Unification Categorial Grammar

by Henk Zeevat (Amsterdam), Ewan Klein (Edinburgh)
and Jo Calder (Univ. of Arizona)

1. *Setting the Scene*

Unification categorial grammar (UCG) is a version of categorial grammar enriched by several insights from Head-driven Phrase Structure Grammar (Flickinger, Pollard and Wasow, 1985; Pollard, 1985; Proudian and Pollard, 1985 and PATR-II Shieber, 1986; Shieber et al. 1986) [1]. 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 a desire to develop the theory in a manner that was sufficiently precise and detailed to allow testing and implementation in a computational environment.

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:

The work reported here was carried out as part of ESPRIT Project P393 (ACORD). The Construction and Interrogation of Knowledge Bases using Natural Language Text and Graphics. The technical report from which the current paper is derived is Problems of Dialogue Parsing, *ACORD deliverable T2.1, by the current authors and Marc Moens. This version was prepared as a result of a course on UCG given by the authors in collaboration with Claudia Casadio and Antonio Sanfilippo at the University of San Marino in March 1991, for the International Centre for Semiotic and Cognitive Studies. We are grateful to the following people for comments and criticism: Karine Baschung, Gabriel Bes, Bob Carpenter, Annick Corluy, Robert Dale, Thierry Guillotin, Einar Jowsey, Marc Moens and Glyn Morrill. All errors are of course our own.*

[1] 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 and unification grammar.

(1) a. N and S are categories
b. If A and B are categories, A/B is a category.

Functional application is the following rule:

(2) If E_1 is an expression of category A/B and E_2 is an expression of category B, then $E_1\ E_2$ (i.e. the concatenation of E_1 and E_2) is an expression of category A.

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.

For applications to natural language, various extensions of this scheme have been proposed[2]. UCG is just one of these extensions, where the notion of a category is expanded. We assign to each expression a number of representations specifying its properties at the different levels of linguistic analysis. Most importantly, these are:

1. the way in which the expression is phonologically realised (its orthography for our current purposes)
2. a category specification
3. a semantic representation

Following Pollard (1985), 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 a category. It will be observed that (3.b) is just the categorial analog of Pollard's (1985) proposal for subcategorization, according to which phrasal heads are specified for a list of signs corresponding to their complements.

[2] For example, directional categories, Montague grammar (where a notion of syntactic rule subsumes functional application), and combinatory grammar (cf. Geach, 1972; Lambek, 1958, 1961; Montague, 1973; Steedman, 1985 and Van Benthem, 1986).

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 that we call *instantiation* and *stripping*. It will be recalled that if a sign has category A/B, then B is said to be 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 S_1's 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_2 just like S_1 except that its phonology is the concatenation of S_1's 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 well-formed 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

(7)
W
C
S
O

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

(8) W:C:S:O

(9) illustrates a typical case, the lexical entry for the verb *visit*:

(9)
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 $A/S_1/S_2$ (i.e. $(A/S_1)/S_2$, assuming association to the left), where S_1 and S_2 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 difficulties. Thus, the example above can be reduced slightly as follows:

(10)
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

(11) 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 et al. (1985).

	Features	Morphology
on *sent*:		
	FIN	finite verb form
	CFIN	complementized finite verbal element
	BSE	base verb form (i.e. a bare infinitive)
	CBSE	complementized base verb form
	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 beween 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 argument expression.

Information flows whenever unification occurs, and since unification is commutative, the flow can go in either direction. We illustrate this with a simple example. (12) is a lexical entry for the verb *walk*.

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

(13) is plausible as a lexical entry for a proper name (though in fact we adopt a slightly different treatment, to be discussed below):

(13) john
np
JOHN

Now suppose we try to unify the active sign

(14) np[nom]:x:pre

with (15). In order to see what is going on more clearly, let's use a uniform format which includes all the variables:

(15) a. john
np
JOHN
O

b. W
np[nom]
x
pre

What results from unification of these two is the sign (16).

(16) john
np[nom]
JOHN
pre

The value for phonology is contributed by (13), as is the semantics, *JOHN*, while a further specification of *np* is contributed by (14), as is a value for the order attribute. As a result, we obtain the following instantiation of (12):

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

Notice that as side-effect of instantiation, the semantics has been further specified. It can now be interpreted as saying that there is an event *e* in which John – not some anonymous *x* – walks.

The argument sign is now marked by the order declaration *pre*, meaning that functional application only succeeds if *john* comes after *walks* in the phonology after functional application. The role of the order attribute will be explicated in the next section.

Now that we have instantiated (12) in (17), it can be stripped, yielding (18) as a result.

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

The most spectacular changes that instantiation can induce are to be found when unification specifies the result category in the functor sign. For well-known semantic reasons, we follow Montague (1973) and others in assigning *noun* phrases a type-raised category. Our notion of type-raising is slightly more general than usual, since we allow category variables. Thus, our lexical entry for *John* looks like (19) (rather than (13)):

(19) john
C/(C/np:JOHN:O):S:O
S

The active sign

(20) (C/np:JOHN:O):S:O

contains a complex category *C/np:JOHN:O*. This can be unified with the sign for *walk* we gave above, yielding (21).

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

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 (12) with respect to (13) occur here as well. Our original expression (19) has been transformed into (22) as a result of the unification.

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

Functional application can now yield (23).

(23) 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 is 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), Karttunen (1986), Steedman (1985), Uszkoreit (1985, 1986a), Reape (1989) develops – considerably generalising some earlier ideas of Klein – a constraint-based theory of word order for HPSG, that would be a good basis for developing a better word order treatment in UCG, a development that falls outside the scope of this paper.

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. *R1* allows a functor to apply to a constituent of category *B* to its right, while *R2* allows application to a constituent *B* to the left:

(24) R1: A → A/B B
R2: A → B B\A

(25) is a formulation which assumes that unification tests for the appropriate order specifications on the arguments themselves.

(25) R1: W_1W_2:C:S → W_1:C/E:S E(W_2:pre)
R2: W_2W_1:C:S → E(W_2: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/X and semantics S precedes an argument sign E with phonology W_2, and order *pre*, and if E is successfully unified with X, then the result is a sign with phonology W_1W_2, category 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 and 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 (26),

(26) $[A(x_1, \ldots, x_n) \Rightarrow B(y_1, \ldots, y_k)]$

where $x_1, \ldots, x_n$ are all the variables in A outside the scope of any implication occurring in A and $y_1 \ldots y_k$ the analogous variables in B, can be glossed [3] as the predicate logical formula (30).

(30) $\forall x_1 \ldots x_n\, [A(x_1 \ldots x_n) \rightarrow \exists y_1 \ldots y_k B(y_1 \ldots y_k)]$

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

(31) $A(x_1 \ldots x_n)$

is a formula that introduces the indicated variables outside an implication, it is true precisely if the corresponding first order logic formula

(32) $\exists x_1 \ldots x_n\, A(x_1 \ldots x_n)$

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 (cf. Zeevat, 1991, ch. 4). Consider the expressions in (33), and the ontological type associated with them.

[3] A more explicit version is given by defining the set of discourse markers for the InL formulas as follows:
(27) $DM([a]A) = \{a\}$, for atomic formulas
$DM([a][B, C]) = DM(B) \cup DM(C) \cup \{a\}$
$DM([a][B \Rightarrow B] = \{a\}$
and to define the correspondence with predicate logic in terms of a function *trans* mapping InL-formulas into first order logic formulas, which has a clause for implication as in (28)
(28) $trans([a][B \Rightarrow C]) = \forall D\, M(B)(trans(B) \rightarrow \exists\, DM(C)\, trans\,(C))$
and one for the global formula stated in (29).
(29) $trans\, f\, ([a]A) = \exists\, DM([a]A) trans([a]A)$

(33)

	Expression	*Type*
i.	John came to the party	event
ii.	yesterday	an unspecified eventuality
iii.	man in the park	object
iv.	butter	quantity of mass
v.	to the party	some entity with a direction
vi.	came	event
vii.	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. (34) shows translations of the expressions in (33).

(34)

	[*Index*]	*Formula*
i.	[e]	[PARTYx,[e][TO](e,x),[e][PAST(e),COME(e,JOHN)]]]
ii.	[a]	[YESTERDAY(a),[a]A]
iii.	[x]	[PARK(y),[x][IN(x,y), MAN(x)]]
iv.	[m]	BUTTER(m)
v.	[a]	[PARTY(x),[a][TO(a,x),[a]A]]
vi.	[e]	[PAST(e),COME(e)]
vii.	[s]	[A ⇒ ⊥]

In (vii), ⊥ 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

(35) [e][PARTY(x), [e][TO(e, x), [e][PAST(e), COME(e, JOHN)]]]

is written as (36).

(36) [e][PARTY(x), TO(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 (37).

(37) to the party
John
yesterday

These identities are explicitly expressed in their semantic representations:

(38) [a][PARTY(x), TO(a,x), [a]A]
[a]A
[a][YESTERDAY(a), [a]A]

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:

(39) John did not come to the party.
Every citizen walked in the park last Sunday.

These sentences do not report the event mentioned by *come* or *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 stative eventualities of a special kind. Formally, these are realised by a stative index which is introduced by the implications that translate both *every citizen* and *did not*.

(40) [s] [CITIZEN(x) ⇒ [a] A]
[s] [PAST(s), [s][[a]A ⇒ ⊥]]

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 abbreviatory labels on a variable where this is suitable. The list (41) associates variable letters with particular sorts.

(41)

object variables	x, y, z, x_1, x_2, x_3, . . .
mass variables	m, m_1, . . .
event variables	e, e_1, e_2, e_3, . . .
state variables	s, t, s_1, s_2, s_3, . . .
unsorted variables	a, b, c, a_1, a_2, a_3, . . .

Furthermore, for each of the above sorts, and for others not listed, we assume that we can write labeled declaration as in (42).

(42) state(a)
plural(a)
female(x)
singular(a)

Sorts are related by a partial ordering which corresponds to the subset relation on the sets of objects semantically associated with the sorts. Thus, for example, "mass" and "count" are subsorts of "ob-

ject". However, the precise specification of this hierarchy (or lattice) goes beyond the scope of the present paper.

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 specifying 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. (43) recapitulates the notion of sign described in the first section by describing the syntax and associated variables:

(43)

sign	→	{phonology: category: semantics: order, E}
phonology	→	{string, W}
category	→	{sent[feature], np[feature], noun, category/sign, C}
feature	→	{bse, cbse, inf, fin, cfin, psp, prp, pas, obj, nom, to, by, of, for, F}
order	→	{pre, post, O}
semantics	→	{atom, [variable][semantics, semantics], [variable][semantics ⇒ semantics], S, [a]S}
atom	→	predicate(arg*)
arg	→	{variable, constant, semantics}
variable	→	sort integer

(44)

R1:	W_1W_2:C:S	→	W_1:C/E:S	E(W_2:pre)
R2:	W_2W_1:C:S	→	E(W_2:post)	W_1:C/E:S

3.1. *The Basic Case: Finite Verbs and Simple NPs*

The following three examples illustrate some simple NPs from the fragment. The category assigned to NPs is of the form

(45) C/(C/np)

This says I want to combine with anything that wants an *np*, and I'll yield something that no longer wants that *np*. (46) illustrates the case of a proper name:

(46) Louise
C/(C/np[nom or obj]:LOUISE:O):[a]S:O
[a]S

In this case, the resulting semantics is the semantics of the NP's argument expression, as modified by unification: the unspecified argument associated with *np* will be bound to the constant LOUISE.

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 form 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.

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

b. every woman
C/(C/np[nom or obj]:singular(b):O):[a]S:O
[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 (or by a more general system for lexical description, cf. Pollard & Sag, 1988, ch. 7; Flickinger, 1987; Calder 1991).

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

(49) 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:

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

b. loves every woman
sent[fin]/np[nom]: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.

These are best viewed as a generalization of the notion of case in traditional grammar. The fact that a NP is morphologically marked for e.g. nominative makes a combination impossible with a verb that is categorised for an accusative NP, even though the purely categorial specifications do not prevent the combination. Regarding certain PPs as NPs marked for a case introduced by the preposition to have them combine with verbs that take PP-complements is a natural extension. Similarly, if something is a passive participle it cannot become a finite sentence by combining with a subject, or combine with the wrong auxiliaries, even though there is no purely categorial or semantical reason for this. The feature for passive is therefore like case in excluding certain combinations and favouring others.

Less common, and one of the interesting aspects of UCG, is the method of imposing restrictions at the level of semantics [4]. If it is 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

(51) *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 these distinct sorts cannot be unified.

The same mechanism can be used in an example like (52).

(52) *Mary likes to wash himself.

The subject *Mary* is lexically marked as having sort *female*, and thus cannot be unified with the variable *x* in (53).

(53) [s][LIKE(s, male(x), [e]WASH(e,x,x))]

[4] This option is also readily available in frameworks like HPSG and PATR-II.

Finally, consider the observation that the temporal modifier *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 *in an hour* the sort of completed events. As a result, we can successfully distinguish between the following two examples:

(54) a John cleaned the garden in an hour
b *John was working in the garden in an hour.

3.3. *Exstending 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 *a student* is assigned wide scope.

(55) John believes a student to have cheated.
sent[fin]
[s][STUDENT(x), PRESENT(s),
BELIEVE(s, JOHN, [s_1][AFTER(s_1, e), CHEAT(e, x)])]

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

(56) believes a student to have cheated.
sent[fin]/np[nom]:y:pre
[s][STUDENT(x), PRESENT(s),
BELIEVE(s, y, [s_1][AFTER(s_1, e), CHEAT(e, x)])]

It has been customary in monostratal approaches to English syntax to assume that *a student to have cheated* does not form a constituent. Given our treatment of linear order, this leads us to derive the two signs in (57) from (56), where *Z* is used as a temporary place-holder [5].

[5] This analysis departs from other categorial treatments (for example Bach, 1979) and we are not necessarily committed to it.

(57) 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
[t][AFTER(s_1, 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 (57a). This category is encoded by adding a feature specification [*cbse*] to the *sent* symbol that marks verbal heads. Second, infinitives are analysed 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 (57a) has to be cross-identified as the semantics of the active sign in (57a)'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 (57) by the following:

(58) a. believes a student
sent[fin]/np[nom]:y:pre/sent[cbse]/s:[s]S:pre
[s][STUDENT(x), PRESENT(s), BELIEVE(s, y, [s]S])]
b. to have cheated
sent[cbse]/x
[t][AFTER(s_1, e), CHEAT(e, x)

It seems plausible to derive (58.b) 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 [*bse*] for this purpose.

(59) to
sent[cbse]/x/(sent[bse]/x):S:pre
S

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

(60) have cheated
sent[bse]/x
[t][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 may make it possible to treat both the passive and the past participle in the same way, if that is deemed to be desirable. So *have* gets definition (61):

(61) have
sent[bse]/np[nom]:x:pre/(sent[psp]/x):[a]A:pre
[t][AFTER(t, a), [a]T]

For the participle *cheated* we obtain (62).

(62) cheated
sent[psp]/np[nom]:x:pre
[e]CHEAT(e, x)

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

(63) a student
C/(C/np[nom or obj]:x:O):[a]S:O)
[a][STUDENT(x), [a]S]

Believes must therefore be defined as in (64).

(64) believes
sent[fin]/np[nom]:y:pre/(sent[cbse]/x):[s]S:pre/np[obj]: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*.

(65) a. a(n)
(C/(C/np[nom or obj]:singular(b):O):[a]S:O)/noun:[b]R:pre
[a][[b]R, S]
b. student
noun
STUDENT(x)

As a second example, consider the complex nominal

(66) cruel farmer who beats a donkey.

It is fairly clear what this expression should mean, and we propose the sign (67).

(67) cruel farmer who beats a donkey
noun
[x][CRUEL(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:

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

For the noun, we are left with (69).

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

The relative clause is rather similar to the adjective, as appears from (70).

(70) who beats a donkey
noun/noun:[x]A:post
[x][[x]A, [e][DONKEY(y), PRESENT(e), BEAT(e, x, y)]]

This leaves us with the syntax of the relative clause. The analysis proposed here is too simple as it only covers the simplest case. We shall not attempt here to deal with unbounded dependencies necessary for a fuller treatment, though various approaches are compatible with our theoretical framework (cf. Pollard, 1985; Steedman 1985). *Who* combines with the finite verb phrase (71).

(71) beats a donkey
sent[fin]/np[nom]:x:pre
[e][DONKEY(y), PRESENT(e), BEAT(e, x, y)]

It must therefore have definition (72).

(72) who
noun/noun:[x]A:post/(sent[fin]/x):S:pre
[x][[x]A, S]

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 (75) below. A simple example paradigm is illustrated in (73).

(73) eats = [eat: 3sg-pres]
eat = [eat: present]
ate = [eat: past]
eaten = [eat: perfect or passive]
eating = [eat: progressive]

The lexical rules we use are modelled on those in Shieber et al. (1988), and have the general form indicated in (74):

(74) W:Cat:Sem *rightarrow* W1:Cat1:Sem1

That is, they map signs into signs, and we allow them to modify any aspect of the input; this may well be too liberal [6]. Some example rules are illustrated in (75).

(75)

3sg-pres:		
W	→	W + s
sent/x/. . .		sent[fin]/singular(x)/. . .
[a]S		[state(a)][AT (a, NOW), S]
past:		
W	→	W + ed
sent/. . .		sent[fin]/. . .
[a]S		[a][PAST(a), S]
progressive:		
W	→	W + ing
sent/. . .		sent[prp]/. . .
[a]S		[state(s)][WHILE(s, a), [process(a)]S]
perfect:		
W	→	W + en
sent/. . .		sent[psp]/. . .
[a]S		[a]S
infinitive:		
W	→	W
sent/. . .		sent[bse]/. . .
[a]S		[a]S
passive:		
W	→	W + en
sent/np[nom]:y:pre /np[obj]:x:post		sent[pas]/np[nom]:x:pre/ np[by]:y:post
[a]S		[a]S

(76) illustrates how the rules in (75) give rise to verb paradigms like (73) [7].

[6] In particular, these rules allow us to look arbitrarily deep into the category list, whereas our ordinary combinatory rules of syntax do not. We also should note that the lexical rule of passive in clearly inadequate in its present form, since it only applies to transitive verbs.

[7] The treatment of the present tense conflicts in certain respects with our semantic treatment of tense and aspect. Present tense, for example, can only be applied to stative verbs, and is therefore only admissible if we coerce a *habitual* reading for *eat*. If, however, we start from a non-stative reading, the rules for present cannot apply, as the relevant unifications do not succeed. Similarly, if one takes *eat* to refer to completed

(76) a. stemform
eat
sent/np[nom]:a:pre/np[obj]:b:post
[e]EAT(e, a, b)

b. [eat: 3sg-pres]
eats
sent[fin]/np[nom]:x:pre/np[obj]:b:post
[state(e)][PRESENT(e), EAT(e, x, b)]

c. [eat: perfect]
eaten
sent[psp]/np[nom]:a:pre/np[obj]:b:post
[e]EAT(e, a, b)

d. [eat: passive]
eaten
sent[pas]/np[nom]:b:pre/np[by]:a:post
[e]EAT(e, a, b)

3.5. *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 *C/C*. This translates into our framework as the sign

(77) C/C:[a]S

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

(78) noun/noun:[x]A:pre

The normal distinction between intersective, relative and intensional adjectives can be made (cf. Kamp, 1975) [8].

events, the progressive can not be formed. For a discussion of some of these matters, see Moens (1987).

[8] In a language with grammatical gender marking, or a richer system of case inflection, one would require lexical rules to specify the appropriate morphological restriction on the nominal argument of attribute adjectives, as the following Latin example illustrates:

(79) rotundum
noun[acc]/noun[acc]:[male(x)]A
[x][ROUND(x), A]

(80) a. square
noun/noun:[x]A:pre
[x][SQUARE(x), A]

b. big
noun/noun:[x]A:pre
[x][BIG(x, A)), A]

c. fake
noun/noun:[x]A:pre
[x]FAKE([x]A)

As is well known, these same distinctions are typically expressed by meaning postulates in Montague Grammar. For example, the intersective nature of *square* might be expressed by stipulating the analytic validity of (81), where the modifier meaning is related to a property of objects, *square**, which is uniquely determined by (81).

(81) $\exists Z \forall Q \Box \forall x [\mathit{square}(Q)(x) \leftrightarrow (Z(x) \wedge Q(x))]$

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 (81), denotes a function from objects to truth values, 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 more uniform treatment of such modifiers [9].

Predicate adverbs are obtained by instantiating the X in schema (77) to *sent/np*, as illustrated below, (82.a) and (82.b) are intensional, (82.c) is relative.

(82) a. always
C(sent/np)/C(sent/np):[a]S:post
[s]HABIT(s, [a]S)

b. never
C(sent/np)/C(sent/np):[a]S:post
[s][[a] ⇒ ⊥]

c. quickly
C(sent/np)/C(sent/np):[a]S:post
[event(a)][QUICK(a; S), S]

If, on the other hand, we instantiate the X to *sent*, we get the sentential adverbs. (83) illustrates the intensional case.

[9] The one exception is intensionality. In the adjective case, the index of the modified element is preserved, whereas in the case of intensional sentence modifiers it must be reset. This is motivated by the fact that although a *a false coin* denotes a real object that looks like a coin but is not one, the truth of *Allegedly, John walked to Rome on foot* does not require that any event of walking or otherwise took place.

(83) possibly
C(sent)/C(sent):[a]S
POSSIBLE(state(s), [a]S)

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

(84) a. *in*
X/X:[a]S/np[obj]:x:post
[a][IN(a, x), S]
b. *before*
X/X:[a]S/np[obj]:x:post
[a][BEFORE(b, x), [a]S]
c. *when*
sent[fin]/sent[fin]:[b]S:pre/sent[fin]]:[a]T:pre
[b][WHEN(b, a), [a]T, S}
d. *if*
sent[fin]/sent[fin]:S:pre/sent[fin]]:T:pre
[s][T ⇒ S]

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. The case-marking is used for dealing with the subcategorization of certain verbs and nouns for PPs. We illustrate this analysis with to:

(85) to
X/(X/np[to]:x:O):[a]S:O/np[obj]:x:post
[a]S

4. *UCG and Feature Percolation*

In generalized phrase structure grammar (Gazdar et al., 1985), the distribution of morpho-syntactic features is constrained by three major, putatively universal principles: the Head Feature Convention (HFC), the Control Agreement Principle (CAP), and the Foot Feature Principle. As we have said nothing here about unbounded dependency constructions, our remarks will be limited to the claim that whatever is considered to be a foot feature is dealt with by means of the same mechanism as these dependencies. The most obvious candidate for such a treatment is a variant of the threading mechanism. In fact, threading has been extensively used in extensions of the fragment discussed in this paper. Threading also changes the structure of

foot features in such a way, that their inheritance properties follow without any stipulation. However, we will argue that the substantive generalizations expressed by the HFC and the CAP are dealt with in a categorial setting on the level of the formalism with no need for additional stipulations, universal or otherwise, about feature percolation.

4.1. *The Head Feature Convention*

In essence, the HFC dictates that the feature specifications on the mother of a given constituent are identical to the feature specifications on the head of that constituent. This guarantees, for example, that if a verb, such a *require* is subcategorized to take a *S[VFORM BSE]* complement, then the lexical head of that complement will also be specified as [*VFORM BSE*], and thus exihibit base-form morphology; similarly, if a verb imposes [*CASE OBJ*] on an NP complement, then the lexical head of that NP will also bear the same specification for case. Moreover, since the head of a constituent *A*, as fixed by a condition on the phrase structure rule which expands *A*, contains no inherent categorial or bar-level information, the HFC is also responsible for passing category and bar features from mother to head.

By contrast, the fact that a lexical item *A* is the head of some category is automatically encoded in the categorial analysis of *A*. Thus, the category *s[bse]/np[nom]:x:pre* contains the information both that we have a base-form verb, and that the verb is the head of a [*bse*] sentential complement. Nothing further has to be said about head features percolating up to a dominating category. If all category specifications in UCG would be instantiated, the HFC would just fall out of the categorial grammar formalism.

The type-raised XPs and the modifier categories *C/(C/xp)* and *C/C* which contain variable categories complicate this picture. But if we would limit variable categories in UCG, to instances of these two cases, we would succeed in hardwiring the HFC into UCG: the form of the two categories guarantees that modifiers and arguments are non-heads by setting the values of the head features of the result of their application the same as those of the expression to which they apply.

4.2. *The Control Agreement Principle*

In essence, the CAP dictates that the value of a feature AGR on an agreement target has to unify with the categorial specification of

the agreement controller. Thus, for example, the verb *walks* will have the specification [*AGR NP[3s]*], and will thereby be constrained to co-occur with a *NP[3s]* subject (via the mediation of featural information on the dominating VP node). The identification of two constituents as belonging to the controller-target relation depends on a ancillary definition which invokes the semantic types associated with the constituents, where this association in turn is determined by stipulation.

By contrast, in a unification categorial framework there is no need to specify agreement constraints independently of subcategorization information. Thus, as we have seen, the categorization *s[fin]/np-[nom]:singular(x):pre* already tells us that *walks* is a verb which requires a third person singular subject. The fact that we have here used a semantic characterization of third person singular NPs is quite immaterial to the point at issue; we could equally well have used morpho-syntactic features, as in *sent[bse]/np[nom, 3s]:x:pre*.

5. *Conclusion and Further Comparisons*

UCG exhibits a number of similarities with other formalisms in the unification framework. The foremost amongst these similarities 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 allowed in 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 is 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 [10].

Although we have said nothing of interest about phonology here, it seems plausible, in the light of Bach & Wheeler (1981), 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 [11].

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 [12]. These departures from the standard DR formalism do not adversely affect the insights of Kamp's theory, but do offer a sustantial 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, 1987) 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

[10] For more discussion of this general point, see Fenstad et al. (1985).

[11] This would go some way towards vindicating the convinction expressed by Van Riemsdik (1982) that phonologists and syntacticians should take more notice of cach other's work.

[12] Johnson & 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 DRT.

order [13]. On the other hand, the logic is multi-sorted, where the sorts are organized hierarchically so as to form a subsumption lattice. This renders the polymorphism of UCG functions closer in conception to the usual situation 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 *c*.

(i) Suzy likes to walk with every man.

a. suzy
C/(C/np[nom or obj]:SUZY:O):[a]S:O
[a]S

b. every
(C/(C/np[nom or obj]:singular(b):O):[a]S:O)/noun:[b]R:pre
[s][[b]R ⇒ [a]S]

c. man
noun
[x]MAN(x)

d. with
C/C:[a]A:post/np[obj]:x:post
[a][WITH(a, x), A]

dbc. with every man
C/C:[a]A:post
[s][MAN(s) ⇒ [a][WITH(a, x), A]]

e. walk
sent[bse]/x
[e]WALK(e, x)

f. to
sent[cbse]/x/(sent[bse]/x):S:pre
S

ef. to walk: CBSE
sent[cbse]/x
[e]WALK(e, x)

efdbc. to walk with every man
sent[cbse]/x
[s][MAN(x) ⇒ [e][WITH(e, x), WALK(e, y)]]

g. likes
sent[fin]/np[nom]:x:pre/(sent[cbse]/x):S:pre
[s][PRESENT(s), LIKE(s, x, S)]

gefdbc. likes to walk with every man
sent[fin]/np[nom]:x:pre
[t][PRESENT(t), LIKE(t, y, [s][MAN(x) ⇒
[e][WITH(e, x), WALK(e, y)]])]

agefdbc. suzy likes to walk with every man
sent[fin]
[t][PRESENT(t), LIKE(t, SUZY, [s][MAN(x) ⇒
[e][WITH(e, x), WALK(e, SUZY)]])]

This sentence has several other readings, depending on the stage at which the modifier *with every man* is applied.

(ii) Often John visits a cinema

a. often
sent/sent:S:pre
$[s_1]$OFTEN(s_1, S)

b. john
C/(C/np[nom or obj]:JOHN:O):[a]S:O
[a]S

c. visits
sent[fin]/np[nom]:x:pre/np[obj]:y:post
[e][PRESENT(e), VISIT(e, x, y)]

de. a cinema
C/(C/np[nom or obj]:singular(b):O):[b]B:O
[b][CINEMA(x), [b]B]

cde. visits a cinema
sent[fin]/np[nom]:x:pre

[13] We say *largely*, because the question of how to deal with modal contexts still remains unresolved. The semantics of indices for quantifying expressions is naturally thought of as a kind of abstraction and thereby belongs to second order logic. Both issues have still not been settled.

[e][CINEMA(y), PRESENT(e), VISIT(e, x, y)]

bcde. john visits a cinema
sent[fin]
[e][CINEMA(x), PRESENT(e), VISIT(e, JOHN, x)]

abcde. often john visits a cinema
sent[fin]
OFTEN(s, [e][CINEMA(x), PRESENT(e), VISIT(e, JOHN, x)])

REFERENCES

Bach, E. (1979), *Control in Montague Grammar*, «Linguistic Inquiry», 10, pp. 515-531.

Bach, E., Wheeler, D. (1981), *Montague Phonology: A First Approximation*, in Chao, W., Wheeler, D. (eds), *University of Massachusetts Occasional Papers in Linguistics*, volume 7, pp. 27-45.

Calder, J. (1991), *An Interpretation of Paradigmatic Morphology*, Ph.D. Thesis, Edinburgh University.

Davidson, D. (1967), *The Logical Form of Action Sentences*, in Rescher, N. (ed.) *The Logic of Decision and Action*, Pittsburgh, University of Pittsburgh Press.

Fenstad, J.E., Halvorsen, P., Langholm, T., van Benthem, J. (1985), *Equations, Schemata and Situations: a Framework for Linguistic Semantics*, CSLI-report 85-29.

Flickinger, D., Pollard, C., Wasow, T. (1985), *Structure-Sharing in Lexical Representation*, in *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, pp. 262-267.

Flynn, M. (1983), *A Categorial Theory of Structure Building*, in Gazdar, G., Klein, E., Pullum, G. (eds), *Order, Concord and Constituency*, Dordrecht, Foris Publications, pp. 138-174.

Gazdar, G., Klein, E., Pullum, G., Sag, I. (1985), *Generalized Phrase Structure Grammar*, Oxford, Blackwell.

Geach, P.T. (1972), *A Program for Syntax*, in Davidson, D., Harman, G. (eds), *Semantics of Natural Language*, Dordrecht, Reidel.

Goldsmith, J. (1976), *Autosegmental Phonology*, Ph.D. Thesis, MIT.

Heim, I. (1982), *The Semantics of Definite and Indefinite Noun Phrases*, Ph.D. Thesis, University of Massachusetts, Amherst.

Johnson, M., Klein, E. (1986), *Discourse, Anaphora and Parsing*, in *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Bonn University, Bonn.

Kamp, H. (1981), *A Theory of Truth and Semantic Representation*, in Groenendijk, J., Janssen, T.M.V., Stokhof, M. (eds), *Formal Methods in the Study of Language* Amsterdam, Mathematical Centre Tracts, pp. 277-322.

Kamp, J.A.W. (1975), *Two Theories about Adjectives*, in Keenan, E. (ed.), *Formal Semantics of Natural Language: Papers from a Colloquium sponsored by King's College Research Centre*, Cambridge, Cambridge University Press, pp. 123-155.

Karttunen, L. (1986), *Radical Lexicalism*, Paper presented at the Conference on Alternative Conceptions of Phrase Structure, July 1986, New York.

Lambek, J. (1958), *The Mathematics of Sentence Structure*, «American Mathematical Monthly», 65, pp. 154-170.

— (1961), *On the Calculus of Syntactic Types*, in *Structure of Language and its Mathematical Aspects*, Providence, Rhode Island, pp. 166-178.

Moens, M. (1987), *Tense, Aspect and Temporal Reference*, Ph.D. Thesis, University of Edinburgh.

Montague, R. (1973), *The proper Treatment of Quantification in ordinary English*, in Hintikka, J., Moravczik, J.M.E., Suppes, P. (eds), *Approaches to Natural Language*, Dordrecht, Reidel.

Partee, B.H. (1979), *Montague Grammar and the Well-Formedness Constraint*, in Heny, F., Schnelle, H. (eds), *Syntax and Semantics 10: Selections from the Third Groningen Round Table*, New York, Academic Press, pp. 275-313.

— (1987), *Noun Phrase Interpretation and Type Shifting Principles*, in Groenendijk, J., Stokhof, M. (eds), *Studies in Discourse Representation and the Theory of Generalized Quantifiers*, Dordrecht, Foris Publications.

Pollard, C. (1985), *Lectures on HPSG*, Lecture Notes, CSLI, Stanford University.

Proudian, D., Pollard, C. (1985), *Parsing Head-Driven Phrase Structure Grammar*, in *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois.

Reape, M. (1989), *A Logical Treatment of Semi-free Word Order and Bounded Discontinuous Constituency*, in *Proceedings of EACL 1989*, Manchester, pp. 103-110.

Shieber, S. (1986), *An Introduction to Unification-Based Approaches to Grammar*, Chicago, Chicago University Press.

Shieber, S., Uszkoreit, H., Pereira, F., Robinson, J., Tyson, M. (1983), *The Formalism and Implementation of PATR-II*, in Grosz, B. Stickel, M.E. (eds), *Research on the Interactive Acquisition and Use of Knowledge*, SRI International, Menlo Park, pp. 39-79.

Shieber, S., Pereira, F., Karttunen, L., Kay, J. (1986), *A Compilation of Papers on Unification-Based Grammar Formalisms*, CSLI-Report 86-48.

Steedman, M. (1985), *Dependency and Coordination in the Grammar of Dutch and English*, «Language», 61, pp. 523-568.

— (1987), *Combinators and Grammars*, in Oehrle, D., Bach, E., Wheeler, D., *Categorial Grammars and Natural Language Structures*, Dordrecht, Reidel.

Tennent, R.D. (1981), *Principles of Programming Languages*, Englewood Cliffs, NJ, Prentice Hall.

Uszkoreit, H. (1985), *Constraints on Order*, Technical Note 364, SRI International, Menlo Park.

— (1986a), *Constraints on Order*, CSLI-Report 86-46, CSLI, Stanford.

— (1986b), *Categorial Unification Grammars*, in *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Bonn University, Bonn.

Van Benthem, J. (1986), *Categorial Grammar*, Ch. 8 in *Essays in Logical Semantics*, Dordrecht, Reidel.

Van Riemsdijk, H. (1982), *Locality Principles in Syntax and Phonology*, in Korea, T.L.S. (ed.), *Linguistics in the Morning Calm: Selected Papers from SICOL-1981*, Seoul, Hanshin Publishing Company, pp. 693-708.

Wheeler, D. (1981), *Aspects of a Categorial Theory of Phonology*, Ph.D. Thesis, University of Massachusetts at Amherst.

Zeevat, H. (1991), *Aspects of Discourse Semantics and Unification Grammar*, Ph.D. Thesis, Amsterdam University.