table 2, the transitive verb *love* is assigned to the decidedly non-atomic category label $(\text{S} \setminus \text{NP}_{\text{subj}}^{\text{pl}}) / \text{NP}_{\text{obj}}$. Lexicons which violate the criteria of functionality and atomicity are not just inefficient in terms of storage space and development time. They also fail to capture linguistically significant generalisations about the behaviour of the relevant words or morphemes.

The functionality and atomicity of a CCG lexicon can be easily quantified. The functionality ratio of the lexicon in Table 2, with 22 lexical entries for 14 distinct morphemes, is $\frac{22}{14} = 1.6$. The atomicity ratio is calculated by dividing the number of saturated category symbol-tokens by the number of lexical entries, i.e. $\frac{36}{22} = 1.6$.

Numerous, more or less *ad hoc* generalisations of the basic CCG category notation have been proposed with a view to eliminating these kinds of lexical redundancy. One area of interest has involved the structure of the saturated category symbols themselves. (Boz02) presents a version of CCG where saturated category symbols are modified by unary modalities annotated with morphosyntactic features. The features are themselves ordered according to a language-particular join semi-lattice. This technique, along with the insistence that lexicons of ag-

glutinating languages are necessarily *morphemic*, allows generalisations involving the morphological structure of nouns and verbs in Turkish to be captured in an elegant, non-redundant format. (Erk03) generalises this approach, modelling saturated category labels as typed feature structures, constrained by underspecified feature structure descriptions in the usual manner.

(Hof95) resolves other violations of the ideal of functionality in CCG lexicons for languages with ‘local scrambling’ constructions by means of ‘multiset’ notation for unsaturated categories, where scope and direction of arguments can be underspecified. For example, a multiset category label like $S\{\backslash NP_{\text{subj}}, \backslash NP_{\text{obj}}\}$ is to be understood as subsuming both $(S\backslash NP_{\text{subj}})\backslash NP_{\text{obj}}$ and $(S\backslash NP_{\text{obj}})\backslash NP_{\text{subj}}$.

Computational implementations of the CCG formalism, including successive versions of the Grok/OpenCCG system¹, have generally dealt with violations of the ideal of atomicity by allowing for the definition of macro-style abbreviations for unsaturated categories, e.g. using the macro ‘TV’ as an abbreviation for $(S\backslash NP_{\text{subj}})/NP_{\text{obj}}$. One final point of note involves the project reported in (Bea04), who implements CCG within the LKB system, i.e. as an application of the Typed Feature Structure Grammar formalism of (Cop02), with the full apparatus of unrestricted typed feature structures, default inheritance hierarchies, and lexical rules.

3 Inheritance-driven CCG

One of the aims of the project reported here has been to take a bottom-up approach to the problem of redundancy in CCG lexicons, adding just enough formal machinery to allow the relevant generalisations to be formulated, whilst retaining a restrictive theory of human linguistic competence which satisfies the ‘strong competence’ requirement, i.e. the competence grammar and the processing grammar are identical.

3.1 Type hierarchies

One of the main features of the resulting grammar formalism, which I have called ‘inheritance-driven’ CCG or I-CCG for short, is that the alphabet of saturated category symbols are organised into a ‘type hierarchy’, in the sense of (Car92). From this perspective, a type hierarchy is a weak order $\langle A, \sqsubseteq_A \rangle$, where A is an alphabet of types, \sqsubseteq_A is the ‘subsumption’ ordering on A (with a least element), and every subset of A with an upper bound has a least upper bound. An example type hierarchy is in Figure 2, where for example types ‘Nom_{sg}’ and ‘NP’ are compatible since they have a non-empty set of upper bounds, the least upper bound (or ‘unifier’) being ‘NP_{sg}’.

By itself, this innovation is enough to eliminate a certain amount of the redundancy inherent in the CCG formalism. For example, the CCG lexicon in Table 3, in conjunction with the type hierarchy in Figure 2, generates the fragment of English in Table 1.

¹ <http://openccg.sourceforge.net>

ratio of 1.6, the former's is $\frac{16}{14} = 1.1$.

This improved functionality ratio results from the underspecification of saturated category symbols inherent in the subsumption relation. For example, whereas the proper noun *John* is assigned to two distinct categories in the lexicon in Table 2, in the revised lexicon it is assigned to a single *non-maximal* type 'NP_{sg}' which subsumes the two maximal types 'NP_{sbj}^{sg}' and 'NP_{obj}^{sg}'. In other words, the phenomenon of case syncretism in English proper nouns is captured by having a general singular noun phrase type, which subsumes a plurality of case distinctions.

The CCG formalism which incorporates a type hierarchy of saturated category symbols is equivalent to the 'morphosyntactic CCG' formalism of (Boz02), where features are ordered in a join semi-lattice. Any generalisation which can be expressed in a morphosyntactic CCG can also be expressed in a type-hierarchical CCG, since any lattice of morphosyntactic features can be converted into a type hierarchy. In addition, the revised CCG formalism is equivalent to the formalism described in (Erk03), where saturated categories are modelled as typed feature structures. Any lexicon from either of these formalisms can be translated into a type-hierarchical CCG lexicon whose functionality ratio is either equivalent or lower.

3.2 Simple category descriptions

A second key feature of the I-CCG formalism involves the incorporation of a flexible description language allowing a more fine-grained specification of sets of categories. This feature-based language is inspired by previous attempts to implement categorial grammars within the unification-based paradigm, e.g. (Usz86) and (ZKC87).

The set of simple category descriptions over alphabet A of saturated category symbols is defined as the smallest set Φ such that:

1. $A \subseteq \Phi$
2. for all $\delta \in \{f, b\}$, (SLASH δ) $\in \Phi$
3. for all $\phi \in \Phi$, (ARG ϕ) $\in \Phi$
4. for all $\phi \in \Phi$, (RES ϕ) $\in \Phi$

Note that category descriptions may be infinitely embedded, in which case they are considered to be right-associative, e.g. RES ARG RES SLASH f.

A simple category description like (SLASH f) or (SLASH b) denotes the set of all expressions which seek their argument to the right/left. A description of the form (ARG ϕ) denotes the set of expressions which take an argument of category ϕ , and one like (RES ϕ) denotes the set of expressions which combine with an argument to yield an expression of category ϕ .

Complex category descriptions are simply sets of simple category descriptions, where the assumed semantics is simply that of conjunction.

3.3 Lexical inheritance hierarchies

A third feature of the I-CCG formalism is that it incorporates a second alphabet of non-terminal symbols, in this case a set of ‘lexical types’. The lexical types are organised into an ‘inheritance hierarchy’, constrained by the simple category descriptions defined in the previous section.

Lexical inheritance hierarchies (Fli87) are type hierarchies where each type is associated with a set of expressions drawn from some category description language Φ . Formally, they are ordered triples $\langle B, \sqsubseteq_B, b \rangle$, where $\langle B, \sqsubseteq_B \rangle$ is a type hierarchy, and b is a function from B to $\wp(\Phi)$.

An example lexical inheritance hierarchy over the set of category descriptions over the alphabet of saturated category symbols in Table 2 is presented in Figure 4. The intuition underlying these (monotonic) inheritance hierarchies is that instances of a type must satisfy all the constraints associated with that type, as well as all the constraints it inherits from its supertypes.

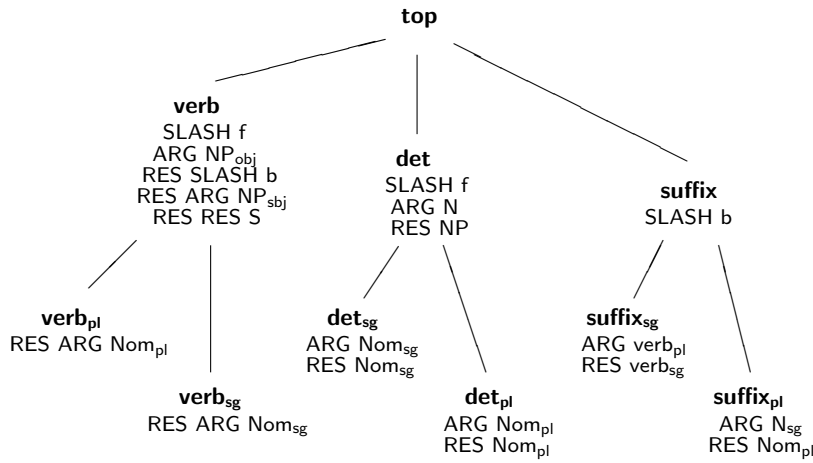


Figure 4
A lexical inheritance hierarchy

This example hierarchy is a *single* inheritance hierarchy, since every lexical type has no more than one immediate supertype. However, *multiple* inheritance hierarchies are also allowed, where a given type can inherit constraints from two supertypes, neither of which subsumes the other. In addition, given some lexical inheritance hierarchy $\langle B, \sqsubseteq_B, b \rangle$, the constraint set of type $t \in B$ is defined as the set of all constraints which are inherited by t , i.e.

$$(6) \quad \bigcup_{t' \sqsubseteq_B t} b(t')$$

3.4 Inheritance-driven CCGs

An I-CCG over alphabet Σ is an ordered 7-tuple $\langle A, \sqsubseteq_A, B, \sqsubseteq_B, b, S, L \rangle$, where $\langle A, \sqsubseteq_A \rangle$ is a type hierarchy of saturated category symbols, $\langle B, \sqsubseteq_B, b \rangle$ is an inheritance hierarchy of lexical types over the set of category descriptions over $A \cup B$, S is a distinguished symbol in A , and lexicon L is a function from Σ to $A \cup B$.

Given an appropriate $\sqsubseteq_{A,B}$ -compatibility relation on the categories over $A \cup B$, the combinatory projection of I-CCG $\langle A, \sqsubseteq_A, B, \sqsubseteq_B, b, S, L \rangle$ can again be defined as the closure of L under the CCG combinatory operations.

The I-CCG lexicon in Table 4, along with the type hierarchy of saturated category symbols in Figure 2 and the inheritance hierarchy of lexical types in Figure 4, generates the fragment of English in Table 1.

John	⊢	NP _{sg}
girl	⊢	N _{sg}
s	⊢	suffix
the	⊢	det
I, we, they	⊢	NP _{sbj} ^{pl}
me, us, them	⊢	NP _{obj} ^{pl}
you	⊢	NP _{pl}
he	⊢	NP _{sbj} ^{sg}
him	⊢	NP _{obj} ^{sg}
love	⊢	verb _{pl}

Table 4
An I-CCG lexicon

Using this lexicon, the sentence *girls love John* is derived as in Figure 5, where derivational steps involve ‘cache-ing out’ sets of constraints from lexical types.

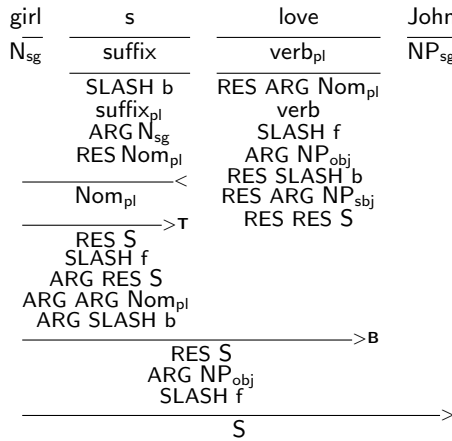


Figure 5
An I-CCG derivation

This derivation relies on a version of the CCG combinatory rules defined in terms of the I-CCG category description language. For example, forward application is expressed as follows — for all complex category descriptions Φ and Ψ such that $(\text{SLASH } b) \notin \Phi$, and $\{\phi \mid (\text{ARG } \phi) \in \Phi\} \cup \Psi$ is compatible, the following is a valid inference:

$$\frac{\Phi \quad \Psi}{\{\phi \mid (\text{RES } \phi) \in \Phi\}} \rightarrow$$

The functionality ratio of the I-CCG lexicon in Table 4 is $\frac{14}{14} = 1$ and the atomicity ratio is $\frac{14}{14} = 1$. In other words, the lexicon is maximally non-redundant, since all the linguistically significant generalisations are encodable within the lexical inheritance hierarchy.

The optimal atomicity ratio of the I-CCG lexicon is a direct result of the introduction of lexical types. In the CCG lexicon in Table 2, the transitive verb *love* was assigned to a non-atomically labelled category $(S \setminus \text{NP}_{\text{subj}}^{\text{pl}}) / \text{NP}_{\text{obj}}$. In the I-CCG's inheritance hierarchy in Figure 4, there is a lexical type 'verb_{pl}' which inherits six constraints whose conjunction picks out exactly the same category. It is with this atomic label that the verb is paired in the I-CCG lexicon in Table 4.

The lexical inheritance hierarchy also has a role to play in constructing lexicons with optimal functionality ratios. The type-hierarchical CCG lexicon in Table 3 assigned the definite article to two distinct categories, one for each grammatical number distinction. The I-CCG lexicon utilises the disjunction inherent in inheritance hierarchies to give each of these a common supertype 'det', which is associated with the properties all determiners share.

Finally, the I-CCG formalism can be argued to subsume the multiset category notation of (Hof95), in the sense that every multiset CCG lexicon can be converted into an I-CCG lexicon with an equivalent or better functionality ratio. Recall that Hoffman uses generalised category notation like $S \{ \setminus \text{NP}_{\text{subj}}, \setminus \text{NP}_{\text{obj}} \}$ to subsume two standard CCG category labels $(S \setminus \text{NP}_{\text{subj}}) \setminus \text{NP}_{\text{obj}}$ and $(S \setminus \text{NP}_{\text{obj}}) \setminus \text{NP}_{\text{subj}}$. Again it should be clear that this is just another way of representing disjunction in a categorial lexicon, and can be straightforwardly converted into a lexical inheritance hierarchy over I-CCG category descriptions.

4 Semantics of the category notation

In the categorial grammar tradition initiated by (Lam58), the standard way of providing a semantics for category notation defines the denotation of a category description as a set of strings of terminal symbols. Thus, assuming an alphabet Σ and a denotation function $[[\dots]]$ from the saturated category symbols to $\wp(\Sigma)$, the denotata of unsaturated category descriptions can be defined as follows, assuming that the underlying logic is simply that of string concatenation:

$$(7) \quad \begin{aligned} [[\phi/\psi]] &= \{s \mid \forall s' \in [[\psi]], ss' \in [[\phi]]\} \\ [[\phi \setminus \psi]] &= \{s \mid \forall s' \in [[\psi]], s's \in [[\phi]]\} \end{aligned}$$

This suggests an obvious way of interpreting the I-CCG category notation defined above. Let's start by assuming that, given some I-CCG $\langle A, \sqsubseteq_A, B, \sqsubseteq_B, b, S, L \rangle$ over alphabet Σ , there is a denotation function $[[\dots]]$ from the *maximal* types in the hierarchy of saturated categories $\langle A, \sqsubseteq_A \rangle$ to $\wp(\Sigma)$. For all non-maximal saturated category symbols ϕ in A , the denotation of ϕ is then the set of all strings in any of ϕ 's subcategories, i.e. $[[\phi]] = \bigcup_{\phi \sqsubseteq_A \psi} [[\psi]]$. The denotata of the simple category descriptions can be defined by universal quantification over the set of simple category descriptions Φ :

- $[[\text{SLASH } \text{f}]] = \bigcup_{\phi, \psi \in \Phi} [[\phi/\psi]]$

- $[[\text{SLASH } b]] = \bigcup_{\phi, \psi \in \Phi} [[\phi \setminus \psi]]$
- $[[\text{ARG } \phi]] = \bigcup_{\psi \in \Phi} [[\psi / \phi]] \cup [[\psi \setminus \phi]]$
- $[[\text{RES } \phi]] = \bigcup_{\psi \in \Phi} [[\phi / \psi]] \cup [[\phi \setminus \psi]]$

This just leaves the simple descriptions which consist of a type in the lexical inheritance hierarchy $\langle B, \sqsubseteq_B, b \rangle$. If we define the constraint set of some lexical type $\phi \in B$ as the set Φ of all category descriptions either associated with or inherited by ϕ , then the denotation of ϕ is defined as $\bigcap_{\psi \in \Phi} [[\psi]]$.

Unfortunately, this approach to interpreting I-CCG category descriptions is insufficient, since the logic underlying CCG is not simply the logic of string concatenation, i.e. CCG allows a limited degree of permutation by dint of the *crossed* composition and substitution operations. In fact, there appears to be no categorial type logic, in the sense of (Moo97), for which the CCG combinatory operations provide a sound and complete derivation system, even in the resource-sensitive system of (Bal02). An alternative approach involves interpreting I-CCG category descriptions against totally well-typed, sort-resolved feature structures, as in the HPSG formalism of (PS94).

4.1 Category frames

Figure 6 illustrates an example of a kind of feature structure I call ‘category frames’:

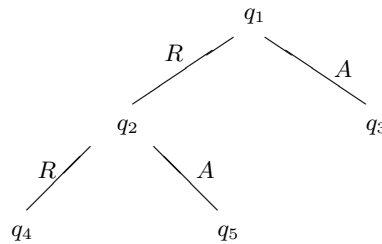


Figure 6
A category frame

Formally, a category frame is an ordered triple $\langle Q, Res, Arg \rangle$, where: (a) Q is a finite, non-empty set of points; (b) Res is the ‘result’ relation on Q i.e. if $\langle q_i, q_j \rangle \in Res$ then point q_j is a ‘result’ of point q_i ; (c) Arg is the ‘argument’ relation on Q i.e. if $\langle q_i, q_j \rangle \in Arg$ then point q_j is an ‘argument’ of point q_i ; (d) Res is a functional mapping from Q to Q i.e. every point has at most one result, although not every point has a result; (e) Arg is a functional mapping from Q to Q i.e. every point has at most one argument, although not every point has an argument; (f) the domains of Res and Arg are identical i.e. every point with a result has an argument and vice versa; and (g) where \prec is the transitive closure of $Res \cup Arg$, $\langle Q, \prec \rangle$ is a strict (partial) order with a first/least element i.e. no point is its own result or argument, and no point is the result or argument of a point contained within its own result or argument.

Note that category frames can be, but are not obliged to be, trees. In other words, reentrancy is allowed. All category frames are rooted, all and only non-end points have exactly one result and exactly one argument, and category frames are acyclic.

4.2 Category models

Figure 7 illustrates an example of a ‘category model’ over the type hierarchy of saturated category symbols in Figure 2. Basically, a category model is a category frame where every end point is annotated with a ‘maximal’ saturated category symbol (i.e. one which has no subtypes), and every non-end point is annotated with a directionality marker, i.e. *f* or *b*.

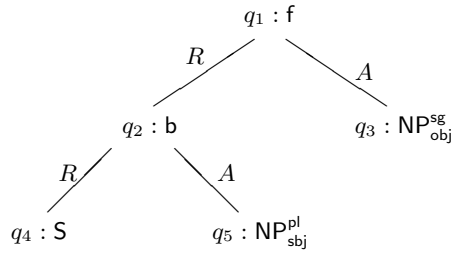


Figure 7
A category model over the type hierarchy in Figure 2

Formally, a category model over type hierarchy $\langle A, \sqsubseteq_A \rangle$ of saturated category symbols is an ordered 5-tuple $\langle Q, Res, Arg, V_S, V_A \rangle$, where: (a) $\langle Q, Res, Arg \rangle$ is a category frame; (b) V_S is a function from the non-end points of $\langle Q, Res, Arg \rangle$ to $\{/, \backslash\}$ i.e. every non-end point is labelled with exactly one directionality symbol; and (c) V_A is a function from the end points of $\langle Q, Res, Arg \rangle$ to A i.e. every end point in the structure is labelled by exactly one maximal saturated category symbol.

4.3 Global satisfaction

Categorial models, as defined in section 4.2, and the simple category descriptions defined in section 3.2 are linked by the familiar notion of logical satisfaction.

Category model $\mathcal{M} = \langle Q, Res, Arg, V_S, V_A \rangle$ over type hierarchy $\langle A, \sqsubseteq_A \rangle$ of saturated category symbols satisfies simple category description ϕ over $A \cup B$, with respect to lexical inheritance hierarchy $\mathcal{H} = \langle B, \sqsubseteq_B, b \rangle$, written $\mathcal{M} \models_{\mathcal{H}} \phi$, if and only if \mathcal{M} locally satisfies ϕ , with respect to \mathcal{H} , from its root point, i.e. $\mathcal{M}, q \models_{\mathcal{H}} \phi$. In other words, global satisfaction is local satisfaction from the root point in a model.

4.4 Local satisfaction

Category model $\mathcal{M} = \langle Q, Res, Arg, V_S, V_A \rangle$ over type hierarchy $\langle A, \sqsubseteq_A \rangle$ of saturated category symbols locally satisfies category description ϕ over $A \cup B$, with respect to lexical inheritance hierarchy $\mathcal{H} = \langle B, \sqsubseteq_B, b \rangle$, from point $q \in Q$, written $\mathcal{M}, q \models_{\mathcal{H}} \phi$, if and only if:

1. where $\phi \in A$: for some $\psi \in A$ such that $\phi \sqsubseteq_A \psi$, $V_A(q) = \psi$, i.e. point q is labelled with some saturated category symbol subsumed by ϕ
2. where $\phi \in B$: for all constraints ψ in the constraint set of ϕ in \mathcal{H} , $\mathcal{M}, q \models_{\mathcal{H}} \psi$, i.e. every simple category description inherited by lexical type ϕ is satisfied from point q
3. where $\phi = (\text{SLASH } \delta)$: $V_S(q) = \delta$, i.e. point q carries the correct directionality symbol
4. where $\phi = (\text{ARG } \psi)$: $\mathcal{M}, \text{Arg}(q) \models_{\mathcal{H}} \psi$, i.e. category description ψ is satisfied from the argument point of q
5. where $\phi = (\text{RES } \psi)$: $\mathcal{M}, \text{Res}(q) \models_{\mathcal{H}} \psi$, i.e. category description ψ is satisfied from the result point of q

To run through a simple example of global satisfaction in I-CCG, note that the category model in Figure 7 satisfies the simple category description ‘ verb_{pl} ’, with respect to the type hierarchy of saturated category symbols in Figure 2 and the lexical inheritance hierarchy in Figure 4. Where the model is denoted \mathcal{M} and the lexical inheritance hierarchy \mathcal{H} , the definition of global satisfaction tells us that $\mathcal{M} \models_{\mathcal{H}} \text{verb}_{\text{pl}}$ just in case $\mathcal{M}, q_1 \models_{\mathcal{H}} \text{verb}_{\text{pl}}$, since q_1 is the root point of \mathcal{M} . The local satisfaction definition then tells us that $\mathcal{M}, q_1 \models_{\mathcal{H}} \text{verb}_{\text{pl}}$ if and only if for every simple category description ϕ in the constraint set of verb_{pl} in \mathcal{H} , $\mathcal{M}, q_1 \models_{\mathcal{H}} \phi$, i.e.

$$(8) \quad \mathcal{M}, q_1 \models_{\mathcal{H}} \text{RES ARG Nom}_{\text{pl}}, \text{SLASH f}, \text{ARG NP}_{\text{obj}}, \\ \text{RES SLASH b}, \text{RES ARG NP}_{\text{subj}}, \text{RES RES S}$$

With regard to the first of these, $\mathcal{M}, q_1 \models_{\mathcal{H}} \text{RES ARG Nom}_{\text{pl}}$, the following process of deduction shows us that it is true:

$$(9) \quad \mathcal{M}, q_1 \models_{\mathcal{H}} \text{RES ARG Nom}_{\text{pl}} \\ \text{iff. } \mathcal{M}, q_2 \models_{\mathcal{H}} \text{ARG Nom}_{\text{pl}} \text{ [since } \text{Res}(q_1) = q_2\text{]} \\ \text{iff. } \mathcal{M}, q_5 \models_{\mathcal{H}} \text{Nom}_{\text{pl}} \text{ [since } \text{Arg}(q_2) = q_5\text{]}$$

Finally, it is clear that $\mathcal{M}, q_5 \models_{\mathcal{H}} \text{Nom}_{\text{pl}}$ since point q_5 is labelled with a saturated category symbol $\text{NP}_{\text{subj}}^{\text{pl}}$ which is subsumed by Nom_{pl} in the type hierarchy in Figure 2.

Proving that the other simple category descriptions listed in (8) are satisfied from the root point in Figure 7 is left as an exercise for the interested reader.

4.5 Combinatory projection of an I-CCG

Given I-CCG $\langle A, \sqsubseteq_A, B, \sqsubseteq_B, b, S, L \rangle$ over alphabet Σ , the combinatory projection of lexicon L is defined recursively. The ‘base’ is defined as the smallest set L' where for all $\langle s, \phi \rangle \in L$ and all category models \mathcal{M} over $\langle A, \sqsubseteq_A \rangle$ such that $\mathcal{M} \models_{\langle B, \sqsubseteq_B, b \rangle} \phi$, $\langle s, \mathcal{M} \rangle \in L'$. In other words, the base of the combinatory projection of an

I-CCG lexicon is a mapping from each lexical form to all the category models which globally satisfy its lexical type.

The combinatory projection of an I-CCG lexicon is then the closure of the base under the small, finite set of CCG combinatory operations discussed in section 1.

5 Conclusion

This paper has focused on an extension of the CCG formalism of (Ste00), incorporating various aspects of typed feature structure grammars such as lexical inheritance hierarchies and the distinction between structures and structural descriptions, the two notions being linked by logical satisfaction. The resulting formalism, termed ‘inheritance-driven’ CCG or I-CCG for short, involves two main innovations: (a) the alphabet of saturated category symbols is organised into a type hierarchy; and (b) there is a secondary alphabet of ‘lexical types’, itself organised into an inheritance hierarchy over a simple, feature-based constraint language. In particular, I presented a detailed semantics for the category description language in terms of a simple notion of tree-like, abstract category models.

The I-CCG formalism was shown to have significant advantages over the baseline CCG formalism, in that it allows us to formulate *non-redundant* natural language lexicons. In addition, I-CCG is shown to subsume a number of other, more or less *ad hoc* extensions to the CCG formalism, which have been developed in order to reduce lexical redundancy, in particular the ‘morphosyntactic’ CCG of (Boz02) and the ‘multiset’ CCG of (Hof95).

References

- Jason Baldridge. *Lexically Specified Derivational Control in Combinatory Categorical Grammar*. PhD thesis, University of Edinburgh, 2002.
- John Beavers. Type-inheritance Combinatory Categorical Grammar. In *Proceedings of the 20th International Conference on Computational Linguistics, University of Geneva, 2004*.
- Cem Bozsahin. The combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186, 2002.
- Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.
- Ann Copestake. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford CA, 2002.
- Gunes Erkan. A Type System for Combinatory Categorical Grammar. Master’s thesis, Middle East Technical University, Ankara, 2003.
- Daniel Paul Flickinger. *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford University, 1987.
- Beryl Hoffman. *The Computational Analysis of the Syntax and Interpretation of “Free” Word Order in Turkish*. PhD thesis, University of Pennsylvania, 1995.
- Joachim Lambek. The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65:154–170, 1958.
- Michael Moortgat. Categorical type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. North Holland, Amsterdam, NL, 1997.
- Carl Jesse Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, 1994.

- Mark Steedman. *The Syntactic Process*. MIT Press, Cambridge MA, 2000.
- Hans Uszkoreit. Categorical Unification Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics, Bonn*, pages 187–194, 1986.
- Henk Zeevat, Ewan Klein, and Jo Calder. Unification Categorical Grammar. In Nicholas Haddock, Ewan Klein, and Glyn Morrill, editors, *Categorical Grammar, Unification Grammar and Parsing*, Working Papers in Cognitive Science. Centre for Cognitive Science, University of Edinburgh, 1987.