Inheritance and the CCG Lexicon

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

Abstract

I propose 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 'constraint-based', in that the category notation is interpreted against an underlying set of tree-like feature structures. I argue that this version of CCG subsumes a number of other proposed category notations devised to allow for the construction of more efficient lexicons. The formalism retains desirable properties such as tractability and strong competence, and provides a way of approaching the problem of how to generalise CCG lexicons which have been automatically induced from treebanks.

1 The CCG formalism

In its most basic conception, a 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 Baldridge (2002). For example, assuming the saturated category symbols 'S' and 'NP', here is a simple CCG lexicon (modalities omitted):

(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 (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:

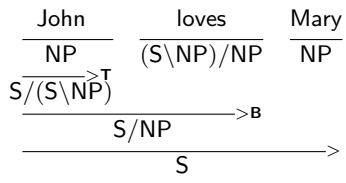

Figure 1: A CCG derivation

2 Lexical redundancy in CCG

CCG has many advantages both as a theory of human linguistic competence and as a tool for practical natural language processing applications (Steedman, 2000). However, in many cases development has been hindered by the absence of an agreed uniform approach to eliminating redundancy in CCG lexicons. This poses a particular problem for a radically lexicalised formalism such as CCG, where it is customary to handle bounded

dependency constructions such as case, agreement and binding by means of multiple lexical category assignments. Take for example, the language schematised in Table 1. This fragment of English, though small, exemplifies certain 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

The simplest CCG lexicon for this fragment is presented in Table 2:

$\mathsf{John} \vdash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}},\ \mathsf{NP}_{\mathsf{obj}}$
$\mathsf{girl} \vdash \mathsf{N}_{\mathsf{sg}}$
$\mathsf{s} \vdash \mathsf{N}_{\mathsf{pl}} \backslash \mathsf{N}_{\mathsf{sg}},\ \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}} \backslash \mathsf{N}_{\mathsf{sg}},\ \mathsf{NP}_{\mathsf{obj}} \backslash \mathsf{N}_{\mathsf{sg}}$
$\mathsf{the} \vdash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}} / \mathsf{N}_{\mathsf{sg}},\ \mathsf{NP}_{\mathsf{obj}} / \mathsf{N}_{\mathsf{sg}},$
$\mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}} / \mathsf{N}_{\mathsf{pl}},\ \mathsf{NP}_{\mathsf{obj}} / \mathsf{N}_{\mathsf{pl}}$
$\mathsf{I}, \mathsf{we}, \mathsf{they} \vdash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}}$
$\mathsf{me}, \mathsf{us}, \mathsf{them}, \mathsf{him} \vdash \mathsf{NP}_{\mathsf{obj}}$
$\mathsf{you} \vdash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}},\ \mathsf{NP}_{\mathsf{obj}}$
$\mathsf{he} \vdash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}}$
$\mathsf{love} \vdash (\mathsf{S} \backslash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}}) / \mathsf{NP}_{\mathsf{obj}}$
$\mathsf{s} \vdash ((\mathsf{S} \backslash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}}) / \mathsf{NP}_{\mathsf{obj}}) \backslash ((\mathsf{S} \backslash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}}) / \mathsf{NP}_{\mathsf{obj}})$

Table 2: A CCG lexicon

This lexicon exhibits 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 of case and number agreement distinctions. Since in each of these three cases there is no pretheoretical ambiguity involved, it is clear that 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 can be argued to violate is the following:

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

In the above example, the transitive verb *love* is mapped to the decidedly non-atomic category label $(\mathsf{S} \backslash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}}) / \mathsf{NP}_{\mathsf{obj}}$. 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. $\frac{36}{22} = 1.6$.

Various, 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 nature of the saturated category symbols themselves. Bozsahin (2002) 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 agglutinating 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. Erkan (2003) generalises this approach, modelling saturated category labels as typed feature structures, constrained by underspecified feature structure descriptions in the usual manner.

Hoffman (1995) 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 $\mathsf{S}\{\backslash\mathsf{NP}_{\mathsf{sbj}}, \backslash\mathsf{NP}_{\mathsf{obj}}\}$ is to be understood as incorporating both $(\mathsf{S} \backslash \mathsf{NP}_{\mathsf{sbj}}) \backslash \mathsf{NP}_{\mathsf{obj}}$ and $(\mathsf{S} \backslash \mathsf{NP}_{\mathsf{obj}}) \backslash \mathsf{NP}_{\mathsf{sbj}}$.

Computational implementations of the CCG formalism, including successive versions of the

Grok/OpenCCG system[1], 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 $(\mathsf{S}\backslash\mathsf{NP}_{\mathsf{sbj}})/\mathsf{NP}_{\mathsf{obj}}$. One final point of note involves the project reported in Beavers (2004), who implements CCG within the LKB system, i.e. as an application of the Typed Feature Structure Grammar formalism of Copestake (2002), with the full apparatus of unrestricted typed feature structures, default inheritance hierarchies, and lexical rules.

3 Type-hierarchical 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.

I start with a generalisation of the CCG formalism where the alphabet of saturated category symbols is organised into a 'type hierarchy' in the sense of Carpenter (1992), i.e. 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 '$\mathsf{Nom}_{\mathsf{sg}}$' and '$\mathsf{NP}$' are compatible since they have a non-empty set of upper bounds, the least upper bound (or 'unifier') being '$\mathsf{NP}_{\mathsf{sg}}$'.

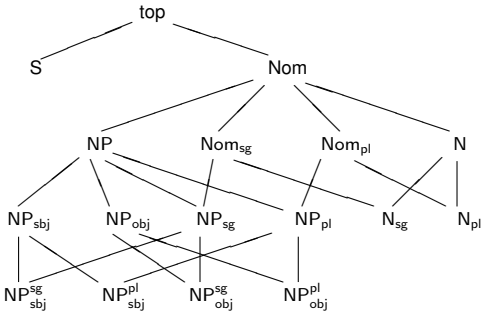

Figure 2: Type hierarchy of saturated categories

A type-hierarchical CCG (T-CCG) over alphabet Σ is an ordered 4-tuple $\langle A, \sqsubseteq_A, S, L \rangle$, where $\langle A, \sqsubseteq_A \rangle$ is a type hierarchy of saturated category symbols, S is a distinguished element of A, and lexicon L is a mapping from Σ to categories over A. Given an appropriate $\sqsubseteq_A$-compatibility relation on the categories over A, the combinatory projection of T-CCG $\langle A, \sqsubseteq_A, S, L \rangle$ can again be defined as the closure of L under the CCG combinatory operations, assuming that variable Y in the type raising rule (4) is restricted to maximally specified categories.

The T-CCG lexicon in Table 3, in tandem with the type hierarchy in Figure 2, generates the fragment of English in Table 1:

$\mathsf{John} \vdash \mathsf{NP}_{\mathsf{sg}}$
$\mathsf{girl} \vdash \mathsf{N}_{\mathsf{sg}}$
$\mathsf{s} \vdash \mathsf{Nom}_{\mathsf{pl}}\backslash\mathsf{N}_{\mathsf{sg}}$
$\mathsf{the} \vdash \mathsf{NP}_{\mathsf{sg}}/\mathsf{N}_{\mathsf{sg}},\ \mathsf{NP}_{\mathsf{pl}}/\mathsf{N}_{\mathsf{pl}}$
$\mathsf{I}, \mathsf{we}, \mathsf{they} \vdash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}}$
$\mathsf{me}, \mathsf{us}, \mathsf{them} \vdash \mathsf{NP}^{\mathsf{pl}}_{\mathsf{obj}}$
$\mathsf{you} \vdash \mathsf{NP}_{\mathsf{pl}}$
$\mathsf{he} \vdash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}}$
$\mathsf{him} \vdash \mathsf{NP}^{\mathsf{sg}}_{\mathsf{obj}}$
$\mathsf{love} \vdash (\mathsf{S}\backslash\mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}})/\mathsf{NP}_{\mathsf{obj}}$
$\mathsf{s} \vdash ((\mathsf{S}\backslash\mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}})/\mathsf{NP}_{\mathsf{obj}})\backslash((\mathsf{S}\backslash\mathsf{NP}^{\mathsf{pl}}_{\mathsf{sbj}})/\mathsf{NP}_{\mathsf{obj}})$

Table 3: A T-CCG lexicon

Using this lexicon, the sentence *girls love John* is derived as in Figure 3:

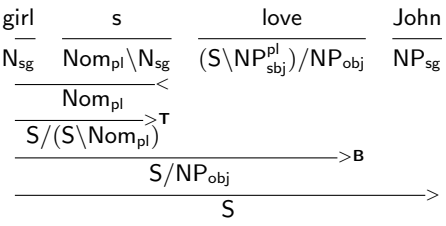

Figure 3: A T-CCG derivation

The T-CCG lexicon in Table 3 comes closer to satisfying the ideal of functionality than does the lexicon in Table 2. While the latter has a functionality 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 T-CCG lexicon it is assigned to a single *non-maximal* type '$\mathsf{NP}_{\mathsf{sg}}$' which subsumes the two maximal types '$\mathsf{NP}^{\mathsf{sg}}_{\mathsf{sbj}}$' and '$\mathsf{NP}^{\mathsf{sg}}_{\mathsf{obj}}$'. In other

[1] http://openccg.sourceforge.net

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 T-CCG formalism is equivalent to the 'morphosyntactic CCG' formalism of Bozsahin (2002), 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 T-CCG, since any lattice of morphosyntactic features can be converted into a type hierarchy. In addition, T-CCG is equivalent to the formalism described in Erkan (2003), where saturated categories are modelled as typed feature structures. Any lexicon from either of these formalisms can be translated into a T-CCG lexicon whose functionality ratio is either equivalent or lower.

4 Inheritance-driven CCG

A second generalisation of the CCG formalism involves adding a second alphabet of non-terminals, in this case a set of 'lexical types'. The lexical types are organised into an 'inheritance hierarchy', constrained by expressions of a simple feature-based category description language, inspired by previous attempts to integrate categorial grammars and unification-based grammars, e.g. Uszkoreit (1986) and Zeevat et al. (1987).

4.1 Simple category descriptions

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 $\delta \in \{\mathsf{f}, \mathsf{b}\}$, $(\mathsf{SLASH}\ \delta) \in \Phi$
3. for all $\phi \in \Phi$, $(\mathsf{ARG}\ \phi) \in \Phi$
4. for all $\phi \in \Phi$, $(\mathsf{RES}\ \phi) \in \Phi$

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.

4.2 Lexical inheritance hierarchies

Lexical inheritance hierarchies (Flickinger, 1987) 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.

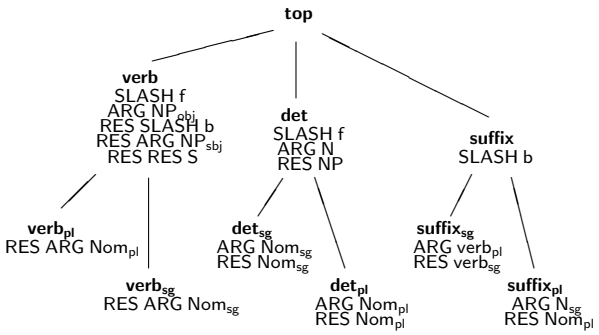

Figure 4: A lexical inheritance hierarchy

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.

4.3 I-CCGs

An inheritance-driven CCG (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. Using this lexicon, the sentence

$\mathsf{John} \vdash \mathsf{NP_{sg}}$
$\mathsf{girl} \vdash \mathsf{N_{sg}}$
$\mathsf{s} \vdash \mathsf{suffix}$
$\mathsf{the} \vdash \mathsf{det}$
$\mathsf{I, we, they} \vdash \mathsf{NP^{pl}_{sbj}}$
$\mathsf{me, us, them} \vdash \mathsf{NP^{pl}_{obj}}$
$\mathsf{you} \vdash \mathsf{NP_{pl}}$
$\mathsf{he} \vdash \mathsf{NP^{sg}_{sbj}}$
$\mathsf{him} \vdash \mathsf{NP^{sg}_{obj}}$
$\mathsf{love} \vdash \mathsf{verb_{pl}}$

Table 4: An I-CCG lexicon

girls love John is derived as in Figure 5, where derivational steps involve 'cache-ing out' sets of constraints from lexical types.

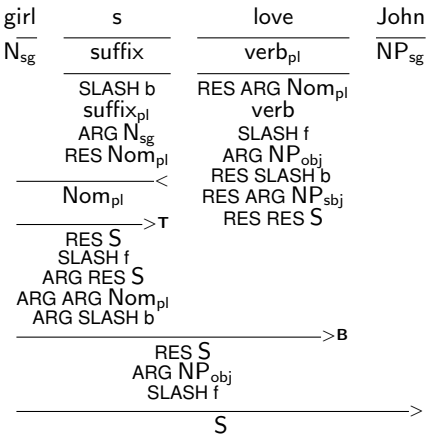

Figure 5: An I-CCG derivation

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 compex category descriptions Φ and Ψ such that $(\mathsf{SLASH\ b}) \notin \Phi$, and $\{\phi \mid (\mathsf{ARG}\ \phi) \in \Phi\} \cup \Psi$ is compatible, the following is a valid inference:

$$\frac{\Phi \qquad \Psi}{\{\phi \mid (\mathsf{RES}\ \phi) \in \Phi\}}>$$

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 T-CCG lexicon in Table 3, the transitive verb *love* was assigned to a non-atomically labelled category $(\mathsf{S}\backslash\mathsf{NP^{pl}_{sbj}})/\mathsf{NP_{obj}}$. In the I-CCG's inheritance hierarchy in Figure 4, there is a lexical type '$\mathsf{verb_{pl}}$' 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 T-CCG lexicon in Table 3 assigned the definite article to two distinct categories, one for each grammatical number distinction. The I-CCG 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 Hoffman (1995), 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 $\mathsf{S}\{\backslash\mathsf{NP_{sbj}}, \backslash\mathsf{NP_{obj}}\}$ to subsume two standard CCG category labels $(\mathsf{S}\backslash\mathsf{NP_{sbj}})\backslash\mathsf{NP_{obj}}$ and $(\mathsf{S}\backslash\mathsf{NP_{obj}})\backslash\mathsf{NP_{sbj}}$. 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.

5 Semantics of the category notation

In the categorial grammar tradition initiated by Lambek (1958), 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 $[[\ldots]]$ 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:

(6) $[[\phi/\psi]] = \{s \mid \forall s' \in [[\psi]], ss' \in [[\phi]]\}$
$[[\phi\backslash\psi]] = \{s \mid \forall s' \in [[\psi]], s's \in [[\phi]]\}$

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 $[[\ldots]]$ 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:

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

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 Moortgat (1997), for which the CCG combinatory operations provide a sound and complete derivation system, even in the resource-sensitive system of Baldridge (2002). An alternative approach involves interpreting I-CCG category descriptions against totally well-typed, sort-resolved feature structures, as in the HPSG formalism of Pollard and Sag (1994).

Given some type hierarchy $\langle A, \sqsubseteq_A\rangle$ of saturated category symbols and some lexical inheritance hierarchy $\langle B, \sqsubseteq_B, b\rangle$, we define a class of 'category models', i.e. binary trees where every leaf node carries a maximal saturated category symbol in A, every non-leaf node carries a directional slash, and every branch is labelled as either a 'result' or an 'argument'. In addition, nodes are optionally labelled with maximal lexical types from B. Note that since only maximal types are permitted in a model, they are by definition sort-resolved. Assuming the hierarchies in Tables 2 and 4, an example category model is given in Figure 6, where arcs by convention point downwards:

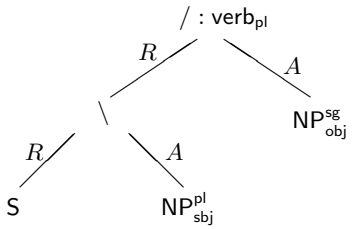

Figure 6: A category model

Given some inheritance hierarchy $\langle B, \sqsubseteq_B, b\rangle$ of lexical types, not all category models whose nodes are labelled with maximal types from B are 'well-typed'. In fact, this property is restricted to those models where, if node n carries lexical type ϕ, then every category description in the constraint set of ϕ is satisfied from n. Note that the root of the model in Figure 6 carries the lexical type 'verb$_{\mathsf{pl}}$'. Since all six constraints inherited by this type in Figure 4 are satisfied from the root, and since no other lexical types appear in the model, we can state that the model is well-typed.

In sum, given an appropriate satisfaction relation between well-typed category models and I-CCG category descriptions, along with a definition of the CCG combinatory operations in terms of category models, it is possible to provide a formal interpretation of the language of I-CCG category descriptions, in the same way as unification-based formalisms like HPSG ground attribute-value notation in terms of underlying totally well-typed, sort-resolved feature structure models. Such a semantics is necessary in order to prove the correctness of eventual I-CCG implementations.

6 Extending the description language

The I-CCG formalism described here involves a generalisation of the CCG category notation to incorporate the concept of lexical inheritance. The primary motivation for this concerns the ideal of non-redundant encoding of lexical information in human language grammars, so that all kinds of linguistically significant generalisation can be captured somewhere in the grammar. In order to fulfil this goal, the simple category description language defined above will need to be extended somewhat. For example, imagine that we want to specify the

set of all expressions which take an $\mathsf{NP}_{\mathsf{obj}}$ argument, but not necessarily as their first argument, i.e. the set of all 'transitive' expressions:

(7) $\mathsf{ARG}\ \mathsf{NP}_{\mathsf{obj}}$
$\cup\ \mathsf{RES}\ \mathsf{ARG}\ \mathsf{NP}_{\mathsf{obj}}$
$\cup\ \mathsf{RES}\ \mathsf{RES}\ \mathsf{ARG}\ \mathsf{NP}_{\mathsf{obj}}$
$\cup\ \ldots$

It should be clear that this category is not finitely specifiable using the I-CCG category notation.

One way to allow such generalisations to be made involves incorporating the $*$ modal iteration operator used in Propositional Dynamic Logic (Harel, 1984) to denote an unbounded number of arc traversals in a Kripke structure. In other words, category description $(\mathsf{RES}^{\star}\ \phi)$ is satisfied from node n in a model just in case some finite sequence of result arcs leads from n to a node where ϕ is satisfied. In this way, the set of expressions taking an $\mathsf{NP}_{\mathsf{obj}}$ argument is specified by means of the category description $\mathsf{RES}^{\star}\ \mathsf{ARG}\ \mathsf{NP}_{\mathsf{obj}}$.

7 Computational aspects

At least as far as the I-CCG category notation defined in section 4.1 is concerned, it is a straightforward task to take the standard CKY approach to parsing with CCGs (Steedman, 2000), and generalise it to take a functional, atomic I-CCG lexicon and 'cache out' the inherited constraints online. As long as the inheritance hierarchy is non-recursive and can thus be theoretically cached out into a finite lexicon, the parsing problem remains worst-case polynomial.

In addition, the I-CCG formalism satisfies the 'strong competence' requirement of Bresnan (1982), according to which the grammar used by or implicit in the human sentence processor is the competence grammar itself. In other words, although the result of cache-ing out particularly common lexical entries will undoubtedly be part of a statistically optimised parser, it is not essential to the tractability of the formalism.

One obvious practical problem for which the work reported here provides at least the germ of a solution involves the question of how to generalise CCG lexicons which have been automatically induced from treebanks (Hockenmaier, 2003). To take a concrete example, Cakici (2005) induces a wide coverage CCG lexicon from a 6000 sentence dependency treebank of Turkish. Since Turkish is a *pro*-drop language, every transitive verb belongs to both categories $(\mathsf{S}\backslash\mathsf{NP}_{\mathsf{sbj}})\backslash\mathsf{NP}_{\mathsf{obj}}$ and $\mathsf{S}\backslash\mathsf{NP}_{\mathsf{obj}}$. However, data sparsity means that the automatically induced lexicon assigns only a small minority of transitive verbs to both classes. One possible way of resolving this problem would involve translating the automatically induced lexicon into sets of fully specified I-CCG category descriptions, generating an inheritance hierarchy of lexical types from this lexicon (Sporleder, 2004), and applying some more precise version of the following heuristic: if a critical mass of words in the automatically induced lexicon belong to both CCG categories X and Y, then in the derived I-CCG lexicon assign all words belonging to either X or Y to the lexical type which functions as the greatest lower bound of X and Y in the lexical inheritance hierarchy.

8 Acknowledgements

The author is indebted to the following people for providing feedback on various drafts of this paper: Mark Steedman, Cem Bozsahin, Jason Baldridge, and three anonymous EACL reviewers.

References

Baldridge, J. (2002). *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh.

Beavers, J. (2004). Type-inheritance Combinatory Categorial Grammar. In *Proceedings of the 20th International Conference on Computational Linguistics, University of Geneva*.

Bozsahin, C. (2002). The combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186.

Bresnan, J., editor (1982). *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge MA.

Cakici, R. (2005). Automatic induction of a CCG grammar for Turkish. In *Proceedings of the Student Research Workshop, 43rd Annual Meeting of the Association for Computational Linguistics, University of Michigan*, pages 73–78.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford CA.

Erkan, G. (2003). A Type System for Combinatory Categorial Grammar. Master's thesis, Middle East Technical University, Ankara.

Flickinger, D. P. (1987). *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford University.

Harel, D. (1984). Dynamic logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic, Volume 2*, pages 497–604. Reidel, Dordrecht.

Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh.

Hoffman, B. (1995). *The Computational Analysis of the Syntax and Interpretation of "Free" Word Order in Turkish*. PhD thesis, University of Pennsylvania.

Lambek, J. (1958). The Mathematics of Sentence Structure. *American Mathematical Monthly*, 65:154–170.

Moortgat, M. (1997). Categorial type logics. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Language*, pages 93–177. North Holland, Amsterdam, NL.

Pollard, C. J. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. The University of Chicago Press.

Sporleder, C. (2004). *Discovering Lexical Generalisations: A Supervised Machine Learning Approach to Inheritance Hierarchy Construction*. PhD thesis, University of Edinburgh.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge MA.

Uszkoreit, H. (1986). Categorial Unification Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics, Bonn*, pages 187–194.

Zeevat, H., Klein, E., and Calder, J. (1987). Unification Categorial Grammar. In Haddock, N., Klein, E., and Morrill, G., editors, *Categorial Grammar, Unification Grammar and Parsing*, Working Papers in Cognitive Science. Centre for Cognitive Science, University of Edinburgh.