Unification Categorial Grammar*

Henk Zeevat, Ewan Klein, and Jo Calder

1. Setting the Scene

Unification categorial grammar (UCG) is a version of categorial grammar enriched by several insights from Head-driven Phrase Structure Grammar (Pollard 1985a,b; Flickinger, Pollard, and Wasow 1985) and PATR-II (Shieber et al. 1986; Shieber 1986). The framework is informed by a combination of theoretical and practical considerations. On the theoretical side, there has been a concern to integrate semantics as tightly as possible with syntax, and moreover to reap the benefits of Kamp’s work on Discourse Representation, while still preserving compositionality. On the practical side, we have been motivated by the desire to develop a theory which could be implemented as a parser in a reasonably efficient manner.

Classical categorial grammar is best presented by defining the relevant notion of category and by stating the rule of functional application. It is customary to start with two primitive categories: N (name) and S (sentence). The set of categories is then defined as:

\[
\begin{align*}
&\text{a. } N \text{ and } S \text{ are categories} \\
&\text{b. If } A \text{ and } B \text{ are categories, } A/B \text{ is a category.}
\end{align*}
\]

Functional application is the following rule:

\[
\text{If } E_1 \text{ is an expression of category } A/B \text{ and } E_2 \text{ is an expression of category } B, \text{ then } E_1 E_2 \text{ (i.e. the concatenation of } E_1 \text{ and } E_2 \text{) is an expression of category } A.\\n\]

A categorial grammar is defined by specifying a list of basic expressions together with their categories. The set of expressions that the grammar generates is the closure of the set of basic expressions under functional application.

*The work reported here was carried out as part of ESPRIT Project 393 (ACORD), “The Construction and Interrogation of Knowledge Bases using Natural Language Text and Graphics”. The paper is a shortened and revised version of “Problems of Dialogue Parsing”, ACORD deliverable T2.2, by the current authors and Marc Moeus. We are grateful to the following people for comments and criticism: Karine Bauchong, Gabriel Bea, Bob Carpenter, Annette Corley, Robert Dale, Thiyri Goldblum, Foru’ Jowsey, Marc Moeus, and Glynn Mondie. All errors are of course our own.

1Recent work carried out at SRI within the PATR framework, in particular Uszkoreit (1986b) and Karttunen (1986) has independently arrived at a similar integration of ideas from categorial grammar. Such a convergence augurs well for the success of this approach.
For applications to natural language, various extensions of this scheme have been proposed. UCG is just one of these extensions, where the notion of a category is expanded. We assign to each expression a number of representations. Most importantly, these are: (a) the way in which the expression is phonologically realised (its orthography, for our purposes), (b) a category specification, and (c) a semantic representation. Following Pollard (1985b), a (complete or incomplete) list of such representations is called a sign.

In UCG, we employ three primitive categories: nouns ('noun'), sentences ('sent') and noun phrases ('np'). These primitive categories admit further specification by features, so that we can distinguish finite and non-finite sentences, nominative and accusative NPs, and so on. Categories are now defined as follows:

(3) a. Any primitive category (together with a syntactic feature specification) is a category.
b. If A is a category, and B is a sign, then A/B is a category.

In a category of the form A/B, we call B the active part of the category, and also of the sign as a whole in which A/B occurs as category. It will be observed that (3b) is just the categorial analog of Pollard's (1985a) proposal for subcategorization, according to which phrasal heads are specified for a list of signs corresponding to their complements.

Within the grammar, we allow not just constant symbols like 'sent' and 'np', but also variables, at each level of representation. Variables allow us to capture the notion of incomplete information, and a sign which contains variables can be further specified by unification. The unification of two representations (if defined) is a third representation which combines all the complete specifications in the first two. Confining our attention to atomic expressions, the situation can be summarized as follows: the unification of two variables is a variable, the unification of a variable and a constant is that constant, and the unification of two distinct constants always fails. We will presently see more complex illustrations of this simple idea.

Unification plays an important role in our use of signs. Functional application in UCG splits into two separate operations: instantiation and stripping. It will be recalled that if a sign has category A/B, then we call B its active part. Instantiation is defined as follows:

(4) \( S_3 \) is the instantiation of \( S_1 \) with respect to \( S_2 \) if it results from \( S_1 \) by unifying its active part with \( S_2 \).

Since unification can fail, there may be many signs with respect to which a given sign \( S_1 \) cannot be instantiated.

The second notion, stripping, receives the definition in (5).

(5) Given a sign \( S_1 \) with category A/B, the result of stripping \( S_1 \) is the sign \( S_1 \) just like \( S_1 \) except that its phonology is the concatenation of \( S_1 \) and \( B \)’s phonology, and its category is stripped down to A.

The rule of functional application now takes the following form:

(6) Let \( S_1 \) and \( S_2 \) be wellformed signs. Then stripping the instantiation of \( S_1 \) with respect to \( S_2 \) also results in a wellformed sign.

The set of wellformed expressions can be defined as the phonologies of the set of wellformed signs. These in turn can be defined as the closure of the lexicon under functional application.

To find out if \( S_1 \) can be applied as a functor to an argument sign \( S_2 \), all that we need to do is look at the actual definition of \( S_1 \)’s category, say A/C, and try to unify C with \( S_2 \). If unification is successful, then stripping the instantiated functor sign will give rise to a result sign \( S_1' \); moreover, instantiation will have made \( S_1' \) more completely specified in various useful ways.

This, in essence, is the structure of UCG. We will complicate the picture by distinguishing two rules of functional application, and by giving more content to the notions of semantics, features and linear order.

2. The Elements of UCG

2.1. Some Notational Conventions

A UCG sign contains four major attributes: phonology (W), syntactic category (C), semantics (S) and order (O). These are usually presented as a vertical list:

\[ W: C: S: O \]

though where convenient they are also written as a sequence, separated by colons:

\[ W/C/S/O \]

(7) illustrates a typical case, the lexical entry for the verb visir.

Unification Categorial Grammar
(7) visit
sent[fin]/W_1:np:x:pre/W_2:np:y:post
[e]VISIT(e, x, y)
O
This is a sign whose phonology attribute is the string *visit*, whose syntactic category is
sent[fin]/W_1:np:x:pre/W_2:np:y:post, whose semantics is [e]VISIT(e, x, y), and whose order is
the unspecified variable O. The significance of these attributes will be explained shortly.
However, some further comment on the complex category may be helpful at this point. It
has the form A/S/S' (i.e. (A/S)/S', assuming association to the left), where S and S' are
themselves signs. Thus, the active part of the category is a sign whose phonology is the
variable W_2, whose category is np, whose semantics is the individual variable y, and whose
order is post.

In order to simplify notation, we feel free to omit unspecified attributes from the
description of the sign (unless the variable occurrence in question is cross-identified with
some other occurrence elsewhere in the sign). In practice, this does not seem to lead to dif­
ficulties. Thus, the example above can be reduced slightly as follows:

(8) visit
sent[fin]/np:x:pre/np:y:post
[e]VISIT(e, x, y)

It is sometimes convenient to have a notation for a sign or attribute that is itself unspecified,
but some of whose components are specified or cross-identified. This is achieved by using
variable functors. Thus
E(W:C:S:O)
introduces a sign E with (specified or unspecified) phonology W, category C, semantics S
and order O.

2.2. Categories

We pointed out earlier that our grammar employs the primitive categories *sent*, *np* and
*noun*. The first two of these can carry additional feature specifications. These are drawn
from the following list inspired by Gazdar, Klein, Pullum, and Sag (1985).

<table>
<thead>
<tr>
<th>Features</th>
<th>Morphology</th>
</tr>
</thead>
<tbody>
<tr>
<td>on <em>sent</em></td>
<td></td>
</tr>
<tr>
<td>FIN</td>
<td>finite verb form</td>
</tr>
<tr>
<td>CFIN</td>
<td>complemented finite verbal element</td>
</tr>
<tr>
<td>BSE</td>
<td>base verb form (i.e. a bare infinitive)</td>
</tr>
<tr>
<td>CBSE</td>
<td>complemented base verb form</td>
</tr>
</tbody>
</table>

**Unification Categorial Grammar**

| INF   | infinitive verb form              |
| PRP   | present participle                |
| PSP   | past participle                   |
| PAS   | passive participle                |

on *np*:

| NOM   | nominative                        |
| OBJ   | objective                          |
| TO    | marked with the preposition to     |
| BY    | marked with the preposition by     |
| OF    | marked with the preposition of     |
| FOR   | marked with the preposition for    |

Having features on these two primitive categories allows for an extra variable, so that
*sent*
can be read as
sent[F]
where F stands for an arbitrary feature.

The main motivation for defining complex categories as C/Sign is that it yields a very
simple notion of functional application, while simultaneously allowing information from the
argument sign to flow to the sign that results from application. This is made possible by
sharing variables between the sign and the active part of its category. The information that is
transmitted can involve semantics, features, order or even the syntactic category of the argu­
ment expression.

Information flows whenever unification occurs, and since unification is commutative, the
flow can go in either direction. We illustrate with a simple example. (9) is a lexical entry
for the verb *walk*:

(9) walks
sent[fin]/np:nom:x:pre
[e]WALK(e, x)

(10) is plausible as a lexical entry for a proper name (though in fact we adopt a slightly dif­
ferent treatment, to be discussed below).

(10) john
np
JOHN

Now suppose we try to unify the active sign

(11) np:nom:x:pre

with (10). In order to see what is going on more clearly, let's use a uniform format which
includes all the variables:
What results from unification of these two is the sign (14).

(14) john np[nom] JOHN pre

The value for phonology is contributed by (12), as is the semantics, JOHN, while a further specification of np is contributed by (13), as is a value for the order attribute. As a result, we obtain the following instantiation of (9):

(15) walks sent[fin](john:np[nom]:JOHN:pre)
[e]WALK(e,JOHN)

That is, C has been unified with sent[fin], O with pre, S with [e]WALK(e,JOHN), and the (omitted) phonology variable with walks. Note that all the changes we obtained in instantiating (15) with respect to (12) occur here as well. Our original expression (17) has been transformed into (19) as a result of the unification.

(19) john sent(fin)/walks:sent(fin)/np[nom]:JOHN:pre:
[e]WALK(e,JOHN):pre

Functional application can now yield (20).

(20) john walks
sent(fin)
[e]WALK(e,JOHN)

Note that this time walks, whose sign is marked for order pre, is indeed preceded by its functor in the phonology of the result sign.

2.3. Linear Order

Natural languages typically exhibit a subtle combination of constraint and freedom in constituent order that are difficult for most linguistic theories to capture, and categorial grammar fares no worse here than other frameworks. Interesting proposals have been made, for example, by Flynn (1983), Kaittunen (1986), Steedman (1985b), Uszkoreit (1985, 1986a).

For the time being, we adopt the restriction that only adjacent constituents can combine grammatically, and that the only order specifications are post and pre. Post says, on a sign: ‘if I am an argument in a functional application, my functor follows me’. Pre says: ‘if I am an argument in a functional application, my functor precedes me’.

Functional application is realized by two rules in our current system, depending on the order of functor and argument. The easiest way to understand them is probably to look first at their non-unification categorial equivalents:
That is, every constituent has a binary analysis into a functor and an argument, and the only variation is whether the argument precedes or follows the functor. (22) is a formulation which assumes that unification tests for the appropriate specifications.

(22) R1: \( W_1,W_2;C:S \rightarrow W_1;C/E:S \ E(W_2;\text{pres}) \)

R2: \( W_1,W_2;C:S \rightarrow E(W_2;\text{post}) \ W_1;C/E:S \)

Let us look at the interpretation of the first rule: if a functor sign with phonology \( W_1 \), category \( C \), and semantics \( S \) precedes an argument sign \( E \) with phonology \( W_2 \), and order \( \text{pres} \), and if \( E \) is successfully unified with \( X \), then the result is a sign with phonology \( W_1,W_2;C \), and semantics \( S \), where \( C \) and \( S \) may have been altered as a result of unifying \( X \) with \( E \). Exactly the same thing happens with \( R2 \), except that the order of functor and argument is reversed.

2.4. Semantics

The semantic representation language that we use is called InL (for Indexed Language), and is derived from Discourse Representation Theory (cf. Kamp 1981; Heim 1982), supplemented with a Davidsonian treatment of verb semantics (cf. Davidson 1967). The main similarity with the Discourse Representation languages lies in the algebraic structure of InL.

There are only two connectives for building complex formulas: an implication that at the same time introduces universal quantification, and a conjunction. The meaning of an implication like (23),

(23) \[ A(x_1, \ldots, x) \rightarrow B(y_1, \ldots, y) \]

where \( x_1, \ldots, x \) are all the variables in \( A \) outside the scope of any implication occurring in \( A \), and \( y_1, \ldots, y \) the analogous variables in \( B \), can be glossed as the predicate logical formula (24).

(24) \( \forall x_1, \ldots, x \ A(x_1, \ldots, x) \rightarrow B(y_1, \ldots, y) \) \)

A formula as a whole has an existential interpretation; i.e. if

(25) \( A(x_1, \ldots, x) \)

is a formula that introduces the indicated variables outside an implication, it is true precisely if the corresponding predicate logical formula

(26) \( \exists x_1, \ldots, x A(x_1, \ldots, x) \)

is true.

The language InL differs in one important respect from the DRT formalism, and thus earns its name. We assume that every formula introduces a designated variable called its index. This does not mean that (sub)formulas may not introduce other variables, only that the index has a special status. The postulation of indices is crucial for the treatment of modifiers (see section 3.5), but it is independently plausible on other grounds. Consider the expressions in (27), and the ontological type associated with them.

(27) Expression Type

- a. John came to the party event
- b. yesterday an unspecified eventuality
- c. man in the park object
- d. butter quantity of mass
- e. to the party some entity with a direction
- f. came event
g. does not absence

All these expressions can be understood as reporting the existence of some kind of entity, or putting a restriction on some kind of entity. The semantic formulas into which they are translated will carry an index which denotes the reported or restricted entity. The index of a formula is written between square brackets in prenex position. We also adopt the convention that the first variable in the argument-list of an atomic formula is its index; this allows us to omit the prenex index on atomic formulas which occur within a larger expression. (28) shows translations of the expressions in (27).

(28) [\text{Index}] Formula

- a. [e] \( \text{PARTY}(x), [e] \text{TOO(x,x), [e] \text{PAST(e,COME(e,JOHN))}} \)
- b. [e] \( \text{YESTERDAY}(x), [e] \text{A} \)
- c. [x] \( \text{PARK(y), [y] \text{IN(y,x), MAN(x)}} \)
- d. [m] \( \text{BUTTER(m)} \)
- e. [e] \( \text{PARTY}(x), [e] \text{TOO(x,x), [e] \text{A}} \)
- f. [e] \( \text{PAST(e), COME(e)} \)
- g. [e] \( [e] \rightarrow [1] \)

In (28). '1' stands for the necessarily false formula. For notational efficiency, a conjunction whose index is the same as that of its conjuncts will be written as a many-place conjunction. Thus

(29) \( [e][\text{PARTY(e), [e] \text{TOO(e, x), [e] \text{PAST(e, COME(e, JOHN))}}]} \)

is written as (30).

(30) \( [e][\text{PARTY(e), TOO(e, x), PAST(e, COME(e, JOHN)}] \)

Many modifiers or NPs maintain the index of the expression with which they combine; examples are given in (31).
These identities are explicitly expressed in their semantic representations:

\[(31) \text{John to the party } \text{yesterday}\]

Here, \([a]A\) stands for the formula with index \(a\) that translates the argument to which the expression will be applied.

However, the situation is more complex when negation and quantification are involved:

\[(32) \text{John did not come to the party. Every townsman walked in the park last Sunday.}\]

These sentences do not report the event mentioned by \(\text{come}\) or \(\text{walk}\) but state the absence of such an event, or a regularity concerning events of that kind. We take the view, mainly for reasons of simplicity, that both regularities and absences are static eventualities of a special kind. Formally, these are realised by a static index which is introduced by the implications that translate both \(\text{every townsman}\) and \(\text{did not}\).

\[(33) \text{[s](TOWNSMAN(x) \rightarrow [a]A)} \text{[s](PAST(t), [s](aJ \rightarrow JLH}\]

The different ontological types mentioned earlier are formalized by dividing semantic variables into sorts. The regime for sorted variables is one where the sort is a bundle of features associated with a particular variable or referential constant. In this way, unifications can be performed on sorts. This is useful, since it provides a way of expressing selectional restrictions (cf. section 3.2), and allows the sort of a variable to be determined by different references to it by different subexpressions. Since feature bundles clutter up the notation, we use special variable letters for some standard sorts, or use abbrevatory labels on a variable where this is suitable. The list (35) associates variable letters with particular sorts.

\[(34) \text{[s](TOWNSMAN(x) \rightarrow [a]A)} \text{[s](PAST(t), [s](aJ \rightarrow JLH}\]

Furthermore, for each of the above sorts, and for others not listed, we assume that we can write labeled declarations as in (36).

\[(35) \text{[variable] \text{[semantics]-> semantics}}\]

3. A Fragment

In this section, an attempt will be made to present a fairly large part of the UCG fragment we have been working on. After what has been discussed above, it will be clear that this is mostly a question of stating the lexicon. As is customary in unification grammars, the lexicon consists of a set of primitives and a number of lexical rules working on those primitives to produce the full lexicon. (37) recapitulates the notion of sign described in the first section by describing the syntax and associated variables:

\[(37) \text{sign} \rightarrow \{\text{phonology}: \text{category}: \text{semantics}: \text{order}, \text{E}\} \rightarrow \{\text{string}, \text{W}\} \rightarrow \{\text{bse}, \text{cbse}, \text{inf}, \text{fin}, \text{pnp}, \text{ppp}, \text{pas}, \text{obj}, \text{nom, to, by, of, for, F}\} \rightarrow \{\text{atom}, [\text{variable}][\text{semantics} semantics], \text{S}, [\text{aS}] \rightarrow \text{predicate} [\text{arg}]\} \rightarrow \{\text{variable, constant, semantics}\} \rightarrow \text{sort number}\]
Although proper names could be treated by letting the verb be a functor that takes the name as argument, the next two examples show that such a scheme does not work for NP's in general. The semantics of the NP combined with a verb derives in these two cases from the NP, and the semantics of the verb only fills a slot in the resulting representation. Moreover, we observe a fundamental principle in our grammar, namely that whenever two signs are combined, the semantics of the result is always derived by instantiation from the semantics of the functor. This principle compels us to treat the NP as the functor.

(39) a. a man
   C(C[np:nom or obj:singular(b):O]:a):S:0
   [a][MAN(x), [a]S]

   b. every woman
   C(C[np:nom or obj:singular(b):O]:a):S:0
   [state(s)][WOMAN(x) => [a]S]

The next two examples involve finite verbs. Inflected verb forms are not listed as basic items in the lexicon, but are derived from a root form by lexical rule.

(40) a. walks
    sent[fin]/np[nom]:x:pre
    [e][PRESENT(e), WALK(e, singuiar(x))]

   b. love
    sent[fin]/np[nom]:x:pre/np[obj]:y:post
    [s][PRESENT(s), LOVE(s, x, y)]

The next example shows a phrase composed of an auxiliary and base-form verb:

(41) does not walk
    sent[fin]/np[nom]:x:pre
    [s][PRESENT(s), [s][WALK(e, x) => ]]

We also can use the signs above to derive more complex constructions:

(42) a. Louise walks
    sent[fin]
    [e][PRESENT(e), WALK(e, LOUISE)]

   b. loves every woman
    sent[fin]/np[pre][obj]:x:pre
    [s][WOMAN(y) => [s][PRESENT(s), LOVE(s, x, y)]]

   c. Louise loves every woman
    sent[fin]
    [s][WOMAN(y) => [s][PRESENT(s), LOVE(s, LOUISE, y)]]

   d. a man does not walk
    sent[fin]
    [s][MAN(x), [s][PRESENT(s), [s][WALK(e, x) => ]]]

3.2. Expressing Combinatorial Restrictions in UCG

UCG offers a number of devices to prevent the application of one sign to another. The most fundamental one is built into the formalism of categorial grammar, according to which the active part of one sign's category must match the other sign's category. The fact that this combinatorial restriction is expressed in terms of unification does not lead to any significant difference.

We have already noted that the categorial system can be refined by allowing further specification of primitive categories by features. The use of features in this way is standard practice in generative grammar, and should not require further justification.

Less common, and one of the interesting aspects of UCG, is the method of imposing restrictions at the level of semantics. If it not possible to construct a new semantics by unification, the derivation is blocked. This resource is particularly useful for dealing with agreement. Thus, a string like

(43) *The boys walks
is ruled out because the variable for the subject in the sign for walks has sort singular, whereas the boys introduces a plural variable, and variables with distinct sorts cannot be unified.

The same mechanism can be used in an example like (44).

(44) *Mary likes to wash himself

The subject Mary is lexically marked as having sort female, and thus cannot be unified with

---

This option is also readily available in frameworks like HPSO and PATR-II.
the variable \( x \) in (45).

\[ (45) \quad [s][L1KE(s, \text{male}(x), [e]\text{WASH}(e,x,x))] \]

Finally, consider the observation that the temporal modifier \textit{in an hour} can only be combined (at least in one use) with predicates which are aspectually marked as introducing a completed event. This can be captured by assigning the index of \textit{in an hour} the sort of completed events. As a result, we can successfully distinguish between the following two examples:

(46) a. John cleaned the garden \textit{in an hour}

b. *John was working in the garden \textit{in an hour}

The treatment of subject-verb agreement by means of semantics is of course rather controversial, given the distinction that is often drawn between 'natural' and 'grammatical' gender and number (cf. Corbett 1979, 1981; Cooper 1983). Certainly, it may be argued that these agreement categories are more deeply grammaticised in languages other than English. The evidence for treating number as syntactic - or rather 'non-natural' - in English rests on a small handful of examples like (47), where plural morphology and agreement is associated with NPs whose referents are not typically conceptualised as plural objects.

(47) The scissors are\textit{is} sharp
The oats are\textit{is} in the bin

A slightly different case arises with certain collective nouns in British English, which despite singular morphology sometimes trigger plural agreement:

(48) The committee \textit{meets} at 2.00 on Wednesday

One could maintain a solely semantic account of such cases, and still take into account their anomalous status, by allowing a slightly more complex semantic representation as in (49).

(49) a. \[ [a]\text{SCISSORS(plural(a)), COINCIDE(\text{asingular(b)})} \]

b. \[ [a \text{or } b]\text{COMMITTEE(singular(a), COINCIDE(plural(b),a)} \]

(49a) renders the index of \textit{scissors} necessarily plural, but captures the intuition that the object denoted is in some sense singular by relating the plural index to a singular variable. In (49b) \textit{committee} is assigned a 'disjunctive index', and we can choose between the standard singular variable, and the associated plural variable. This allows for both types of agreement, and makes \textit{committee} a potential antecedent for both singular and plural pronouns. What we say in such cases is that there are two different objects: a plural one and a singular one. Such associated objects, though nonidentical, also coincide with each in the sense of sharing the matter of which they are composed.\(^4\)

There would clearly be no formal difficulty in extending our feature system so as to allow a syntactic analysis of gender and number in a language like German. But even here it may be interesting to think of the syntactic gender as defining an object in a sort, even if one does not take the objects in this sort very seriously. Thus, referring to a girl by the German \textit{Maedchen} makes a literal reference to a coinciding neuter object. There may be ontological objections against this approach, but it has the advantage of accounting in a uniform manner for the fact that anaphoric links to an antecedent NP such as \textit{das Maedchen} can be established on the basis of either natural or grammatical gender.\(^5\)

3.3. Extending the Fragment

In this section, we try to sketch the underlying principles which might be used to extend the fragment. The procedure is based on the constraints inherent in categorial syntax: once certain basic categorizations are imposed, combinatorial considerations largely dictate the categorization of other expressions. We will run through two more complicated examples, and in the course of that arrive at notions of the category of determiner, noun, auxiliary and controlled complement. The analyses we suggest are not intended to be definitive, but serve to illustrate a particular working methodology.

The first example shows a fairly plausible representation for a raising-to-object construction, where the NP \textit{a student} is assigned wide scope.

(50) John believes \textit{a student} to have cheated.

Assuming that this is formed by functional application of the subject, \textit{John}, we obtain the following analysis for the predicate:

(51) believes \textit{a student} to have cheated.

It has been customary in monostatal approaches to English syntax to assume that \textit{a student to have cheated} does not form a constituent. Given our treatment of linear order, this leads

\(^4\)See Link (1983) for some model-theoretic reflections on this notion of coincidence.

\(^5\)In addition to the references cited earlier, see also Johnson (1983), Tascowicz-De Rijck and Verheyen (1981), and Wiese (1986).
us to derive the two signs in (52) from (51), where ‘Z’ is used as a temporary place-holder.6

(52) a. believes a student
   sent[fin]/np[nom]:y:pre/Z
   [s][STUDENT(x), PRESENT(s), BELIEVE(s, y, [s]S)]

b. to have cheated
   Z
   [[AFTER(t, e), CHEAT(e, x)]

Let us now try to spell out what constraints should be imposed on Z. To begin with, we note that only to-infinitives are syntactically permissible as arguments to (52a). This category is encoded by adding a feature specification [cbse] to the sent symbol that marks verbal heads. Second, infinitives areanalysed as being unsaturated; otherwise their subject position in the semantics would not be free for control by the matrix object. Third, the schema [s]S in the semantics of (52a) has to be cross-identified as the semantics of the active sign in (52a)’s category. Fourth, in order to express object control, we want the subject of the infinitive to bind the same variable as STUDENT does. This leads us to replace (52) by the following:

(53) a. believes a student
   sent[fin]/np[nom]:y:pre/(sent[cbse]/x):[s]S:pre
   [s][STUDENT(x), PRESENT(s), BELIEVE(s, y, [s]S)]

b. to have cheated
   Z
   [[AFTER(t, e), CHEAT(e, x)]

It seems plausible to derive (53b) from the combination of to with a naked infinitive. Since some verbs are categorised for naked infinitives complements, they must be recognisable as such, and we use the feature specification [bsc] for this purpose.

(54) to
   (sent[cbse]/x)(sent[bsc]/x):S:pre
   S

To only changes the feature specification from [bsc] to [cbse]. The naked infinitive accordingly has the sign

(55) have cheated
   sent[bsc]/x
   [[AFTER(t, e), CHEAT(e, x)]

It is easiest to let the auxiliary have (here in its infinitival form) carry the semantic effect of the perfect. This makes it possible to treat both the passive and the past participle in the same way. So have gets definition (56):

(56) have
   sent[bse]/np[nom]:x:pre/(sent[pip]/x):[s]A:pre
   [t][AFTER(t, a), [s]AT]

For the participle cheated we obtain (57).

(57) cheated
   sent[ppp]/np[nom]:x:pre
   [e]CHEAT(e, x)

Returning to believe a student, it will be recalled that indefinite NPs were already introduced in the previous section.

(58) a student
   C(Cnp[nom or obj]:x:[a]S:O)
   [s][STUDENT(x), [a]S]

Believe must therefore be defined as in (59).

(59) believes
   sent[fin]/np[nom]:y:pre/(sent[cbse]/x):[s]S:pre/np[subject]:x:post
   [s][PRESENT(s), BELIEVE(s, y, [s]S)]

Note that the variable introduced by the object NP only appears as the subject of the infinitive. From a student we can easily reconstruct the determiner a and the common noun student:

(60) a(n)
   C(Cnp[nom or obj]:singular(b).*0):[a]R:O
   [a]R

b. student
   noun
   STUDENT(x)

As a second example, consider the complex nominal cruel farmer who beats a donkey

(61) cruel farmer who beats a donkey

It is fairly clear what this expression should mean, and we propose the sign (62).

(62) cruel farmer who beats a donkey
   noun
   [x]CRUEL(x), [x]FARMER(x), [e]DONKEY(y), PRESENT(e), BEAT(e, x, y)]

This can be constructed either by applying the adjective to the complex noun, or by applying the relative to cruel farmer. Since it does not make any difference, let’s start with the adjective. Adjectives apply to nouns to yield nouns. So cruel has the following sign:

(63) cruel
   noun/noun:[x]A:pre
   [x]CRUEL(x), [x]A

For the noun, we are left with (64).
The relative clause is rather similar to the adjective, as appears from (64.)

This leaves us with the syntax of the relative clause. The analysis proposed is simple but only covers the simplest case; we shall not attempt here to deal with unbounded dependencies, though various approaches are compatible with our theoretical framework (cf. Pollard 1985a,b; Sneedman 1985a,b). Who combines with the finite verb phrase (66).

It must therefore have definition (67).

3.4. The Verbal Paradigm

The featural distinctions within the verbal paradigm have a number of functions. On the one hand, they affect the distribution of phrases with a verbal head, and on the other hand they are associated with operations that change the morphological realization, the categorization and the semantics of those verbal heads. Following fairly standard lexicalist assumptions, the operations all apply to lexical stems. Any member of a verb paradigm can therefore be decomposed into a stem together with a specification of some of the operations defined in (70) below. A simple example paradigm is illustrated in (68).

The lexical rules we use are modelled on those in Shieber (1983) and have the general form indicated in (69):

That is, they map signs into signs, and we allow them to modify any aspect of the input.
33. Modifiers

One of the advantages of categorial syntax over X-bar syntax is that it allows a general characterization of modifiers, namely as any expression of category A/A. This translates into our framework as the sign

\[ X7X: [a]S \]

As we saw earlier, attributive adjectives can be obtained from the general definition by instantiating X to the category of common nouns:

\[ \text{noun/noun:}[x]A:pre \]

The normal distinction between intersective, relative and intensional adjectives can be made (cf. Kamp 1975).'

\[ \text{a. square} \]
\[ \text{noun/noun:}[x]A:pre \]
\[ [x][\text{SQUARE}(x), A] \]

\[ \text{b. big} \]
\[ \text{noun/noun:}[x]A:pre \]
\[ [x][\text{BIG}(x), A] \]

\[ \text{c. fake} \]
\[ \text{noun/noun:}[x]A:pre \]
\[ [x][\text{FAKE}(x), A] \]

As is well known, these same distinctions are typically expressed by meaning postulates in

\[ (73) \text{noun/noun:}[x]A:pre \]

Montague Grammar. For example, the intersective nature of square might be expressed by stipulating the logical validity of (68.)

\[ (75) \text{VQVx}(\text{square}(x) \leftrightarrow \text{square}(x) \land Q(x)) \]

However, such a strategy seems to depend on the fact that the common noun argument, indicated by the variable 'Q' on the left-hand side of (75), denotes a function from objects to truthvalues, and can hence appear in an independent predication on the right-hand side of the postulate. In a standard Montagovian approach, there is no obvious way of distinguishing between analogous classes of predicate- or sentence-modifiers. By contrast, the combination of a Davidsonian treatment of verb meanings with the InL theory of indices gives rise to a completely uniform treatment of such modifiers. \[ \text{Predicate adverbs are obtained by instantiating the C in schema (72) to sent/np, as illustrated below. (76a) and (76b) are intensional, (76c) is relative.} \]

\[ (76) \text{a. always} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{HABIT}(a, [a]S) \]

\[ \text{b. never} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{[a]HABIT}(a, [a]S) \rightarrow \text{false} \]

\[ \text{c. quickly} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{QUICK}(a, [a]S) \]

\[ \text{Possibly} \]
\[ \text{C(sent/np):}[a]S \]
\[ \text{POSSIBLE}(state(a), [a]S) \]

We regard most adverbial phrases as being a species of prepositional phrase, following Emonds (1976). The following illustrates some representative prepositions.

\[ (78) \text{in} \]
\[ X/X: [a]S:np[obj]:x:post \]

\[ \text{Note:} \]
\[ \text{noun/noun:}[x]A:pre \]
\[ [x][\text{ROUND}(x), A] \]

\[ \text{Montague Grammar. For example, the intersective nature of square might be expressed by stipulating the logical validity of (68.)} \]

\[ (75) \text{VQVx}(\text{square}(x) \leftrightarrow \text{square}(x) \land Q(x)) \]

\[ \text{However, such a strategy seems to depend on the fact that the common noun argument, indicated by the variable 'Q' on the left-hand side of (75), denotes a function from objects to truthvalues, and can hence appear in an independent predication on the right-hand side of the postulate. In a standard Montagovian approach, there is no obvious way of distinguishing between analogous classes of predicate- or sentence-modifiers. By contrast, the combination of a Davidsonian treatment of verb meanings with the InL theory of indices gives rise to a completely uniform treatment of such modifiers.} \]

\[ \text{Predicate adverbs are obtained by instantiating the C in schema (72) to sent/np, as illustrated below. (76a) and (76b) are intensional, (76c) is relative.} \]

\[ (76) \text{a. always} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{HABIT}(a, [a]S) \]

\[ \text{b. never} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{[a]HABIT}(a, [a]S) \rightarrow \text{false} \]

\[ \text{c. quickly} \]
\[ \text{C(sent/np):}[a]S:post \]
\[ \text{QUICK}(a, [a]S) \]

\[ \text{Possibly} \]
\[ \text{C(sent/np):}[a]S \]
\[ \text{POSSIBLE}(state(a), [a]S) \]

\[ \text{We regard most adverbial phrases as being a species of prepositional phrase, following Emonds (1976). The following illustrates some representative prepositions.} \]

\[ (78) \text{in} \]
\[ X/X: [a]S:np[obj]:x:post \]

\[ \text{Note:} \]
\[ \text{noun/noun:}[x]A:pre \]
\[ [x][\text{ROUND}(x), A] \]
As noted earlier, we adopt the view of Gazdar et al. (1985) that prepositions in English are also used as a kind of case-marking on a noun phrase. We illustrate this analysis with to:

\[
(79) \quad X(X/Np[0];:x;O);[a;S:O/mp[0];:x;post
[a;S]
\]

4. Conclusion and Comparisons

UCG exhibits a number of similarities with other formalisms in the unification framework. The foremost amongst these is monotonicity, in the sense that information, once gained, is never lost in the course of a derivation. From a purely theoretical vantage point, this has the effect of rendering impossible many analyses which are compatible with a standard transformational framework: it is not possible to postulate an intermediate representation which is then subject to destructive modification. Principles like the Well-Formedness Constraint of Partee (1979) largely fall out on such an approach. Monotonicity also has practical advantages, in that it allows for a more deterministic architecture in parsing.

A further attractive feature of UCG, which it shares with some other approaches, is the manner in which different levels of representation - semantic, syntactic and phonological - are built up simultaneously, by the uniform device of unification. This is not to deny that there are different organizing principles at the different levels. For example, the operations corresponding to conjunction and implication exist at the semantic level, but not at the syntactic or phonological. Nevertheless, the compositional construction of all three levels takes place in the same manner, namely by the accretion of constraints on the possible representations. The schematic variables that we employ stand for a maximally unspecified representation. As the variables become unified with constants in the course of a derivation, more and more constraints are placed on the representation until we end up with a fully specified structure which admits of only one interpretation.\(^{11}\)

Although we have said nothing of interest about phonology here, it seem plausible, in the light of Bach and Wheeler (1981) and Wheeler (1981), that the methodological principles of compositionality, monotonicity and locality can also lead to illuminating analyses in the domain of sound structure. Moreover, it is interesting to note that our manipulation of indices in semantics bears certain resemblances to the specification of an autosegment in phonology (see, for example, Goldsmith 1976), and it should be possible to use the formal techniques of unification grammar in multi-tiered phonological representations.\(^{12}\)

UCG is distinctive in the particular theory of semantic representation which it espouses. As we have already mentioned, InL is based on Kamp’s Discourse Representation (DR) formalism. Two incidental features of InL may obscure this fact. The first is very minor: our formulas are linear, rather than consisting of ‘box-ese’. The second difference is that we appear to make no distinction between the set of conditions in a DR, and the set of discourse markers. In fact, this is not the case. Every InL formula has a major discourse referent, namely the index. However, within a complex condition, the discourse referents are not grouped together into one big set, but are instead prefixed to the atomic formula that was responsible for introducing the marker in question. A simple recursive definition (similar to that for ‘free variable’ in predicate logic) suffices to construct the cumulative set of discourse markers associated with a complex condition.\(^{13}\) These departures from the standard DR formalism do not adversely affect the insights of Kamp’s theory, but do offer a substantial advantage in allowing a rule-by-rule construction of the representations, something which has evaded most other analyses in the literature.

A third respect in which InL differs from standard expositions of DR theory is in the use of polymorphic functions. Recent discussion of polymorphism within a Montague framework (e.g. Partee forthcoming) has concentrated on functions which are generic with respect to the types of Montague’s higher-order logic. In UCG, the issue of type shifting does not arise in quite the same way, since the integration of semantics into (sub)categorization allows us to keep InL largely first order.\(^{14}\) On the other hand, the logic is multi-sorted, with the sorts organized hierarchically so as to form a subsumption lattice. This renders the polymorphism of UCG functions closer in conception to the usual situation

---

11For more discussion of this general point, see Freund et al. (1985)
12This would go some way towards vindicating the conviction expressed by van Riemsdijk (1982) that phonologists and syntacticians should take "more notice of each other's work.
13Horn and Klein (1986) present a method for implementing Kamp-style pronoun resolution rules in a unification grammar, though they use a rather more standard syntax for DR.
in typed programming languages (cf. Tennent 1981, for example).

The effect of polymorphism is perhaps even more striking in syntax. While it is common to use meta-variables in categorial grammar, there have been few attempts to exploit variables in the categories themselves. UCG syntax is heavily polymorphic in the sense that the category identity of a function application typically depends on the make-up of the argument. Thus, the result of applying a type-raised NP to a transitive verb phrase is an intransitive verb phrase, while exactly the same functor applied to an intransitive verb phrase will yield a sentence. Analogously, a prepositional modifier applied to a sentence will yield a sentence, while exactly the same functor applied to a noun will yield a noun. This approach allows us to dramatically simplify the set of categories employed by the grammar, while also retaining the fundamental insight of standard categorial grammar, namely that expressions combine as functor and argument. Such a mode of combination treats head-complement relations and head-modifier relations as special cases, and provides an elegant typology of categories that can only be awkwardly mimicked in X-bar syntax.

Finally, we note one important innovation. Standard categorial grammar postulates a functor-argument pair in semantic representation which parallels the syntactic constituents; typically, lambda-abstraction is required to construct the appropriate functor expressions in semantics. By contrast, the introduction of signs to the right of the categorial slash means that we subsume semantic combination within a generalised functional application, and the necessity of constructing specialised functors in the semantics simply disappears.

Appendix 1: Two Sample Derivations

In the following two examples, we use the notation 'dbc', etc., to indicate a sign which is derived from the signs labelled 'd', 'b', and 'e'.

(A1) Suzy likes to walk with every man.

a. suzy
Cc[np[nom or obj]:SUZY:O]:[a]S

b. every
Cc[np[nom or obj]:singular(b):O]:[a]S
noun[b]:O
noun[b]:pre

14We say "largely", because the question of how to deal with modal contexts still remains unresolved.

c. man

Unification Categorial Grammar

(A2) Often John visits a cinema

a. often
Cc[np[nom or obj]:JOHN:O]:[a]S

b. John
Cc[np[nom or obj]:JOHN:O]:[a]S

15cf. to walk:CBSE
sent[cbse]x
[e]WALK(e)
Appendix 2: UCG in PATR-II

UCG was developed as the grammatical basis for a parser formulated in PATR-II (Shieber et al. 1983), and has been implemented in C-PROLOG running under UNIX on a VAX11/750. While other ways of implementing a UCG parser can certainly be envisaged, it is worth noticing the close affinities: the basic signs discussed in the last section can be seen as PATR-II lexical entries, the rules in the section on the verb paradigm can be seen as PATR-II lexical rules, and the functional application rules can be seen as PATR-II syntactic rules. In this Appendix, we give a PATR-II version for one example of each of these.

(A3) Lexical Entry

(a) UCG:

\[
\text{love} :: \text{sent}(\text{np}[\text{nom}]:x: \text{pre} / \text{np}[\text{obj}]:y: \text{post})
\]

(b) PATR-II:

phonology = love
syntax : head = sent
calist : first : syntax : head = np
feature = obj
: semantics = semantics : arglist : rest : first
rest : first : syntax : head = np
: feature = nom
: semantics = semantics : arglist : first
rest = nil
semantics : predicate = love
index : sort = state
arglist : rest : rest = [ ]

A PATR-II lexical rule constructs a DAG under the label \textit{out}, with the same phonology as the sign under the label \textit{in}, and is not conceived as a transformation on the stem.

(A4) Lexical Rule

(a) UCG

\[
3sg\_\text{pres}:
\]

\[
W \rightarrow W+s
\]

\[\text{sent}(\text{fin})/\text{sent}^\text{fin}: \text{sent}^\text{fin}(\text{np}[\text{nom}]:x: \text{pre} / \text{np}[\text{obj}]:y: \text{post})
\]

\[\text{state}(a)[\text{AT}(a, \text{NOW}), S]
\]

(b) PATR-II

out : syntax : head = in : syntax : head
out : syntax : head = sent
: feature = fin
out : semantics : predicate = conjunction
: index = in : semantics : index
: arglist : first : predicate = present
: index = in : semantics : index
: arglist = nil
: rest = in : semantics

A PATR-II syntax rule consists of a PS rewrite rule together with a number of equations.
Syntax Rule

(a) UCG
R1: $W_1W_2:CS \rightarrow W_1:CES E(W_2:pre)$

(b) PATR-II
\[
\begin{align*}
c_1 & \rightarrow c_2 c_3, \{ \begin{array}{l}
  \text{c2:catlist:rest} \\
  \text{c1:syntax} \\ 
  \text{c1:semantics} \\
\end{array} \\
  = \begin{array}{l}
  \text{c2:catlist:rest} \\
  \text{c2:syntax} \\
  \text{c2:semantics} \\
\end{array}
\end{align*}
\]


