```
    standard( Phi, M ).
standard( Phi, N ).

variable( Sort, N, Var ) :-
    prefix( Sort, I ),
    name( N, L ),
    name( Var, [I|L] ).

prefix( e, 88 ). % X
prefix( s, 73 ). % I
prefix( o, 85 ). % U
prefix( p, 86 ). % V
prefix( q, 87 ). % W
```

Appendix 5: List Processing Utilities

```
appendlist( [], [] ).
appendlist( [H|T], L ) :-
    appendlist( T, R ),
    append( H, R, L ).

append( [], L, L ).
append( [H|T], L, [H|R] ) :-
    append( T, L, R ).

member( X, [H|T] ) :-
    X = H;
    member( X, T ).

select( X, [X|L], L ).
select( X, [H|T], [H|L] ) :-
    select( X, T, L ).
```

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)[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 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:

(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_1E_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.

* 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 T2.1, by the current authors and Marc Moens. 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] Recent 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.[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. 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.

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

[2] For example, directional categories, Montague grammar (where a action of rule is added on top of functional application), and combinatory grammar (cf. Van Benthem categorial essays 1986; Geach 1972; Lambek 1958, 1961; Montague 1973; Steedman 1985a).

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 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 instanstiated 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 *visit*:

(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 difficulties. 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).

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 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 different 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:

(12) john
np
JOHN
O

(13) W
np[nom]
x
pre

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)

Notice that as a 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 (15), it can be stripped, yielding (16) as a result.

(16) 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 (17) (rather than (12)):

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

The active sign

(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 (18).

(18) 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 (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)
[e]WALK(e,JOHN)

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), Karttunen (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:

(21) R1': A --> A/B B

R2': A --> B AB

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_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; 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) $[A(x_1, ..., x) => B(y_1, ..., y)]$

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

(24) $\forall x_1...x_n[A(x_1, ..., x_n) -> \exists y_1...y_k[B(y_1, ..., y_k)]]$

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

(25) $A(x_1, ..., x_n)$

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

(26) $\exists x_1...x_n[A(x_1, .. x_n)]$

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. 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 (26).

(28)

	[Index]	*Formula*
a.	[e]	[PARTY(x),[e][TO(e,x),[e][PAST(e),COME(e,JOHN)]]]
b.	[a]	[YESTERDAY(a),[a]A]
c.	[x]	[PARK(y),[x][IN(x,y),MAN(x)]]
d.	[m]	BUTTER(m)
e.	[a]	[PARTY(x),[a][TO(a,x),[a]A]]
f.	[e]	[PAST(e),COME(e)]
g.	[s]	[A => ⊥]

In (28g), '⊥' 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][PARTY(x), [e][TO(e, x), [e][PAST(e), COME(e, JOHN)]]]

is written as (30).

(30) [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 (31).

(31) to the party
John
yesterday

These identities are explicitly expressed in their semantic representations:

(32) [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:

(33) John did not come to the party. Every townsman 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 townsman* and *did not.*

(34) [s][TOWNSMAN(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 abbrevatory labels on a variable where this is suitable. The list (35) associates variable letters with particular sorts.

(35)

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

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

(36) 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 'object'. However, the precise specification of this hierarchy (or lattice) awaits further work.

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)	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 number

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

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*.' (38) illustrates the case of a proper name:

(38) 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 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: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.

(40) 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:

(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[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.

Less common, and one of the interesting aspects of UCG, is the method of imposing restrictions at the level of semantics[3]. 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

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

the variable *x* in (45).

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

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:

(46) a. John cleaned the garden in an hour

b. *John was working in the garden in an hour

The treatment of subject-verb agreement by means of semantics is of course rather controversial, given the distincdon 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/*is sharp
The oats are/*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 meet/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][SCISSORS(plural(a)), COINCIDE(a,singular(b)]

b. [a or b][COMMITTEE(singular(a), COINCIDE(plural(b),a)]

(49a) renders the index of *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) *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 *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 *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 *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 *a student* is assigned wide scope.

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

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

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

[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 (1985), Tasmowski-De Ryck and Verluyten (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
[t][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 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 (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
sent[cbse]/x
[t][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 *[bse]* for this purpose.

(54) 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

(55) 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 makes it possible to treat both the passive and the past participle in the same way. So *have* gets definition (56):

[6] This analysis departs from that often adopted in categorial treatments, as for example Bach (1979), and we are not necessarily committed to it.

(56) 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 (57).

(57) cheated
sent[psp]/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/(C/np[nom or obj]:x:O):[a]S:O)
[a][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[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*:

(60) 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

(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), 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).

(64) 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 (64.)

(65) 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 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; Steedman 1985a,b). *Who* combines with the finite verb phrase (66).

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

It must therefore have definition (67).

(67) 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 (70) below. A simple example paradigm is illustrated in (68).

(68)		
	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 (1983) and have the general form indicated in (69):

(69) W:Cat:Sem --> W':Cat':Sem'

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

this may well be too liberal.[7] Some example rules are illustrated in (70).

(70) 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

(71) illustrates how the rules in (70) give rise to verb paradigms like (68).

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

[eat: 3sg_pres][8]
eats
sent[fin]/np[nom]:x:pre/np[obj]:b:post

---

[7] 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 is clearly inadequate in its present form, since it only applies to transitive verbs.

[state(e)][PRESENT(e), EAT(e, x, b)]

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

[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 A/A. This translates into our framework as the sign

(72) X/X:[a]S

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

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

The normal distinction between intersective, relative and intensional adjectives can be made (cf. Kamp 1975).[9]

(74) 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

[8] The example 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 events, the progressive can not be formed. For a discussion of some of these matters, see Moens and Steedman (1986).

[9] 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; the following Latin example illustrates:

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

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

(75) ∀Q∀x[square(Q)(x) <-> square'(x) & 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 postulae. 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.[10]

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) 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]S -> ⊥]

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 C to *sent*, we get the sentential adverbs. (77) illustrates the intensional case.

(77) 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.

(78) in
X/X:[a]S/np[obj]:x:post

[10] The 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

a false coin

denotes a real object that looks like a coin but is not one, whereas the truth of

allegedly, John walked to Rome on foot

does not require that any walking event took place.

[a][IN(a, x), S]

before
X/X:[a]S/np[obj]:x:post
[a][BEFORE(b, x), [a]S]

when
sent[fin]/sent[fin]:[b]S:pre/sent[fin]]:[a]T:pre
[b][WHEN(b, a), [a]T, S]

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. We illustrate this analysis with *to*:

(79) to
X/(X/np[to]:x:O):[a]S:O/np[obj]: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 organising 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 discouse 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

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

[12] This 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.

[13] Johnson 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 DRT.

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 exacdy 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'.

(A1) 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(x) -> [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
sem[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.

(A2) 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

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

```
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:0
        [b][CINEMA(x), [b]B]

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

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 syntactical rules. In this Appendix, we give a PATR-II version for one example of each of these.

(A3) *Lexical Entry*

(a) UCG:

```
love
sent/np[nom]:x:pre/np[obj]:y:post
LOVE(s, x, y)]
```

(b) PATR-II:

```
phonology = love
syntax :head     = sent
       :feature = bse
catlist :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 *out*, with the same phonology as the sign under the label *in*, and is not conceived as a transformation on the stem.

(A4) *Lexical Rule*

(a) UCG

3sg_pres:

```
W            -->     W+s
sent/x/...           sent[fin]/singular(x)/...
[a]S                 [state(a)][AT(a, 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.

(A5) *Syntax Rule*

(a) UCG

R1: W_1W_2:C:S --> W_1:C/E:S E(W_2:pre)

(b) PATR-II

```
c1 --> c2 c3, { c2:catlist:first= c3
                c1:catlist       = c2:catlist:rest
                c1:syntax        = c2:syntax
                c1:semantics     = c2:semantics}.
```

References

Ades, A. and Steedman, M. J. (1982) On the Order of Words. *Linguistics and Philosophy*, 4, 517-518.

Aho, A. V., Hopcroft, J. E. and Ullman, J. D. (1974) *The Design and Analysis of Computer Algorithms*. Reading, Mass.: Addison-Wesley.

Ajdukiewicz, K. (1935) Die syntaktische Konnexitat. *Studia Philosophica*, 1, 1-27. English translation in: *Polish Logic: 1920-1939* , ed. by Storrs McCall, pp207-231, Oxford University Press, Oxford.

Altmann, G. (1986) Reference and the Resolution of Local Ambiguity: Interaction in Human Sentence Processing. PhD Thesis, University of Edinburgh.

Altmann, G. (1987) Modularity and interaction in sentence processing. In Garfield, J. (ed.) *Modularity in Knowledge Representation and Natural Language Processing*, pp428-444. Cambridge, Mass.: MIT Press.

Aoun, J. and Clark, R. (1984) On non-overt operators. Ms, MIT.

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

Bach, E. (1980) In Defense of Passive. *Linguistics and Philosophy*, 3, 297-341.

Bach, E. and Wheeler, D. (1981) Montague Phonology: A First Approximation. In Chao, W. and Wheeler, D. (eds.) *University of Massachusetts Occasional Papers in Linguistics*, Volume 7, pp27-45. Distributed by Graduate Linguistics Student Association, University of Massachusetts.

Bach, E. (1983) Generalised categorial grammars and the English auxiliary. In Heny, F. and Richards, B. (eds.) *Linguistic categories: auxiliaries and related puzzles*, Volume II, pp101-120. Dordrecht: D. Reidel.

Bar-Hillel, Y. (1953) A quasi-arithmetical notation for syntactic description. *Language*, 29, 47-58.

Bar-Hillel, Y., Gaifman, C. and Shamir, E. (1960) On categorial and phrase structure grammars. In *Language and Information*, No. Also appeared in The Bulletin of the Research Council of Israel, 9F.1.16.

Barendregt, H. P. (1981) *The Lambda Calculus: Its Syntax and Semantics*. North Holland.

Barton, G. E. (1985) On the Complexity of ID/LP Parsing. *Computational Linguistics*, 11, 205-218.

Barton, G. E. (1985) The Computational Difficulty of ID/LP Parsing. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, July, 1985, pp76-81.

Bennett, M. (1974) Some Extensions of a Montague Fragment of English. PhD Thesis.

Berwick, R. C. and Weinberg, A. S. (1983) The role of grammars in models of language use. *Cognition*, 13, 1-62.

Bobrow, R. J. and Webber, B. L. (1980) PSI-KLONE: Parsing and Semantic Interpretation in the BBN Natural language understanding system. In *Canadian Society for Computational Studies of Intelligence*, 1980, pp131-142.

Bobrow, R. J. and Webber, B. L. (1980) Knowledge Representation for Syntactic/Semantic Processing. In *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford, Ca., August 19-21, 1980.

Brame, M. K. (1976) . Amsterdam: North Holland.

Bresnan, J. W. (1972) Theory of complementation in English syntax. Unpublished doctoral dissertation, MIT Press.

Bresnan, J. (1978) A realistic transformational grammar. In Halle, M., Bresnan, J. and Miller, G. A. (eds.) *Linguistic Theory and Psychological Reality*, pp1-59. Cambridge, Mass.: MIT Press.

Bresnan, J. (1982) The passive in lexical theory. Chapter 1 in Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*, pp3-86. Cambridge, Mass.: MIT Press.

Bresnan, J. and Kaplan, R. M. (1982) Introduction: grammars as mental representations of language. In Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*, ppxvii-lii. Cambridge, Mass.: MIT Press.

Bresnan, J. (ed.) (1982) *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press.

Buneman, P., Frankel, R. E. and Nikhil, R. (1982) An implementation technique for database query languages. *ACM Transactions on Database Systems*, 7, 164-186.

Burge, W. H. (1975) *Recursive Programming Techniques*. Reading, Mass.: Addison-Wesley.

Chierchia, G. (1983) Outline of a Semantic Theory of Obligatory Control. In Barlow, M., Flickinger, D. P. and Westcoat, M. (eds.) *West Coast Conference on Formal Linguistics*, Stanford University, Stanford, 1983, pp19-31.

Chomsky, N. (1970) Remarks on nominalization. In Jacobs, R. and Rosenbaum, P. (eds.) *Readings in English transformational grammar*, pp184-221. Waltham, Mass.: Ginn and Co..

Chomsky, N. and Lasnik, H. (1977) Filters and Control. *Linguistic Inquiry*, 8, 425-504.

Chomsky, N. (1981) *Lectures on Government and Binding*. Dordrecht: Foris Publications.

Chomsky, N. (1982) *Some concepts and consequences of the theory of government and binding*. Cambridge, Mass.: MIT Press.

Church, A. (1940) A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5, 56-68.

Colmerauer, A. (1978) Metamorphosis Grammars. In Bolc, L. (ed.) *Natural Language Communication with Computers*, pp133-189. Berlin: Springer-Verlag.

Contreras, H. (1984) A note on parasitic gaps. *Linguistic Inquiry*, 15, 698-700.

Cooper, R. (1977) Variable Binding and Relative Clauses. In *Proc. Formal Semantics Workshop, June 1976*.

Cooper, R. (1983) *Quantification and Syntactic Theory*. Dordrecht: D. Reidel.

Corbett, G. G. (1979) The Agreement Hierarchy. *Journal of Linguistics*, 15, 203-224.

Corbett, G. G. (1981) Syntactic Features. *Journal of Linguistics*, 17, 55-76.

Cormack, A. (1985) VP Anaphora. In Landesman, F. and Veltman, F. (eds.) *Varieties of formal reference: Proceedings of the fourth Amsterdam Colloquium, 1982*, pp81-102. Dordrecht: Foris Publications.

Crain, S. and Steedman, M. J. (1985) On not being led up the garden path: the use of context by the psychological parser. In Dowty, D., Karttunen, L. and Zwicky, A. (eds.) *Natural Language Parsing: Psychological, Computational, and Theoretical perspectives*.

Creary, L. G. and Pollard, C. J. (1985) A computational semantics for natural language. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, 8-12 July, 1985, pp172-179.

Curry, H. B. and Feys, R. (1958) *Combinatory logic*, Volume I. Amsterdam: North Holland.

Curry, H. B. (1961) Some logical aspects of grammatical structure. In *Structure of language and its mathematical aspects:*, Providence, Rhode Island, 1961, pp56-68.

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

Deliyanni, A. and Kowalski, R. (1979) Logic and Semantic Networks. *Communications of the ACM*, 22, 184-192.

Dowty, D. (1978) Governed transformations as lexical rules in a Montague Grammar. *Linguistic Inquiry*, 9, 393-426.

Dowty, D. R., Wall, R. E. and Peters, S. (1981) *Introduction to Montague Semantics*. Dordrecht: D. Reidel.

Dowty, D. (1982) Grammatical relations and Montague Grammar. In Jacobson, P. and Pullum, G. K. (eds.) *The Nature of Syntactic Representation*, pp79-130. Dordrecht: D. Reidel.

Dowty, D. (1985) Type Raising, Functional Composition, and Non-constituent Conjunction. In Oehrle, R., Bach, E. and Wheeler, D. (eds.) *Categorial Grammars and Natural Language Structures*, Dordrecht, 1985. To appear.

Emonds, J. E. (1976) *A Transformational Approach to English Syntax.* New York: Academic Press.

Engdahl, E. (1981) *Multiple gaps in English and Swedish.* Trondheim: Tapir.

Engdahl, E. (1983) Parasitic gaps. *Linguistics and Philosophy*, 6, 5-34.

Evans, R. (1985) ProGram - a development tool for GPSG grammars. *Linguistics*, 23, 213-243.

Fenstad, J. E., Halvorsen, P., Langholm, T. and Benthem, J. (1985) Equations, Schemata and Situations: a Framework for Linguistic Semantics. Technical Report No. CSLI-85-29, August, 1985.

Flickinger, D., Pollard, C. and 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, July, 1985, pp262-267.

Flynn, M. (1983) A categorial theory of structure building. In Gazdar, G., Klein, E. and Pullum, G. K. (eds.) *Order, Concord and Constituency*, pp138-174. Dordrecht: Foris Publications.

Fodor, J. D. (1978) Parsing strategies and constraints on transformations. *Linguistic Inquiry*, 9, 427-473.

Frazier, L. and Fodor, J. D. (1978) The Sausage Machine: a new two-stage parsing model. *Cognition*, 6, 291-325.

Frege, G. (1977) On Sense and Meaning. In Geach, P. and Black, M. (eds.) *Translations from the Philosophical Writings of Gottlob Frege, 3rd Edition*, pp56-78. Oxford: Basil Blackwell.

Friedman, J. and Warren, D. S. (1978) A Parsing Method for Montague Grammars. *Linguistics and Philosophy*, 2, 347-372.

Gallin, D. (1975) Intensional and Higher-Order Modal Logic with Applications to Montague Semantics. North-Holland, Amsterdam: .

Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman.

Gazdar, G. (1981) Unbounded dependencies and coordinate structure. *Linguistic Inquiry*, 12, 155-184.

Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A. (1982) Coordinate structure and unbounded dependencies. In Barlow, M., Flickinger, D. and Sag, I. A. (eds.) *Developments in Generalized Phrase Structure Grammar: Stanford Working Papers in Grammatical Theory*, Volume 2, pp38-68. Bloomington, Indiana: Indiana University Linguistics Club.

Gazdar, G. (1982) Phrase structure grammar. In Jacobson, P. and Pullum, G. K. (eds.) *The Nature of Syntactic Representation*, pp131-186. Dordrecht: D. Reidel.

Gazdar, G., Klein, E., Pullum, G. and Sag, I. (1985) *Generalized Phrase Structure Grammar.* London: Basil Blackwell.

Geach, P. T. (1972) A program for syntax. In Davidson, D. and Harman, G. (eds.) *Semantics of Natural Language.* Dordrecht: D. Reidel.

Goguen, J. A. and Meseguer, J. (1984) Equality, Types, Modules and Generics for Logic Programming. *Journal of Logic Programming*, 1, 179-210.

Goldsmith, J. (1976) Autosegmental Phonology. PhD Thesis, MIT.

Haddock, N. J. (1985) Computing Noun Phrase Reference. Working Paper No. 182, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, July, 1985.

Haddock, N. J. (forthcoming) PhD Thesis, Centre for Cognitive Science and Department of Artificial Intelligence, University of Edinburgh.

Heim, I. (1982) The Semantics of Definite and Indefinite Noun Phrases. PhD Thesis, University of Massachusetts. Distributed by Graduate Linguistics Student Association.

Henkin, L. (1950) Completeness in the Theory of Types. *Journal of Symbolic Logic*, 5, 81-91.

Hudson, R. (1982) Incomplete conjuncts. *Linguistic Inquiry*, 13, 547-550.

Isard, S. D. (1975) Changing the context. In Keenan, E. (ed.) *Formal Semantics of Natural Language.* Cambridge: Cambridge University Press.

Johnson, M. (1985) Grammatical Gender and Pronoun Reference. Unpublished paper, CSLI, Stanford.

Johnson, M. and 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*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, August, 1986.

Jowsey, E. (1984) Argument crossover in categorial grammar. Working Paper, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1984.

Jowsey, H. E. (1985) Constraining Formal Grammar for Computational Applications: A Thesis Proposal. Research Paper, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1985.

Kamp, J. A. W. (1975) Two Theories about Adjectives. In Keenan, E. L. (ed.) *Formal Semantics of Natural Language: Papers from a colloquium sponsored by King's College Research Centre, Cambridge*, pp123-155. Cambridge: Cambridge University Press.

Kamp, H. (1981) A theory of truth and semantic representation. In Groenendijk, J. A. G., Janssen, T. M. V. and Stokhof, M. B. J. (eds.) *Formal Methods in the Study of Language*, Volume 136, pp277-322. Amsterdam: Mathematical Centre Tracts.

Kaplan, D. (1975) How to Russell a Frege-Church. *Journal of Philosophy*, 72, 716-729.

Kaplan, R. M. and Bresnan, J. (1982) Lexical-Functional Grammar: a formal system for grammatical representation. Chapter 4 in Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*, pp173-281. Cambridge, Mass.: MIT Press.

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

Kay, M. (1980) Algorithm Schemata and Data Structures in Syntactic Processing. Technical Report No. CSL-80-12, XEROX Palo Alto Research Centre, Palo Alto, October, 1980.

Kayne, R. S. (1981) ECP extensions. *Linguistic Inquiry*, 12, 93-133.

Kayne, R. S. (1983) Connectedness. *Linguistic Inquiry*, 14, 223-249.

Keenan, E. and Comrie, B. (1977) Noun phrase accessibility and Universal Grammar. *Linguistic Inquiry*, 8, 63-100.

Keenan, E. L. and Faltz, L. M. (1978) *Logical Types for Natural Language*. UCLA Occasional Papers in Linguistics, No. 3.

Kimball, J. (1973) Seven Principles of Surface Structure Parsing in Natural Language. *Cognition*, 2, 15-47.

Klein, E. and Sag, I. A. (1984) Type-driven translation. *Linguistics and Philosophy*. In press.

Kowalski, R. A. (1979) *Logic for Problem Solving*. Amsterdam: North Holland.

Kowalski, R. and Sergot, M. (1985) A Logic-Based Calculus of Events. Unpublished paper, Department of Computing, Imperial College, London.

Kuno, S. (1973) Constraints on internal clauses and sentential subjects. *Linguistic Inquiry*, 4, 363-386.

Lambek, J. (1958) The mathematics of sentence structure. *American Mathematical Monthly*, 65, 154-170.

Lambek, J. (1961) On the calculus of syntactic types. In *Structure of language and its mathematical aspects*, Providence, Rhode Island, 1961, pp166-178.

Link, G. (1983) The Logical Analysis of Plurals and Mass Terms: A Lattice-Theoretical Approach. In Bauerle, R., Schwarze, C. and von Stechow, A. (eds.) *Meaning, Use and Interpretation of Language*, pp302-323. Berlin: de Gruyter.

Lyons, J. (1968) *Introduction to Theoretical Linguistics*. Cambridge: Cambridge University Press.

Mackworth, A. K. (1977) Consistency in Networks of Relations. *Artificial Intelligence*, 8, 99-118.

Maling, J. M. (1972) On Gapping and the order of constituents. *Linguistic Inquiry*, 3, 101-108.

Marcus, M. P. (1980) *A Theory of Syntactic Recognition for Natural Language*. Cambridge, Mass.: MIT Press.

Marr, D. (1977) Artificial Intelligence - A Personal View. *Artificial Intelligence*, 9, 37-48.

Marslen-Wilson, W. and Tyler, L. K. (1980) The Temporal Structure of Spoken Language Understanding. *Cognition*, 8, 1-74.

McCloskey, J. M. (1978) A fragment of a grammar of Modern Irish. PhD Thesis, University of Texas at Austin. In Texas Linguistic Forum 12.

McConnell-Ginet, S. (1982) Adverbs and logical form: a linguistically realistic theory. *Language*, 58, 145-184.

Mellish, C. S. (1981) Coping with Uncertainty: Noun Phrase Interpretation and Early Semantic Analysis. PhD Thesis, University of Edinburgh.

Mellish, C. S. (1982) Incremental Evaluation: An Approach to the Semantic Interpretation of Noun Phrases.

Mellish, C. S. (1983) Incremental Semantic Interpretation. In Sparck-Jones and Wilks (eds.) *Parsing Natural Language*.

Mellish, C. S. (1985) *Computer Interpretation of Natural Language Descriptions*. Chichester: Ellis Horwood.

Moens, M. and Steedman, M. (1986) Temporal Information and Natural Language Processing. ACORD Deliverable 2.5, Centre for Cognitive Science, Edinburgh.

Montague, R. (1970) Universal Grammar. *Theoria*, 36, 373-398. Reprinted in R H Thomason (ed.) (1974), *Formal Philosophy: Selected Papers of Richard Montague*, pp222-246. Yale University Press: New Haven, Conn.

Montague, R. (1973) The proper treatment of quantification in ordinary English. In Hintikka, J., Moravcsik, J. M. E. and Suppes, P. (eds.) *Approaches to Natural Language*. Dordrecht: D. Reidel. Reprinted in R H Thomason (ed.) (1974), *Formal Philosophy: Selected Papers of Richard Montague*, pp247-270. Yale University Press: New Haven, Conn.

Moortgat, M. (1983) A Fregean restriction on metarules. In Sells, P. and Jones, C. (eds.) *Proceedings of the fourteenth annual meeting of the North Eastern Linguistics Society*. Amherst: Graduate Linguistic Student Association.

Moortgat, M. (1985) Mixed Composition and Discontinuous Dependencies. Paper presented at the Conference on Categorial Grammar, Tucson, June 1985. Ms, Leiden.

Pareschi, R. (1987) Combinatory Grammar, Logic Programming, and Natural Language. In Haddock, N. J., Klein, E. and Morrill, G. (eds.) *Edinburgh Working Papers in Cognitive Science*, Volume 1: *Categorial Grammar, Unification Grammar, and Parsing*.

Pareschi, R. (forthcoming) PhD Thesis, University of Edinburgh.

Partee, B. H. (1979) Montague Grammar and the Well-Formedness Constraint. In Heny, F. and Schnelle, H. S. (eds.) *Syntax and Semantics*, Volume 10: *Selections from the Third Groningen Round Table*, pp275-313. New York: Academic Press.

Partee, B. H. (1984) Compositionality. In Landman, F. and Veltman, F. (eds.) *Varieties of Formal Semantics: Proceedings of The Fourth Amsterdam Colloquium, Sept 1982*, Dordrecht, 1984.

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

Pereira, F. C. N. and Warren, D. H. D. (1980) Definite Clause Grammars for Language Analysis – A Survey of the Formalism and a Comparison with Augmented Transition Grammars. *Artificial Intelligence*, 13, 231-278.

Pereira, L., Pereira, F. and Warren, D. (1982) User's Guide to DEC System-10 Prolog. Technical Report, Department of Artificial Intelligence, University of Edinburgh, 1982.

Perlmutter, D. M. (1971) *Deep and surface structure constraints in syntax*. New York: Holt, Rinehart and Winston.

Phillips, J. D. and Thompson, H. S. (1985) GPSGP - a parser for Generalised Phrase Structure Grammars. *Linguistics*, No. 23, 245-261.

Phillips, J. D. A Parsing Tool for the Natural Language Theme. Manual in preparation.

Pollard, C. J. (1985) Categorial Grammar and Phrase Structure Grammar: an excursion on the syntax-semantics frontier. In Oehrle, R., Bach, E. and Wheeler, D. (eds.) *Categorial Grammars and Natural Language Structures*, Dordrecht, 1985. To appear.

Pollard, C. J. (1985) Lectures on HPSG. Unpublished lecture notes, CSLI, Stanford University.

Proudian, D. and Pollard, C. J. (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, 8-12 July, 1985, pp167-171.

Ristad, E. S. (1986) Complexity of Linguistic Models: A Computational Analysis and Reconstruction of Generalized Phrase Structure Grammar. Technical Report, MIT, 1986. M.S. dissertation.

Ritchie, G. D., Black, A. W., Pulman, S. G. and Russell, G. J. (1985) A Dictionary and Morphological Analyser for English Language Processing Systems. Working Paper No. 179, Department of Artificial Intelligence, Edinburgh, Scotland, June, 1985. Revised Edition May 1986.

Ross, J. R. (1967) Constraints on variables in syntax. PhD Thesis, MIT. Indiana University Linguistics Club.

Ross, J. R. (1970) Gapping and the order of constituents. In Bierwisch, M. and Heidolph, M. (eds.) *Progress in linguistics*, pp249-259. The Hague: Mouton.

Roussel, P. L. (1975) Prolog Manuel de Reference et d'Utilisation. Technical Report, Universite' d'Aix-Marseille, 1975.

Sag, I. A. (1983) On parasitic gaps. *Linguistics and Philosophy*, 6, 35-45.

Schönfinkel, M. (1924) Über die Bausteine der Mathematischen Logik. *Mathematische Annalen*, 92, 305-316.

Schachter, P. and Mordechay, S. (1983) A phrase structure account of "nonconstituent" coordination. In Barlow, M., Flickinger, D. and Westcoat, M. (eds.) *Proceedings of the Second West Coast Conference on Formal Linguistics*, Stanford, 1983, pp260-274.

Schmerling, S. (1983) A new theory of English auxiliaries. In Heny, F. and Richards, B. (eds.) *Linguistic categories: auxiliaries and related puzzles*, Volume II, pp1-54. Dordrecht: D. Reidel.

Shaumyan, S. (1977) *Applicational Grammar as a semantic theory of natural language*. Edinburgh: Edinburgh University Press.

Shieber, S., Uszkoreit, H., Pereira, F. C. N., Robinson, J. J. and Tyson, M. (1983) The Formalism and Implementation of PATR-II. In Grosz, B. and Stickel, M. E. (eds.) *Research on Interactive Acquisition and Use of Knowledge*, SRI International, Menlo Park, 1983, pp39-79.

Shieber, S. (1983) Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, 7, 135-154.

Shieber, S. M. (1985) Separating Linguistic Analyses from Linguistic Theories. Paper presented at the UMIST workshop on Linguistic Theory and Computer Applications, 29th August, 1985.

Shieber, S. M. (1986) *An Introduction to Unification-based Approaches to Grammar*. Chicago, Illinois: The University of Chicago Press.

Shieber, S. M., Pereira, F. C., Karttunen, L. and Kay, M. (1986) A Compilation of Papers on Unification-Based Grammar Formalisms Parts I and II. Report No. CSLI-86-48, CSLI, April, 1986.

Steedman, M. (1985) Combinators and Grammars. In Oehrle, R., Bach, E. and Wheeler, D. (eds.) *Categorial Grammars and Natural Language Structures*, Dordrecht, 1985. To appear.

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

Steedman, M. J. (1986) Incremental Interpretation in Dialogue. ACORD Deliverable T2.4.

Steedman, M. (1987) Combinatory Grammars and Parasitic Gaps. In Haddock, N. J., Klein, E. and Morrill, G. (eds.) *Edinburgh Working Papers in Cognitive Science, Volume 1: Categorial Grammar, Unification Grammar, and Parsing*.

Szabolcsi, A. (1983) ECP in categorial grammar. Ms, Nijmegen: Max Planck Institute.

Szabolcsi, A. (1985) Combinators and the projection principle. Paper to the Conference on Categorial Grammar, Tucson, June 1985.

Taraldsen, T. (1979) The theoretical interpretation of a class of marked extractions. In Belletti, A., Brandi, L. and Rizzi, L. (eds.) *Theory of Markedness in Generative Grammar*. Pisa: Scuole Normale Superiore di Pisa.

Tasmowski-De Ryck, L. and Verluyten, S. P. (1981) Pragmatically Controlled Anaphora and Linguistic Form. *Linguistic Inquiry*, 12, 153-154.

Tennent, R. D. (1981) *Principles of Programming Languages*. Englewood Cliffs, N.J.: Prentice-Hall.

Thompson, H. (1982) Handling metarules in a parser for GPSG. Research Paper No. 175, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, 1982.

Thompson, H. S. (1983) MCHART -- a flexible, modular chart parsing framework. In *Proceedings of the 3rd Annual Meeting of the American Association for Artificial Intelligence*, Washington, DC, 1983.

Thompson, H. and Phillips, J. (1984) An Implementation of GPSG within the MChart Chart Parsing Framework. DAI Software Report No. 13, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, 1984.

Turner, D. A. (1979) Another algorithm for bracket abstraction. *Journal of Symbolic Logic*, 44, 267-270.

Turner, D. A. (1979) A new implementation technique for applicative languages. *Software: Practice and Experience*, 9, 31-49.

Tyler, L. and Marslen-Wilson, W. D. (1977) The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behavior*, 16, 683-692.

Uszkoreit, H. (1985) Constraints on Order. Technical Note No. 364, SRI International, Menlo Park, Ca., October, 1985.

Uszkoreit, H. (1986) Constraints on Order. Report No. CSLI-86-46, Center for the Study of Language and Information, January, 1986.

Uszkoreit, H. (1986) Categorial Unification Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, 25-29 August, 1986, pp187-194.

van Benthem, J. (1986) Categorial Grammar. Chapter 8 in *Essays in Logical Semantics*. Dordrecht: D. 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*, pp693-708. Seoul: Hanshin Publishing Company.

Webber, B. L. (1979) *A Formal Approach to Discourse Anaphora*. London: Garland Publishing.

Wheeler, D. (1981) Aspects of a Categorial Theory of Phonology. PhD Thesis, Linguistics, University of Massachusetts at Amherst, Distributed by Graduate Linguistics Student Association, University of Massachusetts.

Wiese, R. (1986) The Role of Phonology in Speech Processing. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, August, 1986, pp608-611.

Williams, E. S. (1978) Across-the-board rule application. *Linguistic Inquiry*, 9, 31-43.

Winograd, T. (1972) *Understanding Natural Language*. New York: Academic Press.