THE SYNTAX OF RELATIVE CLAUSES UNDER GENERALIZED PHRASE STRUCTURE GRAMMAR, COMBINATORY GRAMMAR, AND HEAD-DRIVEN PHRASE STRUCTURE GRAMMAR

G.V. Morrill

Paper submitted in part requirement for the degree of Master of Science

University of Edinburgh

1985

Acknowledgement

I would like to thank my supervisor Ewan Klein for discussions, help, and advice during the preparation of this paper.

Contents

0. Introduction

1. Data to be covered

2. Outline of the formalisms

2.1. Categories, category labels, and category classes

2.2. The formalisms

3. Analyses of relative clauses

3.1. Analyses under Generalized Phrase Structure Grammar

3.2. Analyses under Combinatory Grammar

3.3. Analyses under Head-driven Phrase Structure Grammar

4. Incorporating the antecedent

5. Summary of data, grammars, and implementations

5.1. Summary of data

5.2. Summary of grammars and implementations

6. Discussion

Appendix A: Generalized Phrase Structure Grammar implementation

Appendix B: Combinatory Grammar implementation

Appendix C: Head-driven Phrase Structure Grammar implementation

References

0. Introduction

Clauses such as

(1) a. who works for John
b. for whom Fred works [e][1]
c. whose book Mary loves [e]

are called *relative clauses* (or *adjective clauses* in traditional grammar). A relative clause contains a *relative pronoun* such as who, whom, whose, or which, and modifies a nominal constituent, the relative pronoun's antecedent. Thus the above act as modifiers in the noun phrases

(2) a. the man [who works for John]
b. the man [for whom Fred works]
c. the man [whose book Mary loves]

Relative clauses exhibit the phenomena of anaphora and 'movement'. A characterization of relative clauses must address the question of the relation between relative pronouns and their antecedents. It must also relate a clause such as (1b), which shares its structure with the topicalized sentence

(3) For Alfred Fred works [e]

to the canonical sentence form:

(4) Fred works for Alfred

---

[1] Throughout this paper I use "[e]" to mark extraction sites.

Note that the movement that can occur in relative clauses is unbounded, for example in

(5) The man whom $[_S$ Pete believes $[_S$ Cathy works for [e]]] laughed

the noun phrase position occupied by the relative pronoun has been extracted across two S nodes. The relative clause data that is considered in this paper is described more fully in section 1.

Generalized Phrase Structure Grammar (GPSG), Combinatory Grammar (CG), and Head-driven Phrase Structure Grammar (HPSG) are three frameworks within which, it is claimed, the syntax and semantics of natural language can be captured. In this paper I examine how the syntax of relative clauses can be characterized under the three frameworks. As has been said, relative clauses are of interest because they exhibit the fundamental phenomena of anaphora and movement. GPSG, CG, and HPSG are considered interesting frameworks to compare because of differences in viewpoints on feature percolation and on the distribution of information between the syntactic and lexical components. Thus while GPSG and HPSG assume feature percolation, CG does not, and while GPSG records subcategorization information in syntactic rules, HPSG and CG record it in lexical entries.

There are two ways in which an attempt has been made to make the comparison of the formalisms disciplined. Firstly the frameworks are required to cover exactly the dame data; secondly they are required to cover it in complete detail. This second requirement is ensured by implementing the grammars constructed under the three frameworks.

The syntactic side of each framework is outlined in section 2. Syntactic analyses obtained for some relative clauses are given in section 3, and in section 4 the manner in which relative clauses are joined to their antecedents is considered. In section 5 the GPSG, CG, and HPSG grammars required to cover the data under consideration are summarised, and their implementation is discussed. The implemented grammars are given in the appendices along with sample output. The paper concludes in section 6 with a comparison of the three formalisms and an outline of a new framework which extends what appear to be the best ideas but also differs radically from all three frameworks.

1. Data to be covered

A classification of the relative pronouns that are considered in this paper is given in fig. 1. This classification may not match all idiolects, for example many people use who in accusative positions:

(6) who Bill works for [e]

and whom in nominative positions:

(7) whom I believe [e] died

Such cases would be admitted if who, and whom, like which, could be accusative as well as nominative[2].

	Case	Gender
who	nominative	masculine, feminine, or common
whom	accusative	masculine, feminine, or common
which	nominative or accusative	neuter
whose	possessive	masculine, feminine, or common

Figure 1

[2] However admitting (7) by such a move would also admit say

i) whom likes Lee

The relative clauses

(8) a. who works for John
b. whom Bill works for [e]

are used in this paper as examples of relative clauses containing nominative and accusative relative pronouns and are the first two clauses that are analysed in section 3. The other three relative clauses considered there are

(9) a. whose book Mary loves [e]
b. for whom Fred works [e]
c. a picture of which exists

These exhibit *pied piping*: the relative pronouns are properly embedded in the subject or left-extracted constituents. Such embedding can proceed to any depth; consider

(10) $[_{NPwh}$ a picture of $[_{NPwh}$ a brother of $[_{NPwh}$ a friend of whom]]] hangs on my wall

Fig. 1 also classifies the relative pronoun antecedent genders. The pronoun which must have a neuter antecedent and who, whom, and whose must have non-neuter ones[3]. Thus for example

---

Obtaining (7) may require something more subtle.

[3] It might seem approprate to make a binary animate/inanimate distinction, however because the finer masculine, feminine etc. distinction would be needed to provide a syntactic account of reflexive pronouns, I feel that we should acknowledge here the finer distinction that will ultimately be needed.

(11) a. the man/woman/person whom I like
b. *the man/woman/person which I like[4]
c. the man/woman/person whose house I like

(12) a. *the book whom I like
b. the book which I like
c. *the book whose cover I like

The gender aspect of relative pronouns is considered in section 4 where incorporation of the antecedent is considered. Another issue addressed there is whether the antecedent is a noun phrase, as pronoun antecedents usually are, or a smaller nominal constituent such as a Nom ($\bar{N}$ of X-bar syntax), or a Noun. In *appositive* relatives, i.e. those with comma intonation such as that in

(13) Susan, who lives in Brazil, loves to rumba

it is clear that we have a noun phrase antecedent. In *restrictive* relatives, such as all the other examples given so far, the issue is not so clear; semantically things are simpler if determiners combine after relative clauses, i.e. if relative clauses combine with a constituent smaller than a noun phrase, but in section 4 I argue for noun phrase antecedents.

The above outlines the data to be addressed in this paper. Phenomena not addressed include *free relatives* and relative pronoun omission. Free relative pronouns (*headless relatives*, or *compound relatives* in traditional grammar) such as <u>whoever</u> frequently seem to include their own antecedents. Thus

---

[4] "*" is used to indicate that a sentence is judged ungrammatical. "?" will be used in a similar way to indicate that the grammaticality or ungrammaticality of a sentence is uncertain. Note that I am using 'sentence' in a loose sense: I am not assuming that a sentence is necessarily grammatical.

(14) Whoever wins will receive this prize

corresponds to

(15) He who wins will receive this prize

Relative pronoun omission such as that possible in

(16) the man (whom) I like

can occur when (and only when) a relative clause has a left-extracted constituent consisting only of a relative pronoun:

(17) a. The man who works for John laughed
b. *The man works for John laughed

(18) a. The man [whom] Bill works for [e] laughed
b. The man Bill works for laughed

(19) a. The man [whose book] Mary loves [e] laughed
b. *The man book Mary loves laughed

(20) a. The man [for whom] Fred works [e] laughed
b. *The man for Fred works laughed

The pronoun <u>that</u> appears to be similar to <u>which</u> in taking both nominative and accusative positions (though it may have any gender antecedent):

(21) a. the man that Bill works for [e]
b. the brick that hit me

However there is an interesting difference; <u>that</u> cannot do pied piping:

(22) *The person for that Fred works [e] is a tyrant

This eleventh hour observation must remain unpursued.

Finally, because relative clauses involve unbounded dependencies a complete account would have to cover subject extraction such as that in

(23) The man who I believed [e] had died stood before me

and *parasitic gaps* such as that in

(24) The papers which I filed [e] without reading [p] were crucial

Parasitic gaps cannot appear on their own:

(25) *The papers which I filed them without reading [p] were crucial

but their *hosts* can:

(26) The papers which I filed [e] without reading the titles were crucial

Unfortunately it is not possible to consider subject gaps and parasitic gaps, and their many constraints, here.

2. Outline of the formalisms

The major source of reference for GPSG is Gazdar, Klein, Pullum and Sag (1985), henceforth GKPS. For CG I have worked from a paper by Steedman, 'Combinatory Grammars and Parasitic Gaps' which is still in preparation; I will refer to this as STDMN. My main source of reference for HPSG is Pollard's (1985) 'Lectures on HPSG', henceforth POLL.

2.1. Categories, category labels, and category classes

In all three formalisms, a syntactic *category* is understood to be a set of *feature specifications*: feature/feature-value ordered pairs, such as {<maj, np>, <numb, plur>}. Such a set can also be viewed as a function with domain the set of features for which there are feature/feature-value pairs (see GKPS pp.24-5). Thus if c is {<maj, np>, <numb, plur>}, then c(numb) = plur. Following GKPS (note 1, pp.40-1) the objects such as "{<maj, np>, <numb, plur>}" used to represent categories (e.g. in rules and on trees) will be called *category labels*. Associated with a category is what will be called a *category class*: a set of expressions - the expressions which bear that category.

The features which are used for GPSG, CG and HPSG in sections 2-4 of this paper are listed in their entirety in fig.s 2 and 3[5]. Each feature is

followed by round brackets enclosing its value-set. 'cat' means category, so the value-set of GPSG's AGR is the set of all GPSG categories, and the value-set of HPSG's SUBCAT is the set of all stacks (lists) of HPSG categories. Features are referred to as atom-valued, category-valued, or

Linguistic dimension	GPSG	CG	HPSG
nominal	N({+, -})	-	-
verbial	V({+, -})	-	-
X-bar level	BAR({0, 1, 2})	-	-
person	PER({1, 2, 3})	PER({1, 2, 3})	PER({1, 2, 3})
number	NUM({SG, PL})	NUM({SG, PL})	NUM({SG, PL})
case	CASE({NOM, ACC})	CASE({NOM, ACC})	CASE({NOM, ACC})
verb form	VFORM({BSE, FIN, INF, PAS, PRP, PSP})	VFORM({BSE, FIN, INF, PAS, PRP, PSP})	VFORM({BSE, FIN, INF, PAS, PRP, PSP})
preposition form	PFORM({for, of, ...})	PFORM({for, of, ...})	PFORM({for, of, ...})
subject present	SUBJ({+, -})	-	-
wh-pronoun present	WH(cat)	-	-
wh-pronoun type	WHMOR({R, Q, FR, EX})	-	-
relative pronoun present	-	-	REL(stack of cat)

Figure 2

[5] In section 4 an additional feature, for gender, is added.

Linguistic dimension	GPSG	CG	HPSG
agreement	AGR(cat)	-	-
subcategorization properties	SUBCAT({1, 2, ...} U{that, ..., and, both, ...})	slash(cat pair)	SUBCAT(stack of cat)
possessive	POSS({+, -})	-	-
realized as empty string	NULL({+, -})	-	-
gap present	SLASH(cat)	-	SLASH(stack of cat)
syntactic type	-	NP, S, PP, N, AP, S' VP	MAJ({N, V, P, A, DET})
determiner	DET({+, -}	-	-

Figure 3

stack-valued according to the type of their values. Note that the absence of a feature generally means that the information will be represented some other way, not that there is no means of representing the information.

In order that GPSG, HPSG and CG can be properly compared, it is necessary that they all be required to provide analyses to the same level of detail. This has made it necessary to make a few very minor additions to the theories as presented in the sources of reference. These are remarked upon below.

The grammar presented in GKPS does not include determiners. They could be incorporated by treating them as adjectives, however I have posited a feature DET to mark for determiners. In GKPS PLU({+, -}) is used for number. I have used NUM({SG, PL}) in all three formalisms.

I have not seen any account of CG which actually invokes features for all of person, number, case, preposition form, and verb form but I think my use is fairly uncontroversial. POLL combines person and number into a single feature AGR and does not mention case; again I trust that the embelishments here are in no way a corruption of the spirit of the theory.

There is a slight problem of ambiguity of category labels to which I would now like to draw attention. Suppose it is asserted that "you is {<maj, np>, <pers, sec>}". The feature for number is not mentioned in this category label. Is it being asserted that you belongs to that category class the category for which does not have number in its domain, or are we asserting that you belongs to all category classes for which maj = np and pers = sec, and for which number has any value. The problem can be phrased as follows: is a category label lacking a feature to be interpreted as meaning the category undefined for that feature, or the set of categories unconstrained for that feature?

When a category label appears in a lexical entry or on a tree it is important that its significance be clear. It would be my preference to adopt the 'unconstrained' interpretation, but we would not want {<maj, np>, <pers, sec>} to be interpreted as unconstrained for a verb form feature -

noun phrases should be undefined for such a feature. So we would want to know the domain of a category in order to know whether a feature absent in the category label is unconstained or undefined. Intuitively it is natural to suppose that categories of the same major type are defined over the same domain (I will call this the 'constant domain' hypothesis), so that we would want to define domains for each major syntactic category type, and then if an absent feature is in this domain it is unconstrained, if it is not then it is undefined.

As will have been noticed, this discussion has proceeded in the subjunctive mood. The problem is that the interpretation of category labels that I am proposing is really a hypothesis about what a grammar should be like, in particular it is closely akin to the ideas of unification grammars. The pattern I have proposed can be imposed on CG and HPSG and I do so in this paper - the domains used in sections 2-4 are defined in fig. 4[6] - but it is not entirely in accord with GPSG. For example in GPSG a wh-pronoun, which is a noun phrase, will be specified for WHMOR, which tells what kind of wh-pronoun it is. But it would not make sense for other noun phrases to be defined for this feature because they are not any kind of wh-pronoun. Again, there is a 'feature co-occurence restriction' (see below) in GPSG that tells us that a category cannot be defined for both WH and SLASH, yet on different occasions noun phrases, prepositional phrases and sentences may be defined for either feature. Thus the 'constant domain' hypothesis cannot

---

[6] The feature for gender introduced in section 4 and used in section 5 will be in the N and NP domains of CG, and the N domain of HPSG.

Combinatory Grammar		Head-driven Phrase Structure Grammar	
major syntactic type	domain	major syntactic type	domain
NP	{NUM, PER, CASE}	N	{NUM, PER, CASE, SUBCAT, REL, SLASH}
PP	{PFORM}	P	{PFORM, SUBCAT, REL, SLASH}
S	{VFORM}	V	{VFORM, SUBCAT, REL, SLASH}
VP	{VFORM}		
N	{NUM}		

Figure 4

be maintained in GPSG.

2.2. The formalisms

GPSG, CG and HPSG are outlined in sections 2.2.1, 2.2.2, and 2.2.3 respectively.

2.2.1. Outline of Generalized Phrase Structure Grammar

GKPS uses several abbreviatory devices for category labels. These are necessary because category labels can get very clumsy without them (as we will see!), but the abbreviations can confuse matters if one is unfamiliar with GPSG to start with. I have decided to write out categories in full

whilst outlining GPSG in this section. The abbreviations are introduced at the end of this section and are used thereonwards.

A GPSG grammar has five components: *feature co-occurence restrictions, immediate dominance rules, linear precedence statements, metarules,* and *feature specification defaults.* Feature co-occurence restrictions (FCRs) constrain the set of acceptable categories. For example the FCR

(27) {<SUBJ, +>} ⊃ {<V, +>, <N, ->, <BAR, 2>}

states that any category including the feature specification <SUBJ, +> must also have the feature specifications <V, +>, <N, ->, and <BAR, 2>. A category is *legal* if and only if every FCR is true of that category. Thus the existence of the FCR (27) in a GPSG grammar means that the categories {<V, +>, <N, ->, <SUBJ, +>} and {<V, ->, <N, ->, <BAR, 2>, <SUBJ, +>} are not legal: the former does not include <BAR, 2>, and the latter has the wrong value for V.

An important notion in GPSG is that of *extension.* This is defined as follows

(28) A category A is an extension of a category B if and only if
i) the atom-valued feature specifications in B are all in A, and
ii) for any category-valued feature f, the value of f in A is an extension of the value of f in B.

So for example {<V, ->, <N, +>, <BAR, 2>} and {<BAR, 2>} are extensions of {<BAR, 2>}, and {<VFORM, FIN>, <AGR, {<V, ->, <N, +>, <BAR, 2>, <CASE, NOM>}>} is an extension of {<VFORM, FIN>, <AGR, {<V, ->, <N, +>, <BAR, 2>}>}. However {<V, ->, <N, +>} and {<BAR, 1>} are not extensions of {<BAR,

2>}, and {<VFORM, FIN>, <AGR, {<V, ->, <N, +>, <BAR, 1>}>} is not an extension of {<VFORM, FIN>, <AGR, {<V, ->, <N, +>, <BAR, 2>}>}.

We use the notion of extension in providing an interpretation for immediate dominance (ID) rules. ID rules are similar to rewrite rules. They present 'skeletons' on which local trees (trees of depth one) must be based: a local tree must be a *projection* of some ID rule, i.e. the categories in the local tree must be legal extensions of the corresponding categories in the ID rule. Consider for example the subject-predicate ID rule

(29) {<V, +>, <N, ->, <BAR, 2>, <SUBJ, +>} -> {<BAR, 2>}, H{<SUBJ, ->}[7]

The local trees in fig.s 5a and 5b are both projections of this rule. These are the same but for the order of the daughters. ID rules do not impose any order on the daughters. The local trees in fig.s 5c and 5d are not projections: the former does not have <SUBJ, +> on the mother and the latter has <BAR, 1> not <BAR, 2> on the left-hand daughter, so they are not extensions of the corresponding categories in the ID rule. The local tree in fig. 5e is not a projection of the ID rule because although the categories are extensions, the category on the left-hand daughter is not legal if we have FCR (27).

---

[7] The "H" tells us that that category is the head but does not provide any featural information. "X" is also sometimes used as a place-marker.

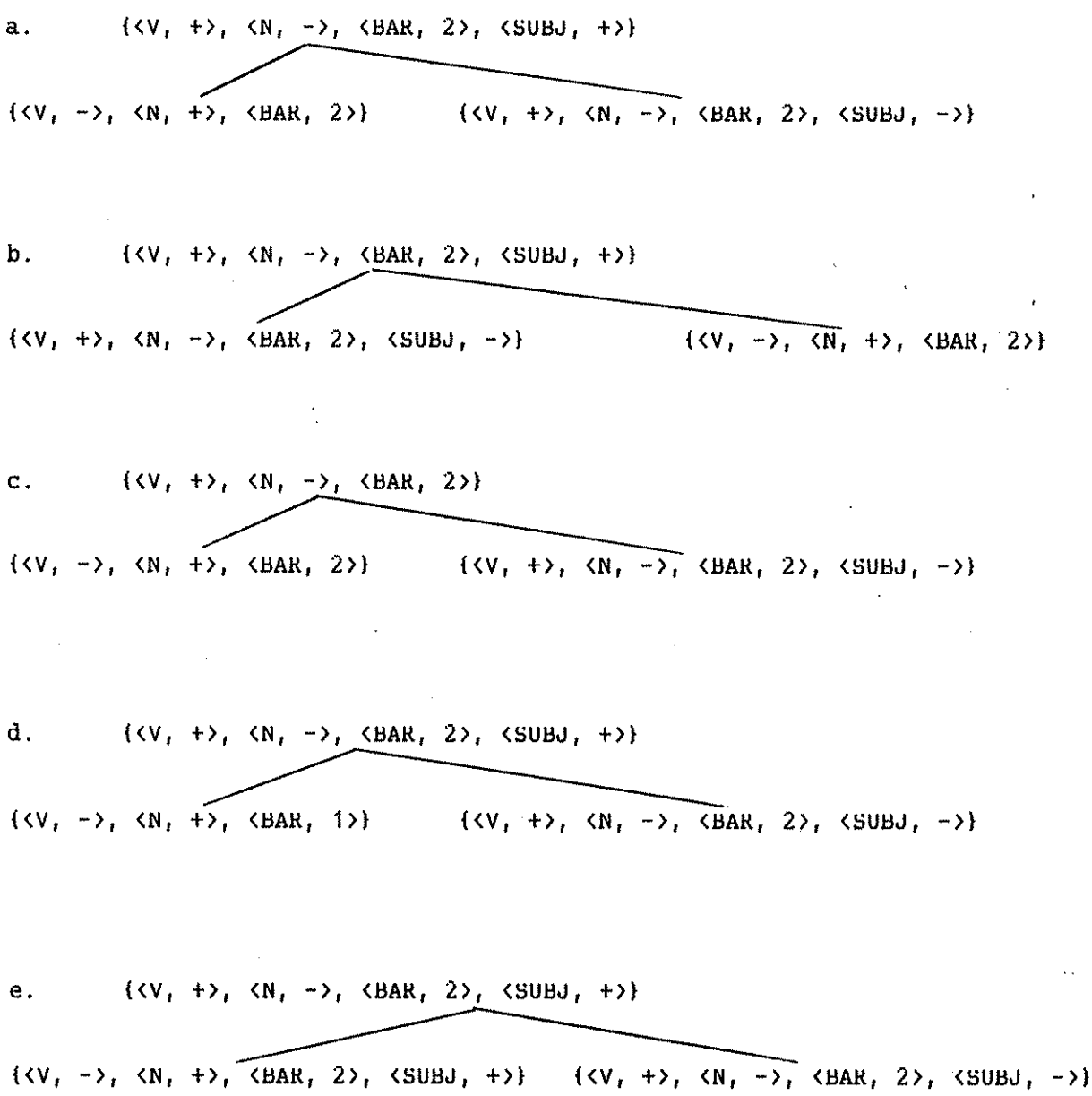

Figure 5

Linear ordering of daughters is fixed by linear precedence (LP) statements such as

(30) {<N, +>} < {<V, ->, <N, ->, <BAR, 2>} < {<V, +>, <N, ->, <BAR, 2>}

"X < Y" means that any category which is an extension of X must come to the left of any sister which is an extension of Y. The precedence operator is transitive. A local tree is *LP-acceptable* if and only if it does not contravene any LP statement. It will be seen that the local tree in fig. 5b is not LP-acceptable since the right-hand daughter is an extension of {<N, +>} and the left-hand daughter is an extension of {<V, +>, <N, ->, <BAR, 2>}, but the former is not left of the latter. All the other trees in fig. 5 are LP-acceptable (with respect to the set of LP rules consisting soley of (30)).

Metarules are rules which have the effect of reducing the set of ID rules that need to be listed by stating that for each ID rule of a certain form, there is another ID rule of a related form. Consider the so-called 'Slash Termination Metarule 1" (STM1):

(31) X{} -> *W*, {<BAR, 2>}

$$\downarrow$$

X{} -> *W*, {<BAR, 2>, <NULL, +>}

W is a variable over sets of categories. The rule tells us that for every ID rule with a category which is an extension of {<BAR, 2>} on the right hand side, there is another ID rule which is the same except that this category also carries the specification <NULL, +>. In fact the application of the rule is slightly more limited than this. Metarules can only apply to ID rules in which the head daughter is specified for SUBCAT (this means it is 'lexical' - it dominates a lexical item). So metarules cannot apply to

(29) for example. Also, metarules cannot apply to their own output. This means that we will not get metarules applying indefinately producing an infinite number of ID rules.

The metarule (31) has the effect of allowing a gap to appear in the position of <BAR, 2> categories which have lexical sisters. This is what happens in the topicalized sentence

(32) For Alfred Bill works [e]

and (31) will apply to a rule expanding works:

(33) {<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->} -> H{<SUBCAT, 999>},
{<V, ->, <N, ->, <BAR, 2>, <PFORM, for>}

to give an ID rule for the topicalized case:

(34) {<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->} -> H{<SUBCAT, 999>},
{<V, ->, <N, ->, <BAR, 2>, <PFORM, for>, <NULL, +>}

A feature specification default (FSD) such as

(35) -{<NULL, v>}

means something like 'unless otherwise indicated, a category on a tree is not specified for NULL'. The 'unless otherwise indicated' part turns out to be very complicated and FSDs are, by GKPS's own admission, an aspect of the theory for which insufficient interpretation has been provided (see for example GKPS note 2, p.14). I do not consider it worthwhile to discuss FSDs further here.

One of the tenets of GPSG is that the correct way to go about describing natural language syntax is to identify feature percolation conventions which define relations between the features on sisters and mothers in local trees. Two classes of feature are distinguished on the basis of their percolation behaviour: **HEAD** and **FOOT**. Limiting ourselves to features used in this paper, these classes are as follows:

(36) a. **HEAD** = {N, V, BAR, PER, NUM, VFORM, PFORM, SUBJ, AGR, SUBCAT, SLASH,}
b. **FOOT** = {WH, SLASH}

Note that SLASH is a member of both classes, and CASE, WHMOR, POSS, NULL, and my DET are members of neither. The feature percolation conventions in GPSG are the Foot Feature Principle, the Control Agreement Principle, and the Head Feature Convention. To give a definition of the Foot Feature Principle (FFP) we need to give a couple of other definitions. Firstly we need to make a distinction between *inherited* and *instantiated* features on a local tree. If a feature specification in a local tree also appears in the corresponding category in the tree's ID rule, that specification is said to be inherited; otherwise a feature specification is said to be instantiated. Thus in the left-hand daughter of fig. 5a (which is projected from (29)), <V, -> and <N, +> are instantiated, and <BAR, 2> is inherited. Secondly, we need to define *unification*:

(37) The unification of a set of categories K (U(K)) is the smallest category which is an extension of every member of K.

(the unification is undefined if there is no such category). Thus the unification of {} and {<SLASH, {<V, ->, <N, ->, <BAR, 2>, <PFORM, for>}} is {<SLASH, {<V, ->, <N, ->, <BAR, 2>, <PFORM, for>}}, and the unification of <N, -> and <N, +> is undefined. We can now define the FFP as follows

(38) The **FOOT** feature specifications that are instantiated on the mother category in a tree must be identical to the unification of the instantiated **FOOT** feature specifications on the daughters.

To see an example of the use of this consider using the subject-predicate ID rule (29) for a relative clause:

(39) who works for John

The subject who is the non-head daughter. Because it is a relative pronoun it will be specified for WH. Now WH is a **FOOT** feature, so the FFP requires that the WH value that is instantiated on this daughter is also instantiated on the mother (the unification of the **FOOT** feature specifications on the daughters will be just this WH specification). Thus the (partial) local tree will be that in fig. 6.

The Control Agreement Principle (CAP) handles control, ensures subject-verb agreement and ensures that extracted items 'fit' the position from which they have been extracted. The principle makes reference to the

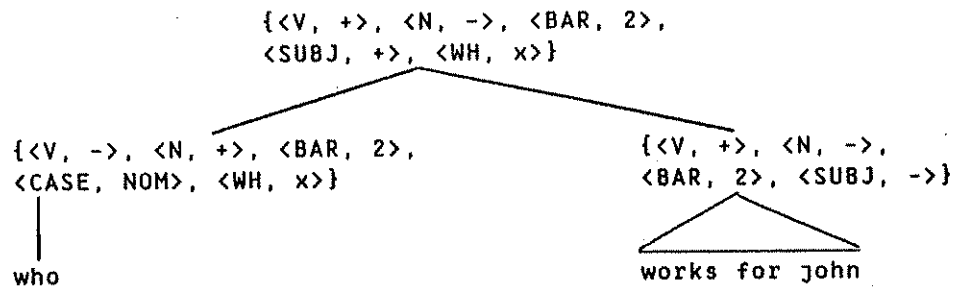

Figure 6

semantic types associated with categories. TYP(x) is the type associated with a category x. I list some of the values of TYP in fig. 7. Additional information about types is given by the fact that a lexical head is always associated with a type that is a function over the type(s) associated with its sister(s). So for example, since the ID rule (33) gives works a sister associated with type np and a mother associated with type <np, s>, works must be <np, <np, s>>[8]. Note that the type is not completely fixed if a lexical head has more than one sister: one or other sister may correspond to

Category(x)	Type(TYP(x))
{<V, +>, <N, ->, <BAR, 2>, <SUBJ, +>}	s
{<V, ->, <N, +>, <BAR, 2>}	np
{<V, ->, <N, +>, <BAR, 1>}	n1
{<DET, +>}	<n1, np>
{<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->}	<np, s>
{<V, ->, <N, ->, <BAR, 1>, <PFORM, v>}	np
{<V, ->, <N, ->, <BAR, 2>, <PFORM, v>}	np
U({y, {<WH, z>}})	<TYP(z), TYP(y)>
U({y, {<SLASH, z>}})	<TYP(z), TYP(y)>

Figure 7

[8] In fact this reasoning is slightly faulty because we are interested in the types corresponding to categories in actual local trees; these may differ from the types associated with the corresponding categories in the projecting ID rules, e.g. they would if SLASH or WH were instantiated. However instantiated SLASHs and WHs will not be present in the "$\overline{X}$"-

the first argument. It is by these alternative possibilities that GPSG captures the different control properties of persuade and promise. It appears that the features NUM, CASE, VFORM, WHMOR, AGR, and NULL are irrelevant when associating types with categories. It seems to me that <POSS, -> should be irrelevant but that {<V, ->, <N, +>, <BAR, 2>, <POSS, +>} should be <n1, np> - see section 3.1.3.

The types that are relevant to the control relations in a local tree are not those associated with the actual categories in the tree, but rather those associated with the $_X$-*specifications* (chi-specifications) of the categories. The $_X$-specification of a category on a local tree is the **HEAD** feature specifications less those that are also **FOOT** specifications (SLASH), plus inherited **FOOT** feature specifications (possibly including SLASH again). So for example the left-hand daughter in fig. 6 (which is projected from (29)) has $_X$-specification {<V, ->, <N, +>, <BAR, 2>} since CASE is not a **HEAD** feature and WH is instantiated. Note that $_X$-specification is relative to an ID rule since reference is made to inheritance and instantiation. This means that $_X$-specification may differ depending on whether a category is viewed as a daughter in one local tree or a mother in another. Where $_X$(x) is the $_X$-specification of x we can now define *control* in a local tree:

(40) A daughter category i controls a sister j if and only if
i) TYP($_X$(j)) = <TYP($_X$(i)), TYP($_X$(m))> where m is the mother category, or
ii) TYP($_X$(j)) = <np, s>, and TYP(h) = <<np, s>, <TYP($_X$(i)), <np, s>>>.

(The second clause is not relevant to this paper).

---

specifications" (see below) that are relevant to control.

We also need to define the notion of *CONTROL* feature. A category i, i(BAR) ≠ 0, has CONTROL feature SLASH if a SLASH-specification is inherited, otherwise it has CONTROL feature AGR (provided AGR is present of course; failing this there is no CONTROL feature). We can now actually define the CAP. Because control and the CONTROL feature are defined relative to an ID rule, the definition is again relative to an ID rule.

(41) The CAP requires that where i,j and k are categories in a local tree
i) if i controls j, then $j(f_j) = X(i)$ plus i's f_j-specification, where f_j is j's CONTROL feature, and
ii) if TYP(k) = ⟨np, X⟩ and k has no controller, then $k(f_k) = m(f_m)$, where m is the mother category and f_k and f_m are the CONTROL features of k and m respectively.

(The second clause is not relevant to this paper).

This outline of GPSG is completed with a consideration of the Head Feature Convention (HFC). The full version of the HFC is complicated by the need to encompass multiple headedness (in coordination). A simpler 'single-head' version suffices here. First we need to define the notion of *free feature specification set* (ffss). A feature specification ⟨f, v⟩ is a member of the ffss of a category x in an ID rule if and only if there exists at least one projection from that ID rule in which ⟨f, v⟩ appears on the category in the tree corresponding to x, and in which the FFP and CAP are met. A local tree meets the HFC if and only if

(42) i) the set of **HEAD** feature specifications on the mother is an extension of the set of **HEAD** feature specifications on the head daughter, less those that are not in the mother's ffss, and
ii) the set of **HEAD** feature specifications on the head daughter is an extension of the set of **HEAD** feature specifications on the mother, less those that are not in the daughter's ffss.

We illustrate the use of the CAP and HFC, as well as all the other devices introduced here, by the complete analysis for

(43) Us Andrew envies [e]

in fig. 8[9]. The ID rules for local trees (a), (b) and (c) are

(44) a. {<V, +>, <N, ->, <BAR, 2>, <SUBJ, +>} -> {<BAR, 2>}, H{<SLASH, {<BAR, 2>}>}
b. {<V, +>, <N, ->, <BAR, 2>, <SUBJ, +>} -> {<BAR, 2>}, H{<SUBJ, ->}
c. {<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->} -> H{<SUBCAT, 2>},
{<V, ->, <N, +>, <BAR, 2>, <NULL, +>}

(44a) is the rule allowing us to expand a sentence as a filler together with a sentence containing a gap; (44b) is the subject-predicate rule we saw earlier. The rule (44c) is obtained from the rule expanding transitive verbs:

(45) {<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->} -> H{<SUBCAT, 2>},
{<V, ->, <N, +>, <BAR, 2>}

by the metarule STM1 in (31). It will be seen that all the categories in the tree are extensions of the corresponding categories in the ID rules.

Consider the linear ordering of the daughters in (a). The left-hand daughter is an extension of {<N, +>}; the right-hand daughter is an extension of {<V, +>, <N, ->, <BAR, 2>}. So this ordering is imposed by the LP rule (30) repeated below

(46) {<N, +>} < {<V, ->, <N, ->, <BAR, 2>} < {<V, +>, <N, ->, <BAR, 2>}

---

[9] I have given the analysis as it is supposed to be; as I discuss below though I am not sure that this is what the feature percolation conventions actually give us.

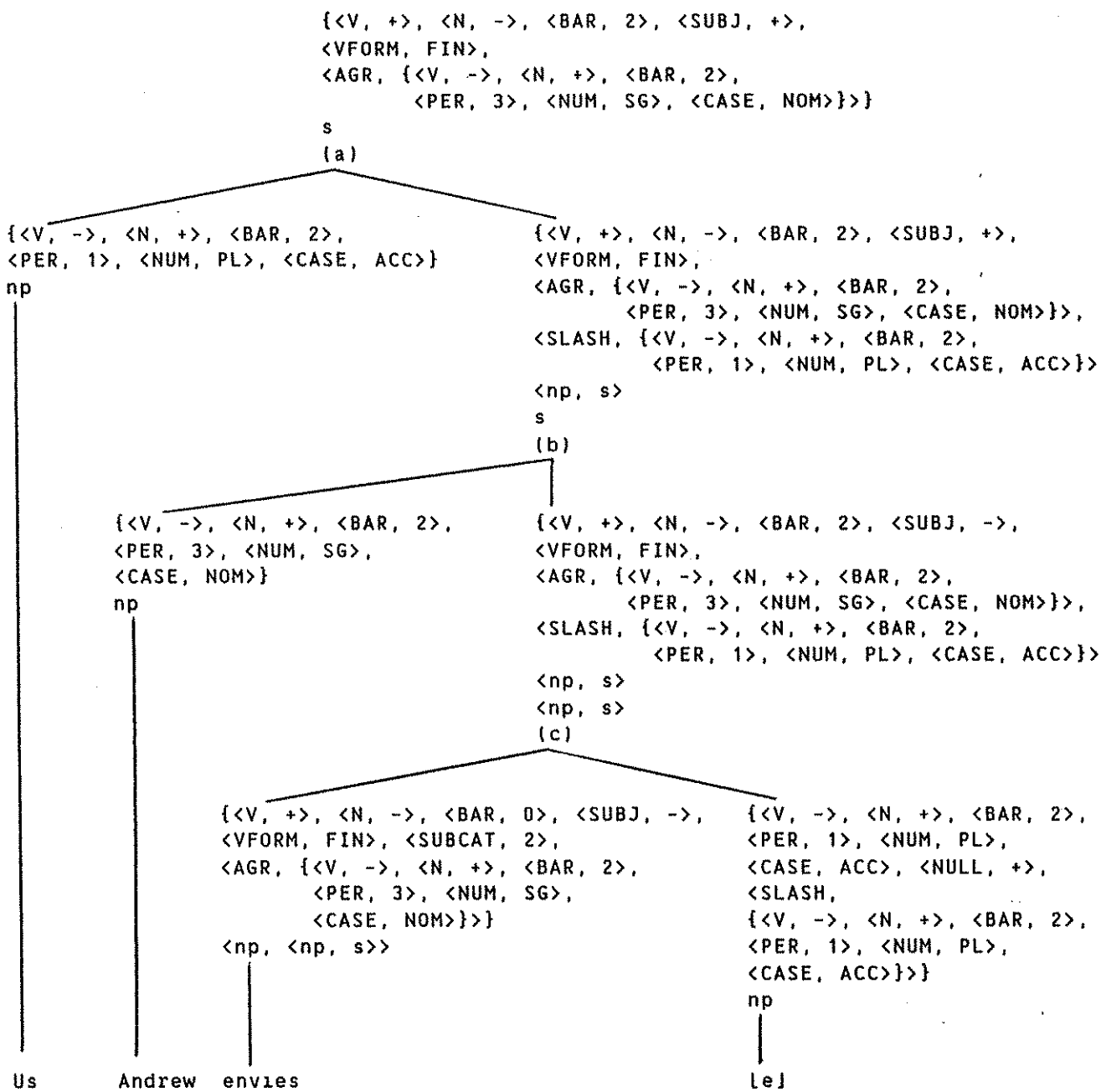

Figure 8

The daughters in (b) are extensions of these same categories, so the

ordering is imposed again. However in (c) the left-hand daughter is not an extension of any category in (46). Of course ordering doesn't really matter in this particular case because the right hand daughter happens to be realized as the empty string, but in general an additional LP rule is needed to ensure that transitive verbs precede their objects:

(47) {<SUBCAT, v>} < -{<SUBCAT, v>}

Does (a) meet the FFP? Since SLASH is inherited not instantiated on the right-hand daughter, the unification of the **FOOT** features instantiated on the daughters is the empty set. There are no **FOOT** features on the mother so the FFP is met. In (b) and (c) the mothers and right hand daughters have the same SLASH values instantiated and no **FOOT** features appear on the left hand daughters, so the FFP is met here as well.

The types associated with the $\bar{X}$-specifications of the categories in the tree are noted below each node. When a node is both a daughter and a mother its 'daughter type' is written above its 'mother type'. SLASH is inherited on the right hand daughter of (a) and so is included in the $\bar{X}$-specification, but it is instantiated on the mothers and right hand daughters of (b) and (c) and is not in the $\bar{X}$-specification. The type for the category dominating envies is that from the type for its sister to that for its mother, since it is a lexical head.

From the definition of control in (40) it will be seen that the left hand daughter controls the right hand daughter in (a) and (b), and the right

hand daughter controls the left hand daughter in (c). In (a) the controlled daughter has CONTROL feature SLASH. The CAP requires that this SLASH's value equals the $\bar{X}$-specification of the left hand daughter. This is almost the case, but it will be noted that because CASE is not a **HEAD** feature, it is not in the $\bar{X}$-specification. So here we seem to have stumbled across a problem with the CAP and CASE in GPSG. The CAP fails to make the subject nominative in case. This problem would be rectified if we stipulated that CASE is a **HEAD** feature

In (c) the controllee, the left hand daughter, does not have a CONTROL feature because although it is specified for AGR, to have a CONTROL feature a category must be non-<BAR, 0>. As a result, the CAP has no effect.

Finally we consider the HFC. In (a) the mother shares all the head daughter's **HEAD** feature specifications except that for SLASH. This violates clause (1) of the HFC unless the SLASH specification is not in the mother's ffss. Unfortunately I think it is; thus where c is any category, I think that the local tree in fig. 9 is a projection of (44a) which meets both the FFP and the CAP.

In (b) the mother and head daughter share their **HEAD** feature specifications except those for SUBJ. This however does not violate the HFC: <SUBJ, -> is not in the mother's ffss because the mother inherits <SUBJ, +>, and <SUBJ, +> is not in the head daughter's ffss because the head daughter inherits <SUBJ, ->[10].

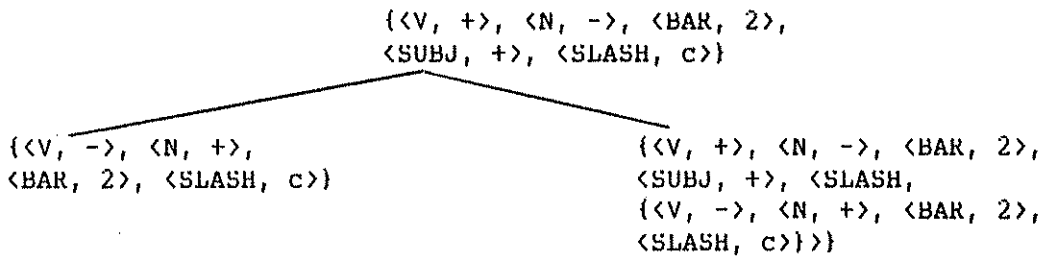

Figure 9

In (c) the mother is an extension of the head daughter except for BAR and SUBCAT. This is alright: the mother inherits <BAR, 2> so <BAR, 0> is not in its ffss, and the FCR

(48) {<BAR, 2>} ⊐ -{<SUBCAT, v>}

means that SUBCAT is not in its ffss either. The head daughter is an extension of the mother but for BAR and SLASH. <BAR, 2> is not in the head daughter's ffss because the head daughter inherits a SUBCAT specification. SLASH is not in its ffss because there is an FCR

(49) {<SUBCAT, v>} ⊐ -{<SLASH, u>}

So (c) meets the HFC.

This section is concluded by introducing some category label abbreviations used in GKPS and which will be used in section 3. Feature-

---

[10] I have not troubled to mention that a category can have only one value for any feature.

feature/value pairs are listed in square brackets, the feature names being omitted if they are fixed by the values. <F, +> is represented +F. The symbols on the left hand side in fig. 10 may be written before the square brackets; they abbreviate the sets of specifications on their right hand sides[11]. "+R" inside the square brackets abbreviates {<WH, {<N, +>, <V, ->, <BAR, 2>, <WHMOR, R>}>}, and /x after the square brackets abbreviates {<SLASH, x>}. The use of these abbreviations is illustrated in fig.

```
S       {<V, +>, <N, ->, <BAR, 2>, <SUBJ, +>}

XP      {<BAR, 2>}
X2      {<BAR, 2>}
X1      {<BAR, 1>}

V       {<V, +>, <N, ->}
VP      {<V, +>, <N, ->, <BAR, 2>, <SUBJ, ->}
V2      {<V, +>, <N, ->, <BAR, 2>}

N       {<V, ->, <N, +>}
NP      {<V, ->, <N, +>, <BAR, 2>}
N2      {<V, ->, <N, +>, <BAR, 2>}
N1      {<V, ->, <N, +>, <BAR, 1>}

P       {<V, ->, <N, ->}
PP      {<V, ->, <N, ->, <BAR, 2>}
P2      {<V, ->, <N, ->, <BAR, 2>}
P1      {<V, ->, <N, ->, <BAR, 1>}
```

Figure 10

---

[11] In GKPS the digits are superscripts.

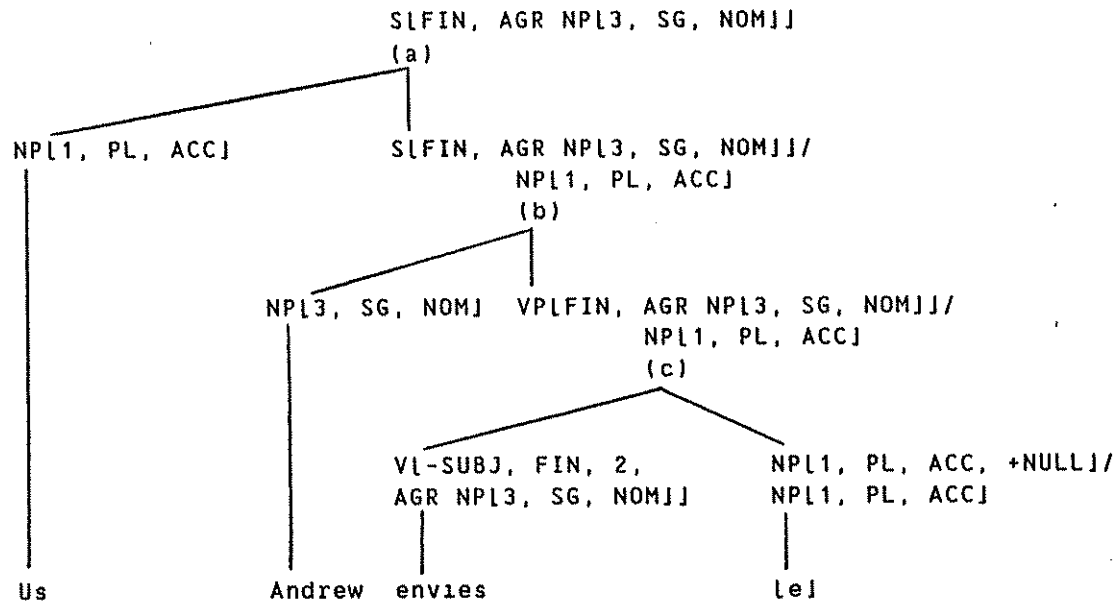

Figure 11

11 which is the abbreviated form of the analysis in fig. 8.

2.2.2. Outline of Combinatory Grammar

In CG, 'atomic' categories - categories without slashes (see below) - will be represented by a symbol for their major type followed by square brackets enclosing the feature values for which they are defined. For example "NP[3, SG]" represents a third person singular noun phrase which is unconstrained for CASE.

Combinatory Grammars are generalizations of Ajdukiewicz' (1935) Categorial Grammar. In each case grammars consist of two components: a

categorial lexicon which associates categories with words, and combinatory rules.

In STDMN, noun phrases such as John and a book have category NP and nouns like book have category N. A determiner like a has category NP/N. Such a category is interpreted as a function over nouns to the right which yields a noun phrase. STDMN assigns verb phrases VP and so adverbs such as quickly are VP\VP: functions which apply to verb phrases to their left to give new verb phrases[12]. A verb like give has category (VP/NP)/NP.

This syntactic function-argument structure is intended to directly reflect the semantic one. We are limiting attention to syntax here but I will give (without comment) the semantic rules which parallel the syntactic rules.

The simplest combinatory rules are the rules of functional application. These are as follows (the semantic interpretations appear right of the colons)

(50) a. Forward Application (">apply")
X/Y:F Y:y => X:F(y)
b. Backward Application ("<apply")
Y:y X\Y:F => X:F(y)

The use of these rules is illustrated in fig. 12. Note how (a) is admissable because NP[ACC] and NP[3, SG] can be unified. Note also that the daughters are the daughter categories as they are before this unification

[12] VP is really just an alternative notation for S\NP; I use the latter in section 5.

has taken place; afterwards they are both NP[3, SG, ACC]. In all CG trees, daughter categories will be represented as they are before unification.

The rules of functional application are the only syntactic rules in Categorial Grammar which, incidently, is weakly equivalent to Context-Free Grammar. In CG further operations are added such as

(51) a. Forward Composition (">compose")
X/Y:F Y/Z:G => X/Z:Lx[F(G(x))]
b. Backward composition ("<compose")
Y/Z:G X\Y:F => X/Z:Lx[F(G(x))]

STOMN posits that similar rules with other combinations of slash directions are not present in English, but are available in universal grammar. In fig. 13a it is shown how (51a) can be used in leftward extraction to build a bridge from the extraction site towards the topicalized item. Fig. 13b shows how (51b) is used in a similar way in rightward extraction to link

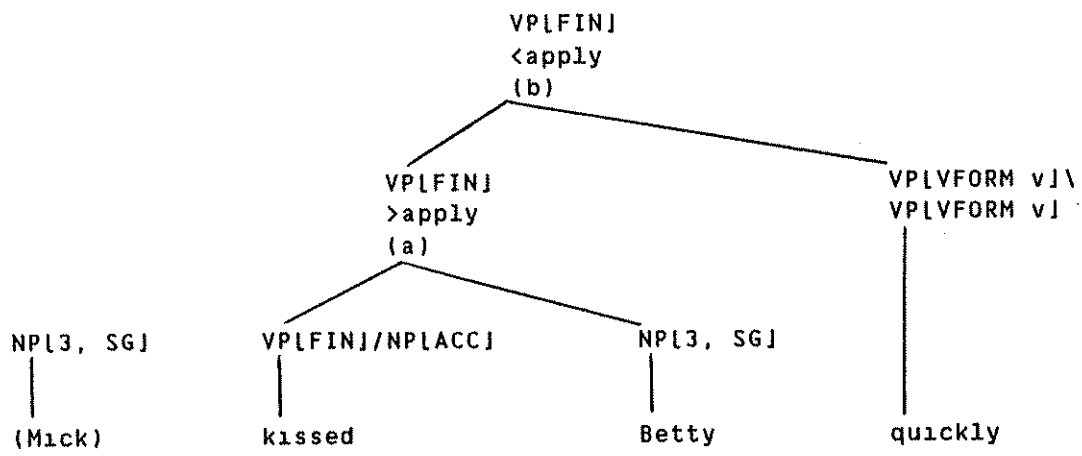

Figure 12

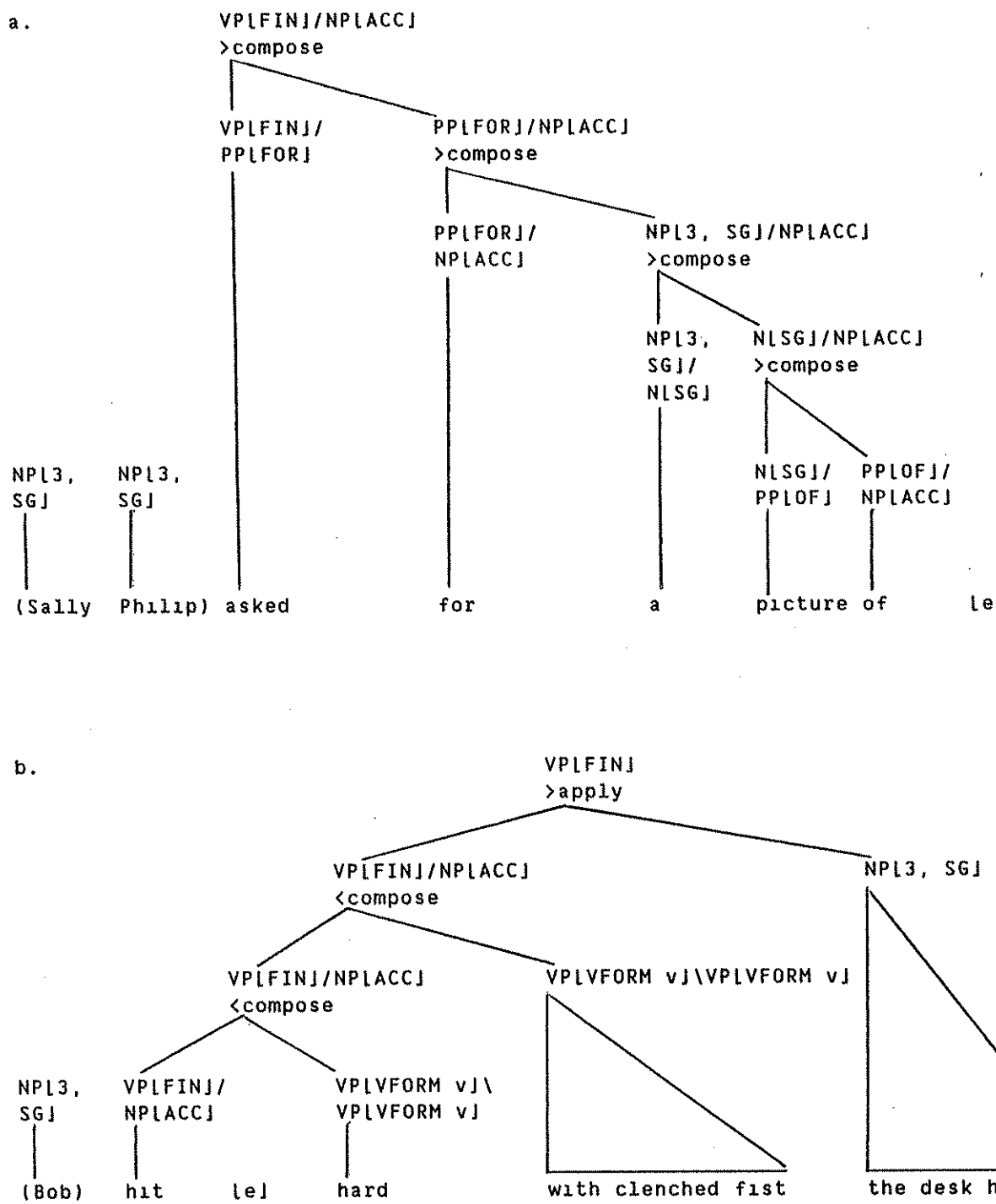
a.
VP[FIN]/NP[ACC]
>compose
VP[FIN]/
PP[FOR]
PP[FOR]/NP[ACC]
>compose
PP[FOR]/
NP[ACC]
NP[3, SG]/NP[ACC]
>compose
NP[3,
SG]/
N[SG]
N[SG]/NP[ACC]
>compose
NP[3,
SG]
NP[3,
SG]
N[SG]/
PP[OF]
PP[OF]/
NP[ACC]
(Sally
Philip)
asked
for
a
picture
of
le]
b.
VP[FIN]
>apply
VP[FIN]/NP[ACC]
<compose
NP[3, SG]
VP[FIN]/NP[ACC]
<compose
VP[VFORM v]\VP[VFORM v]
NP[3,
SG]
VP[FIN]/
NP[ACC]
VP[VFORM v]\
VP[VFORM v]
(Bob)
hit
le]
hard
with clenched fist
the desk he hated

Figure 13

the extraction site to the extracted item in a heavy noun phrase shifted constituent.

Consider now how we could try to continue the analysis in fig. 12 to incorporate the subject. Mick is an NP. If intransitive finite verb phrases were defined to be S[FIN]\NP[NOM] rather than VP[+FIN] then we could use backward application, but this would not help us in fig. 13a because the verb phrase with the gap would be (S[FIN]\NP[NOM])/NP[ACC] - still looking forward for the missing NP - and no existing syntactic rule could apply. STDMN invokes a trick called *type-raising*:

(52) Subject Type-raising ("s⌈")
NP[PER u, NUM v, NOM]:x => S[FIN]/VP[FIN, PER u, NUM v]:L f[f(x)]

[13]

This allows us to complete the analyses in fig.s 12 and 13b by functional application, see fig. 14. We can also follow Subject Type-raising with Forward Composition in fig. 14a but we still cannot complete the analysis because the S is looking forward for the NP, see fig. 15. STDMN invokes another type-raising rule:

(53) Topic Type-raising ("t⌈")
X:x => S[FIN]/(S[FIN]/X):L f[f(x)]
X ∈ {NP, PP, VP, AP, S'}

which allows us to finish; see fig. 16.

---

[13] Note that I have restricted Subject Type-raising to nominative NPs.

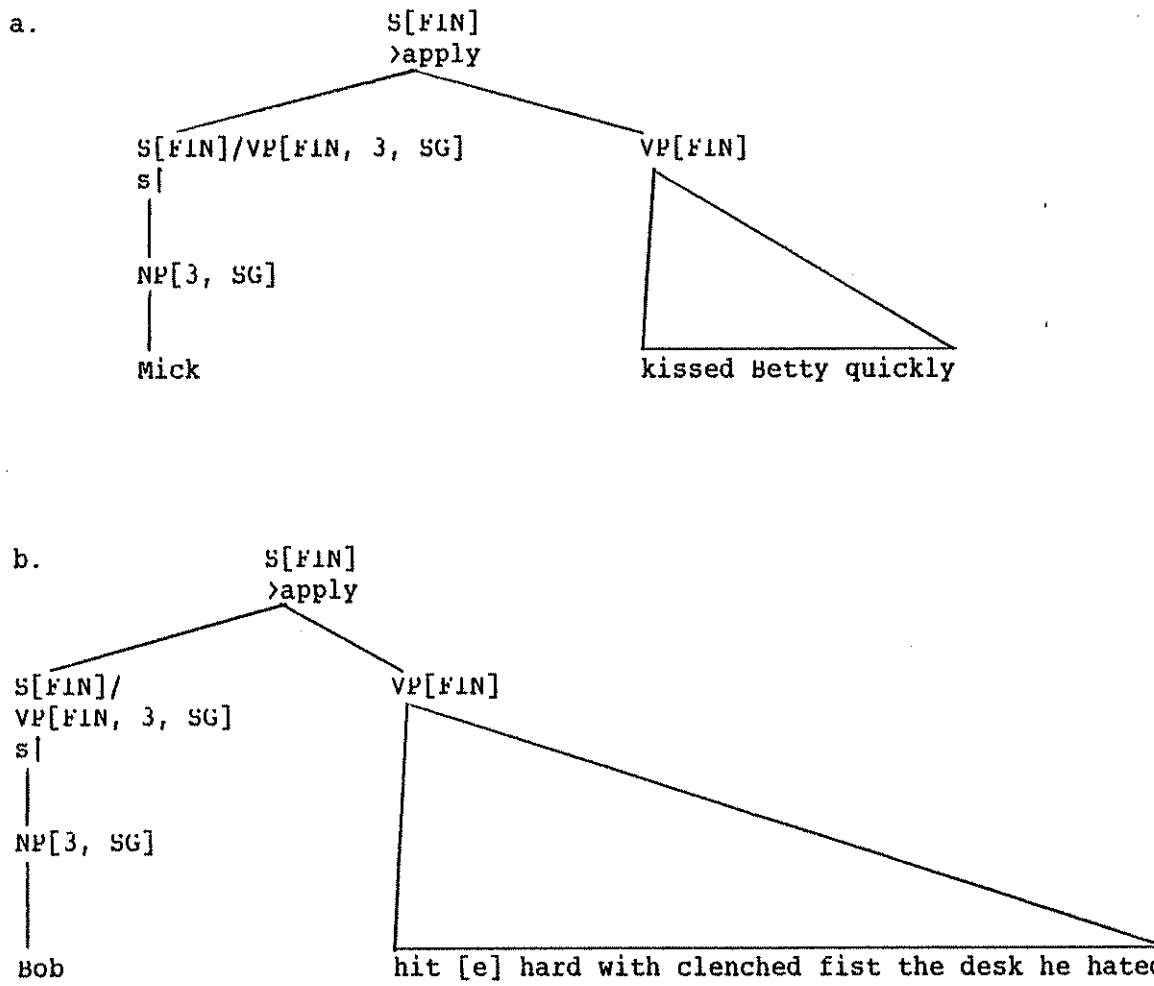
a.
S[FIN]
>apply
S[FIN]/VP[FIN, 3, SG]
s|
NP[3, SG]
Mick
VP[FIN]
kissed Betty quickly
b.
S[FIN]
>apply
S[FIN]/
VP[FIN, 3, SG]
s|
NP[3, SG]
Bob
VP[FIN]
hit [e] hard with clenched fist the desk he hated

Figure 14

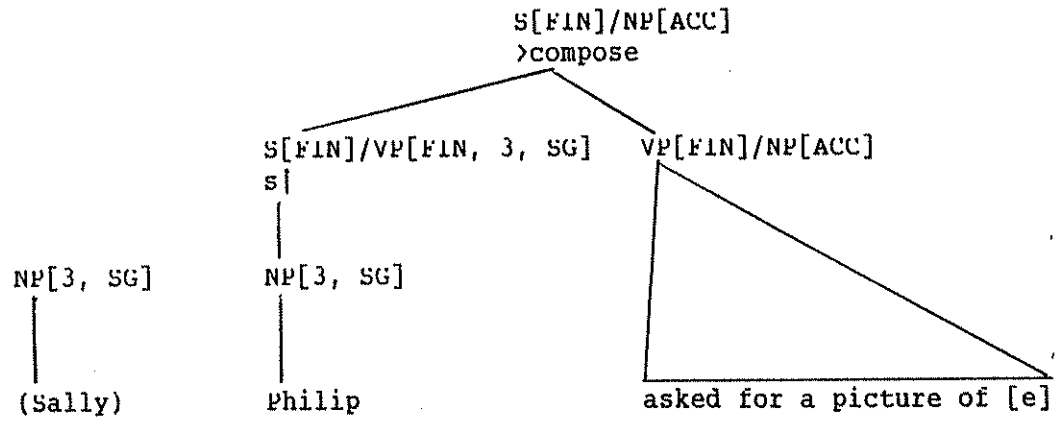

Figure 15

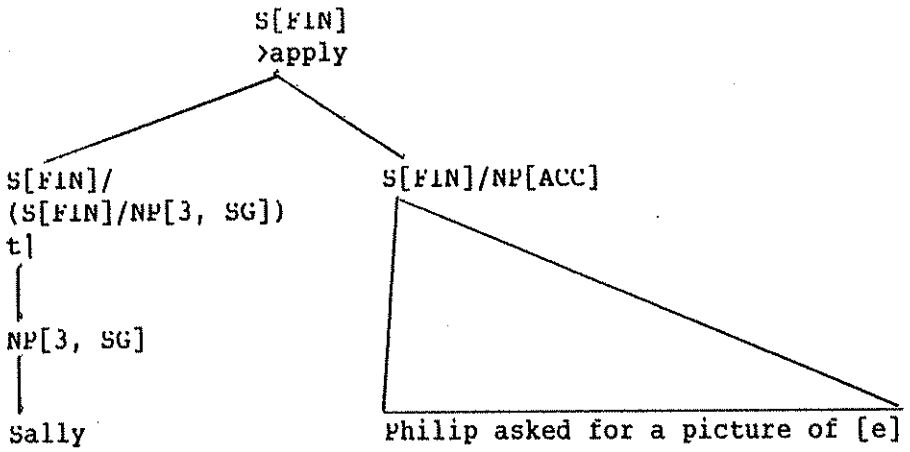

Figure 16

Other combinatory rules are suggested in STDMN but this exhausts those which are necessary for the examination of relative clauses in this paper

(in fact Topic Type-raising is not needed).

2.2.3. Outline of HPSG

In this section I will use abbreviations for HPSG categories as follows. We have square brackets enclosing a list of the feature/feature-value pairs. If a feature-value is associated with only one feature (this turns out to be the case for all and only the atom-valued features), the feature itself is omitted. The value for MAJ may be placed before the bracketed list. So for example V[FIN, SUBCAT <N[3, SG, NOM, SUBCAT <>]>, REL <>, SLASH <>] abbreviates {<MAJ, V>, <VFORM, FIN>, <SUBCAT, <{<MAJ, N>, <PER, 3>, <NUM, SG>, <CASE, NOM>, <SUBCAT, <>>}>>, <REL <>>, <SLASH <>>}. Note also that because REL and SLASH are in the domains of categories which are <MAJ, N>, the category label "{<MAJ, N>, <PER, 3>, <NUM, SG>, <CASE, NOM>, <SUBCAT, <>>}" is interpreted as category {<MAJ, N>, <PER, 3>, <NUM, SG>, <CASE, NOM>, <SUBCAT, <>>, <REL, u>, <SLASH, v>} where u and v are variables marking unconstrainedness.

In section 3 and beyond I will use "NP", "S", and "PP" (outside the square brackets) to abbreviate {<MAJ, N>, <SUBCAT, <>>}, {<MAJ, V>, <SUBCAT, <>>}, and {<MAJ, P>, <SUBCAT, <>>} respectively. So for example, {<MAJ, V>, <VFORM, FIN>, <SUBCAT, <{<MAJ, N>, <PER, 3>, <NUM, SG>, <CASE, NOM>, <SUBCAT, <>>}>>, <REL <>>, <SLASH <>>} will be abbreviated to V[FIN, SUBCAT <NP[3, SG, NOM]>, REL <>, SLASH <>]. As in section 2.2.1. however, I postpone these slightly less obvious abbreviations until after HPSG has been outlined.

Syntactic rules in HPSG divide into *head-complement* rules, *linking* rules, and *coordination* rules. Since coordination is not of immediate relevance to the discussion here, the third type of rule will not be introduced. There is a single Rule Application Algorithm (RAA) which applies at the application of every rule. The version presented here will be simpler than the full version in two respects: both the control and the coordination details are omitted. They are not needed for any analyses here.

HPSG, like GPSG, adopts an ID/LP rule system so that the grammatical rules imply no ordering amongst daughters in a local tree - this aspect is handled by an LP component[14]. The syntactic rules are highly underspecified. Examples of head-complement rules are

(54) a. M -> X, [SUBCAT <x>]
b. M -> [SUBCAT <x, y>], X
c. M -> [SUBCAT <x, y, z>], X2, X1

"M", "X" and "H" are place-holders. "H" marks the head.

The RAA is given in fig. 17. Because the RAA is procedural, the daughters in a local tree could conceivably be interpreted either as the daughter categories before the RAA has applied, or as the daughter categories after it has applied. The former interpretation is more natural and is the one used here. We could use the RAA in a 'declarative' local tree admission procedure as follows. The categories in a local tree would be 'passed by value' to the admission procedure which would use the RAA to generate the set of mother categories that can dominate the daughters. The

[14] POLL however does not mention the ID/LP distinction.

```
Match a category with H;
for X1, X2, ...
  $( do one of the following, according to the type of the rule
       head-complement: $( pop x from SUBCAT of H;
                           either SUBCATEGORIZATION PRINCIPLE
                                  match with X1 a sign which
                                  unifies with x
                           or     GAP INTRODUCTION PRINCIPLE
                                  push x onto SLASH of H
                         $)
       linking          : apply the procedure stated on the rule;
     BINDING INHERITANCE PRINCIPLE
     for each binding feature F
       $( while F of X1 ≠ <>
            $( pop q from F of X1;
               either push q onto F of H (host dependency)
               or     unify q with the top of F of H (parasitic dependency)
            $);
          F of M = F of H
       $)
  $)
HEAD FEATURE PRINCIPLE
for each head feature F
  F of M = F of H
```

Figure 17

admission procedure would then admit the local tree if and only if the actual mother category is one of these.

The analysis in fig. 18 illustrates the use of the Subcategorization Principle (SP) in the RAA, and the rules (54a) and (54b) which are used in local trees (a) and (b) respectively. The SP is the process by which a head is joined to a complement by unifying the complement with the top item in the head's SUBCAT stack. That item is absent from the SUBCAT stack of the mother. We do not want to allow any alternative ordering in (a) or (b) so

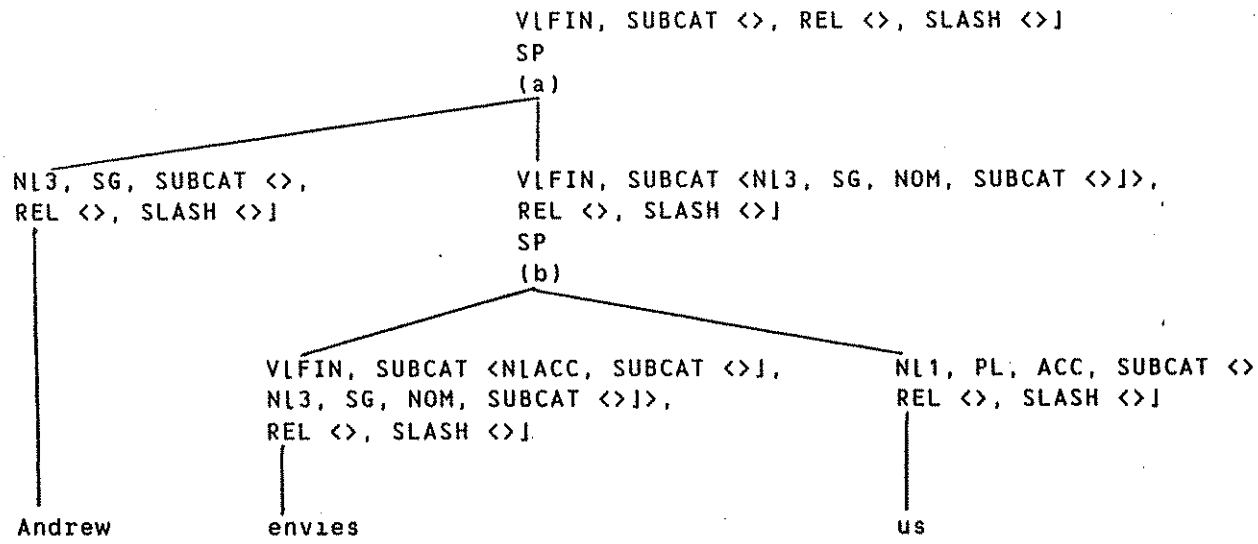

Figure 18

we need LP rules something like

(55) a. V[SUBCAT <x, y>] < X
b. X < V[SUBCAT <x>]

An example of a linking rule is

(56) M -> X, H[V, FIN, SUBCAT <>] [Procedure: SLASH-bind X to H]

The procedure for F-binding p to q, where F is a binding feature and p and q are categories, is as follows: pop r from F of q and unify r with p. So for example if

(57) a. p = N[3, SG, SUBCAT <>, REL <>, SLASH <>]
b. q = V[FIN, SUBCAT <>, REL <N[SUBCAT <>]>, SLASH <>]

then REL-binding p to q gives

(58) a. p = [1 3, SG, SUBCAT <>, REL <>, SLASH <>]
     b. q = [VFIN, SUBCAT <>, REL <>, SLASH <>]

The analysis in fig. 19 illustrates the use of the Gap Introduction Principle (GIP), and the rule (56). The GIP is the process by which an item in a category's SUBCAT stack is moved onto its SLASH stack, corresponding to a gap occuring in that position. In (c) we use (54b) and the GIP, in (b) we use (54a) and the SP. In (a) we use the linking rule (56). The LP rule (55b) constrains the ordering in (b). We do not need an LP rule to constain

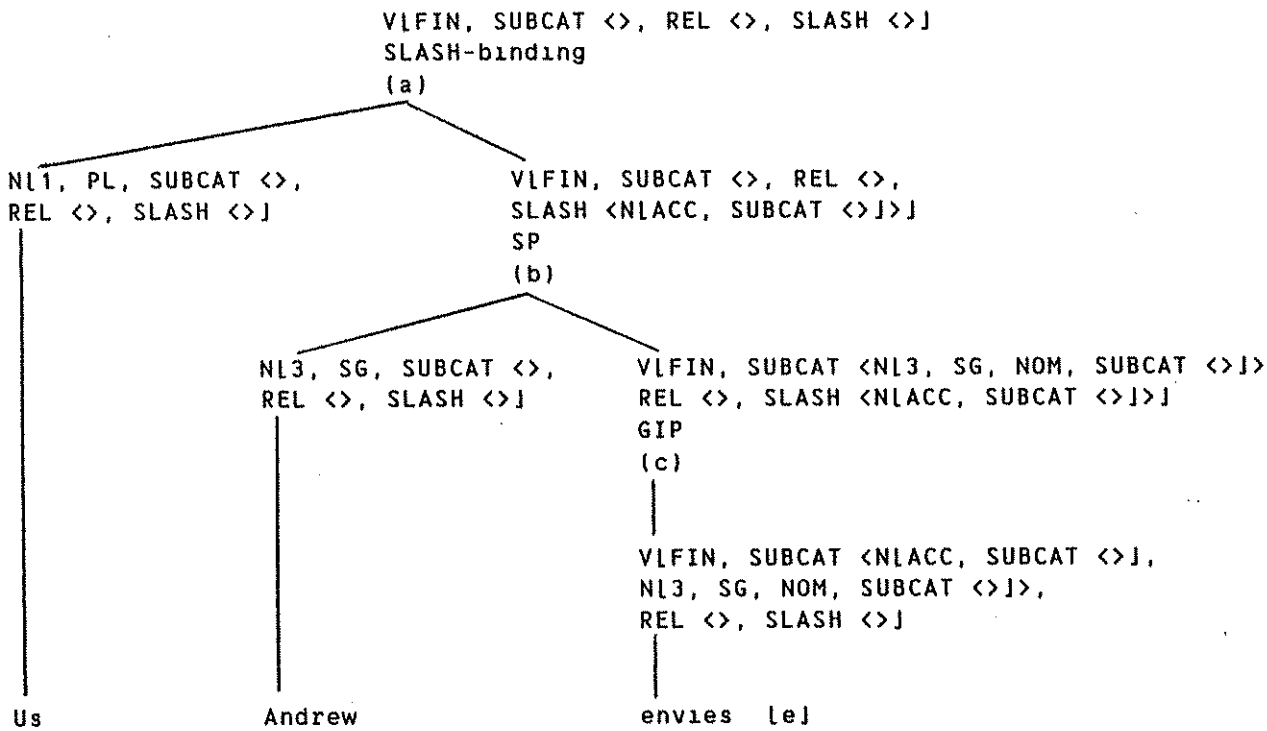

Figure 19

the ordering in (b) but this is only because having the extracted item on the right happens to give the canonical string. In general we need an LP rule like

(59) X < [VISUBCAT <>]

to avoid, say

(60) $[_{S/NP}$ Andrew persuaded [e] to stay] us

3. Analyses of relative clauses

GPSG, CG and HPSG are considered in sections 3.1-3.3 respectively. Each section contains five subsections in which the analyses of the following relative clauses are given.

(61) a. who works for John
b. whom Bill works for
c. whose book Mary loves
d. for whom Fred works
e. a picture of which exists

3.1. Analyses under Generalized Phrase Structure Grammar

The problem with control and CASE discussed in section 2.2.1 applies in all these analyses; the problem with SLASH and the HFC also mentioned there applies in the second, third, and fourth analyses. I construct the analyses as we actually want them.

3.1.1. GPSG: "who works for John"

The analysis is shown in fig. 20. The ID rules for local trees (a)-(d) are

(62) a. S -> X2, H[-SUBJ] (see GKPS p.139, 155, 248)
b. VP -> H[999], PP[for] (assumed; see GKPS p.157)
c. PP -> H1 (assumed; see GKPS p.132)
d. P1 -> H[38], NP (see GKPS p.134, 140, 248)

The LP rule

(63) [+N] < PP < V2 (GKPS p.248)

fixes the ordering in (a), and

(64) SUBCAT < -[SUBCAT] (GKPS p.110, 248)

does so in (b) and (d). These rules actually fix most ordering in the examples here; linear precedence will not be mentioned again except where addi-

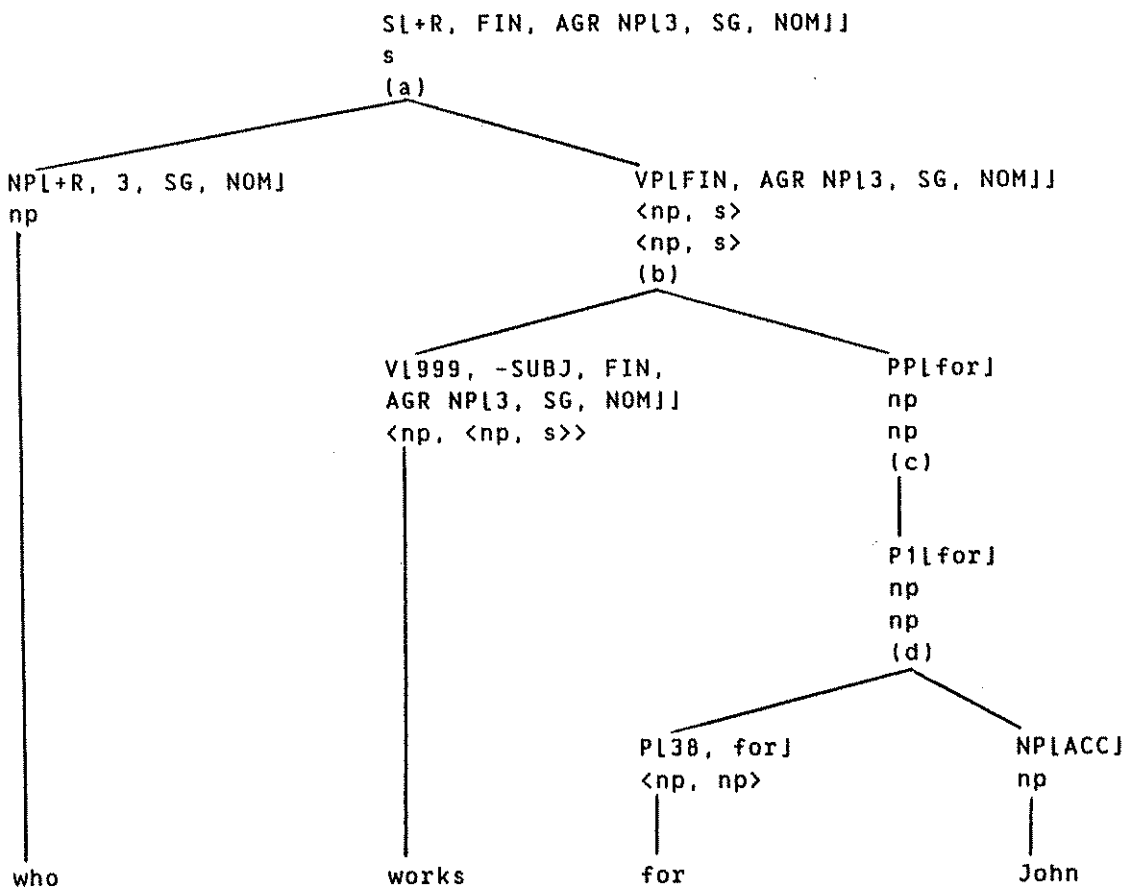

Figure 20

tional rules are required.

3.1.2. GPSG: "whom Bill works for"

The analysis is shown in fig. 21. The ID rules for (a)-(e) are

(65) a. S -> X2, H/X2 (see GKPS p.145, 148, 155, 248)
b. S -> X2, H[-SUBJ] (see GKPS p.139, 155, 248)
c. VP -> H[999], PP[for] (assumed; see GKPS p.157)
d. PP -> H1 (assumed; see GKPS p.132)
e. P1 -> H[38], NP[+NULL] (see GKPS p.143)

The ID rule for (e) is obtained from

(66) P1 -> H[38], NP (GKPS p.134, 140, 248)

by Slash Termination Metarule 1 (STM1; GKPS p.143, 249):

(67) Z -> W, X2
↓
Z -> W, X2[+NULL]

3.1.3. GPSG: "whose book Mary loves"

An analysis is shown in fig. 22. We use the ID rules

(68) a. S -> X2, H/X2 (see GKPS p.145, 148, 155, 248)
b. NP -> NP[+POSS], H1 (see GKPS p.149, 248)
c. S -> X2, H[-SUBJ] (see GKPS p.139, 155, 248)
d. N1 -> H[30] (see GKPS p.247)
e. VP -> H[2], NP[+NULL] (obtained using STM1; see GKPS p.143)

for (a)-(e) respectively. We also need an LP rule for (b):

(69) POSS < -[POSS] (assumed) I have assigned type <n1, np> to the category dominating whose as (68b) seems to require. However in GKPS p.234 <np, np> is suggested. This appears to be an error. Note that whose is viewed as a possessive noun phrase rather than a determiner. This

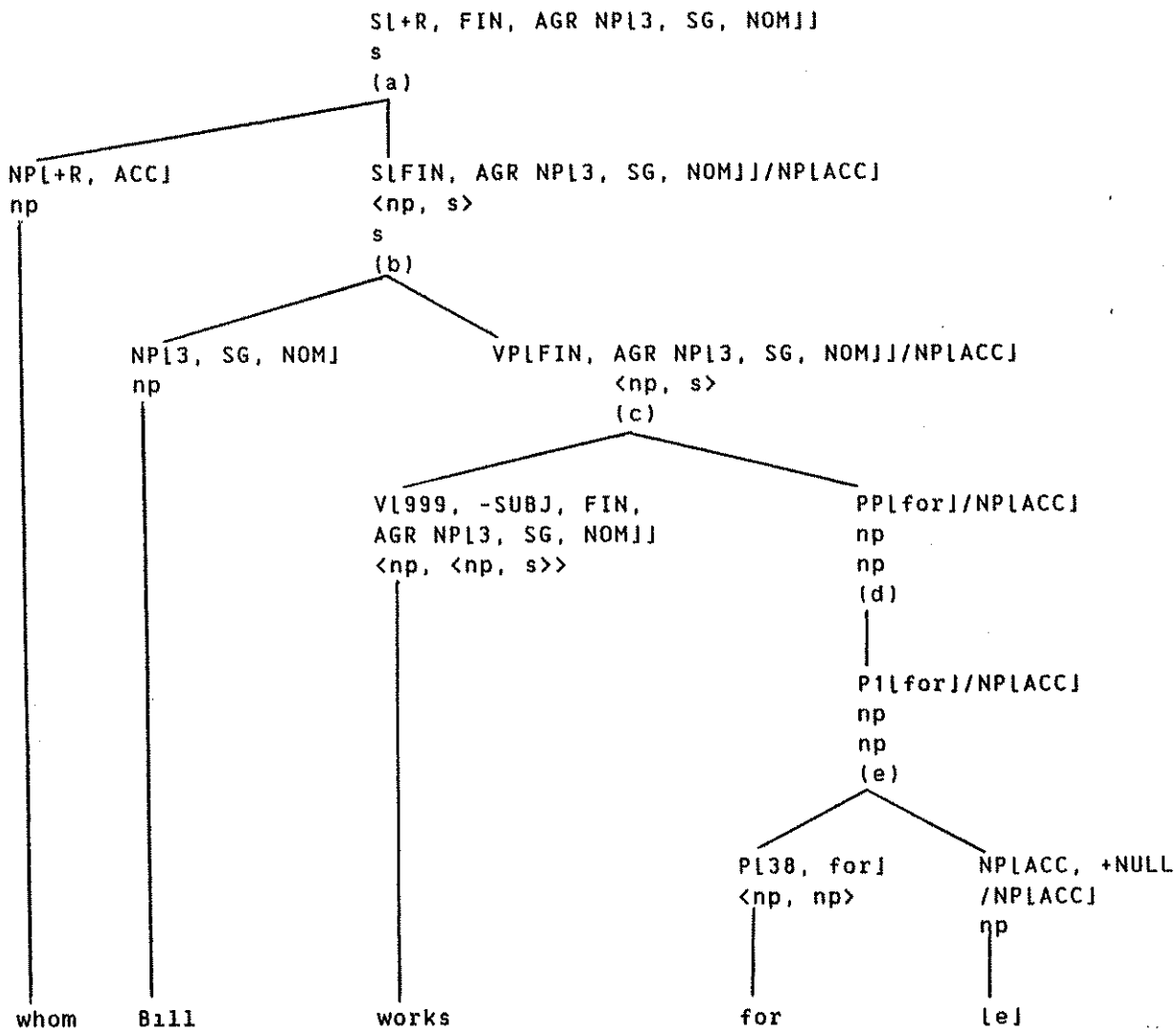
S[+R, FIN, AGR NP[3, SG, NOM]]
s
(a)
NP[+R, ACC]
np
S[FIN, AGR NP[3, SG, NOM]]/NP[ACC]
<np, s>
s
(b)
NP[3, SG, NOM]
np
VP[FIN, AGR NP[3, SG, NOM]]/NP[ACC]
<np, s>
(c)
V[999, -SUBJ, FIN,
AGR NP[3, SG, NOM]]
<np, <np, s>>
PP[for]/NP[ACC]
np
np
(d)
P1[for]/NP[ACC]
np
np
(e)
P[38, for]
<np, np>
NP[ACC, +NULL]
/NP[ACC]
np
whom
Bill
works
for
e

Figure 21

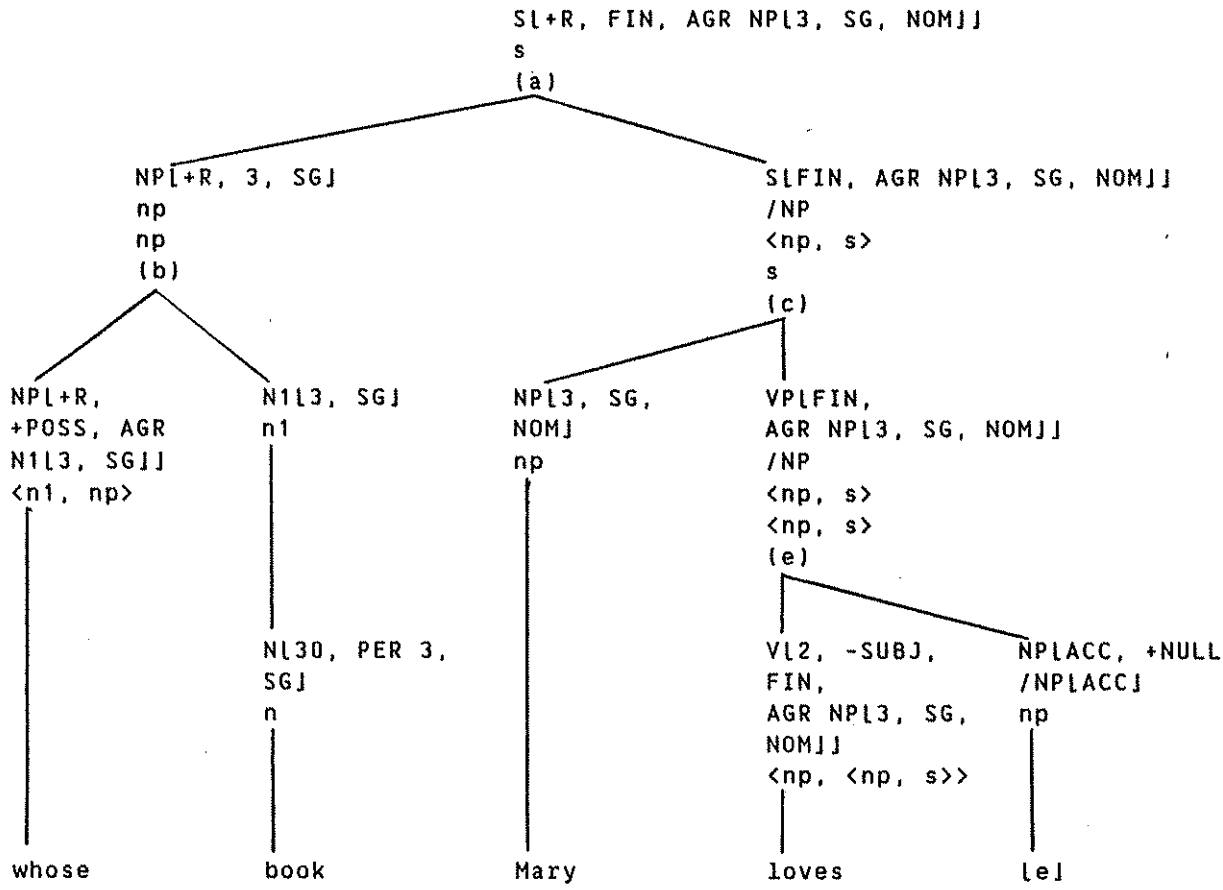

Figure 22

is what is suggested in GKPS p.234 but in GKPS p.153 it is classed as a possessive determiner.

3.1.4. GPSG: "for whom Fred works"

The analysis is given in fig. 23. The ID rules for local trees (a)-(e) are

(70) a. S -> X2, H/X2 (see GKPS p.145, 148, 155, 248)
b. PP -> H1 (assumed; see GKPS p.132)
c. S -> X2, H[-SUBJ] (see GKPS p.139, 155, 248)
d. P1 -> H[38], NP (GKPS p.134, 140, 248)
e. VP -> H[999], PP[for, +NULL]

The ID rule (70e) is obtained from

(71) VP -> H[999], PP[for] (assumed; see GKPS p.157)

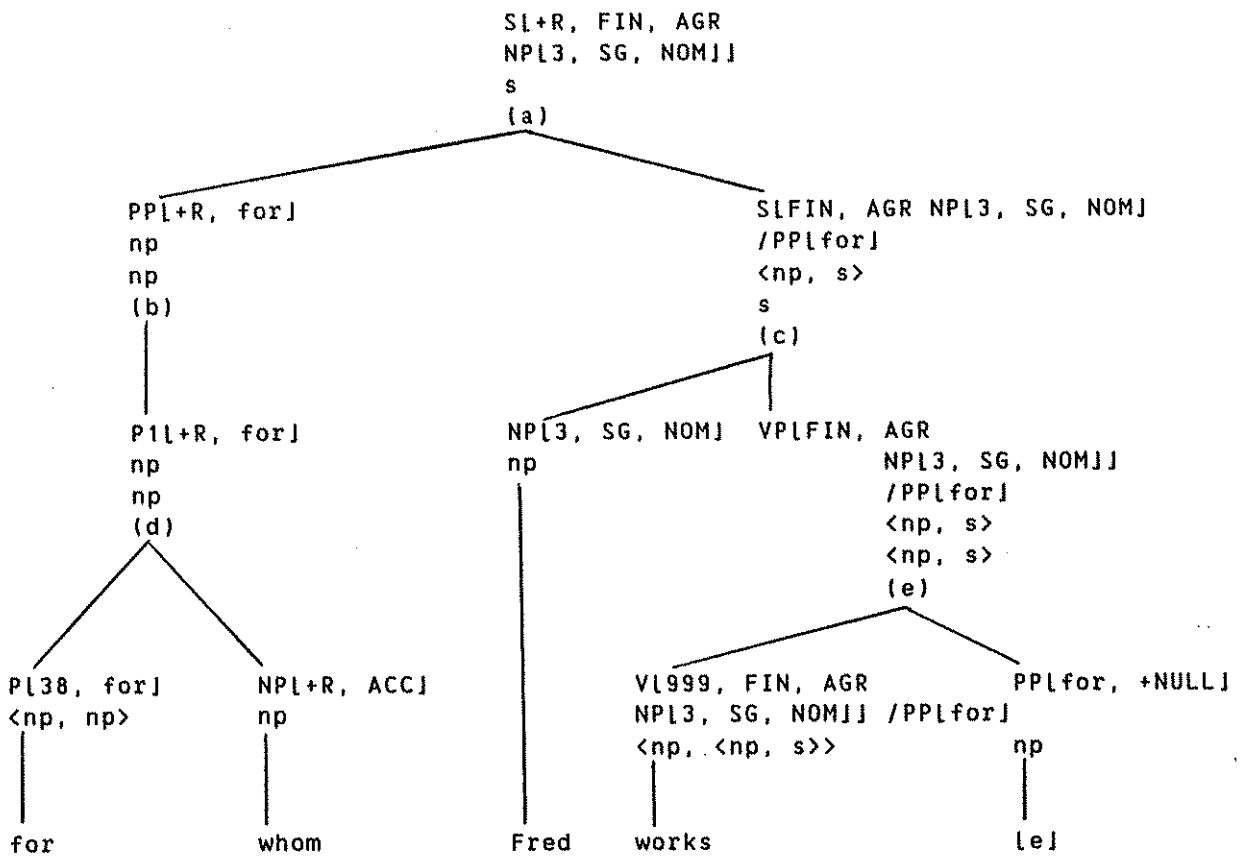

Figure 23

by STM1 (see section 3.1.2).

3.1.5. GPSG: "a picture of which exists"

The analysis is given in fig. 24. The ID rules are

(72) a. S -> X2, H[-SUBJ] (see GKPS p.139, 155, 248)
b. NP -> Det, H1 (assumed; see GKPS p.85, 91, 126, 209)
c. VP -> H[1] (see GKPS p.110, 247)
d. N1 -> H[35], PP[of] (see GKPS p.128, 247)
e. PP -> H1 (assumed; see GKPS p.132)
f. P1 -> H[38], NP (see GKPS p.134, 140, 248)

3.2. Analyses Under Combinatory Grammar

I represent the category of relative clauses by "S[WH X]". This is to be viewed as a notational variant of the actual category which I suggest in section 4 to be NP\NP. Alternatively we could have used "S[WH N]" representing N\N.

The lexical categories for who and whom are S[WH NP[PER u, NUM v]]/VP[FIN, PER u, NUM v] and S[WH NP[PER u, NUM v]]/(S[FIN]/NP[PER u, NUM v]). Since subjects are nominative and all other noun phrases are accusative we do not actually need to mention case. which will have both categories. whose will form either given a noun. The resulting constituent shares its agreement with the noun not the antecedent so, for example, the

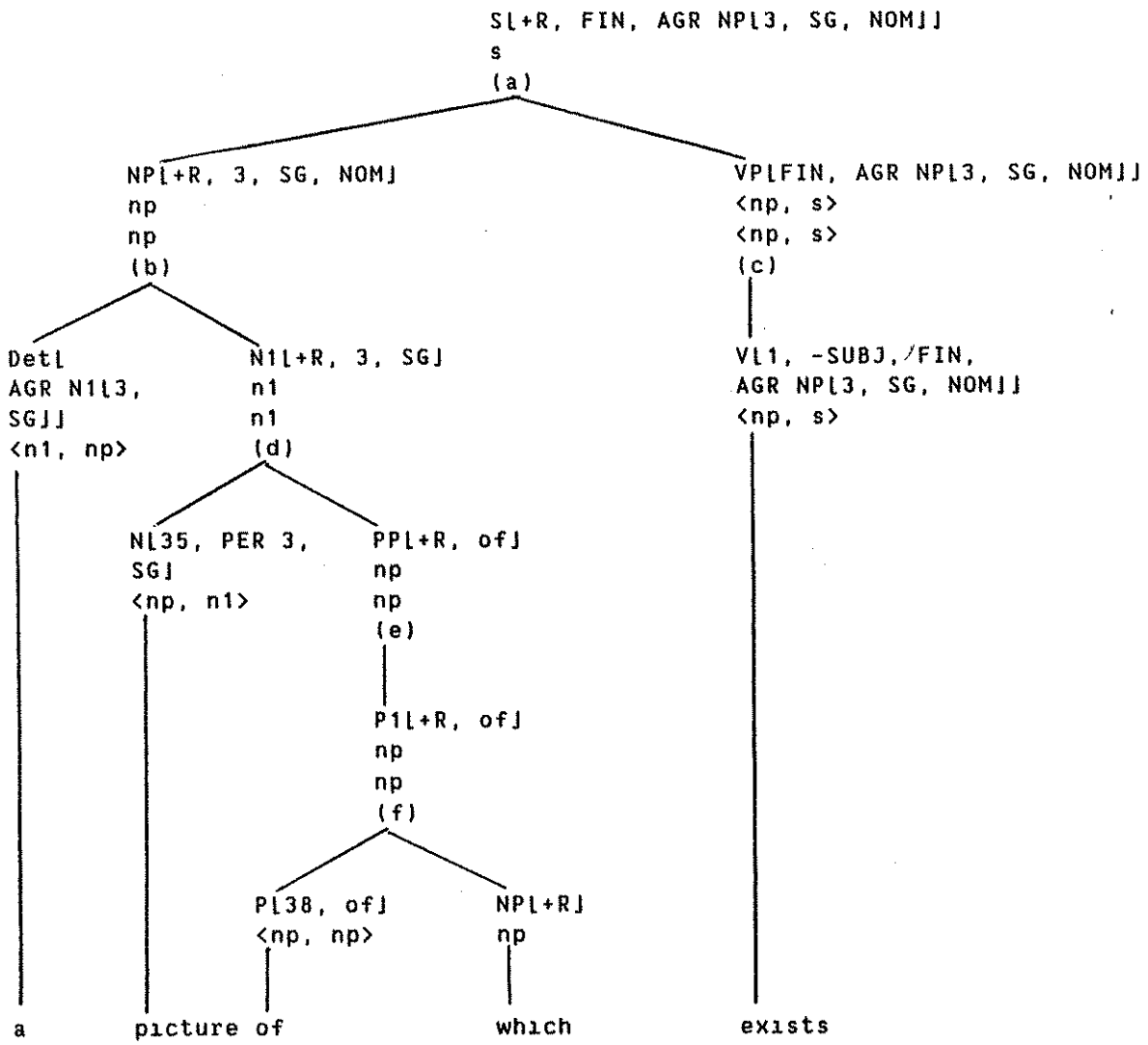

Figure 24

'subject' whose is (S[WH NP]/VP[FIN, 3, NUM X]]/N[NUM X].

3.2.1. CG: "who works for John"

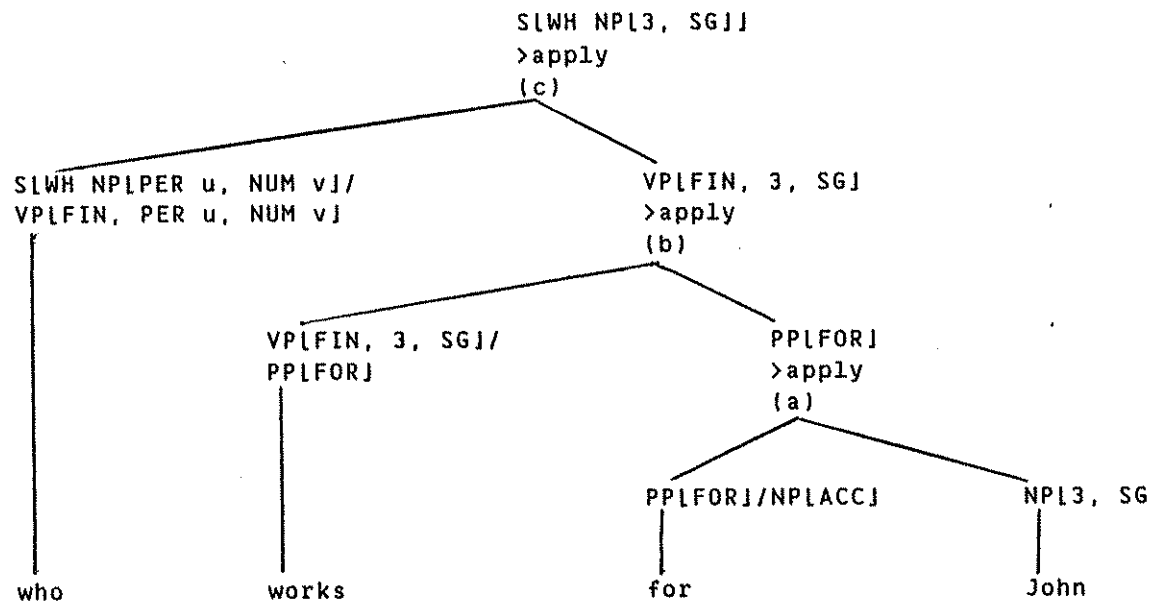

Figure 25

An analysis is given in fig. 25.

3.2.2. CG: "whom Bill works for"

An analysis is given in fig. 26.

3.2.3. CG: "whose book Mary loves"

An analysis is given in 27. whose has the same syntactic type, NP/N, as

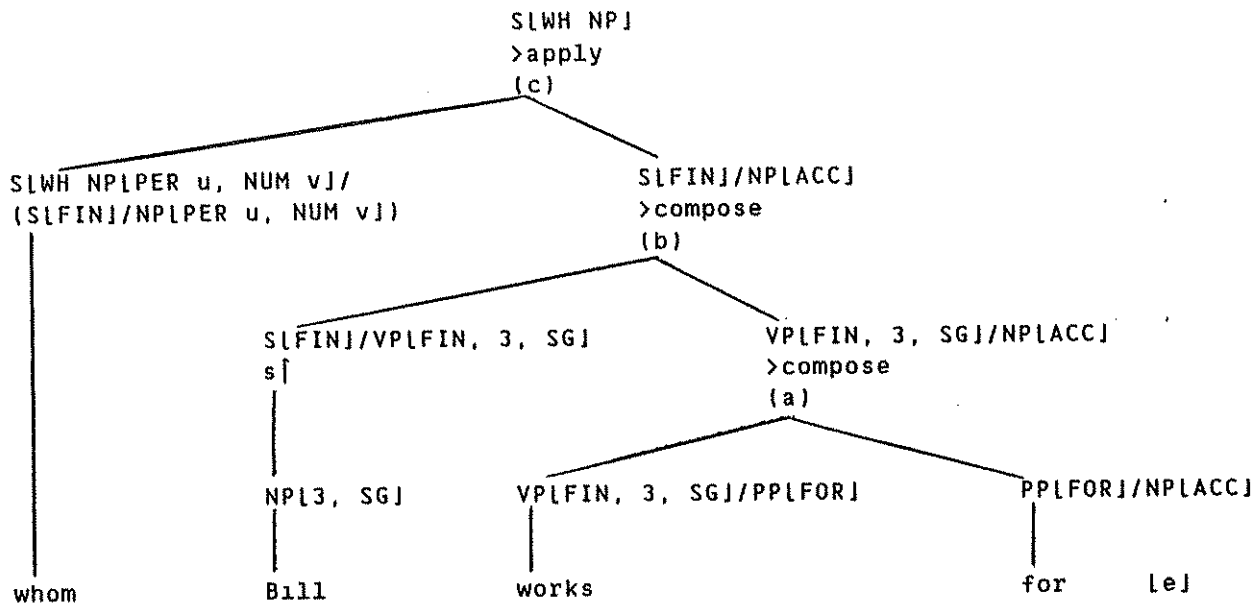
S[WH NP]
>apply
(c)
S[WH NP[PER u, NUM v]/
(S[FIN]/NP[PER u, NUM v])
S[FIN]/NP[ACC]
>compose
(b)
S[FIN]/VP[FIN, 3, SG]
s
VP[FIN, 3, SG]/NP[ACC]
>compose
(a)
NP[3, SG]
VP[FIN, 3, SG]/PP[FOR]
PP[FOR]/NP[ACC]
whom
Bill
works
for
e]

Figure 26

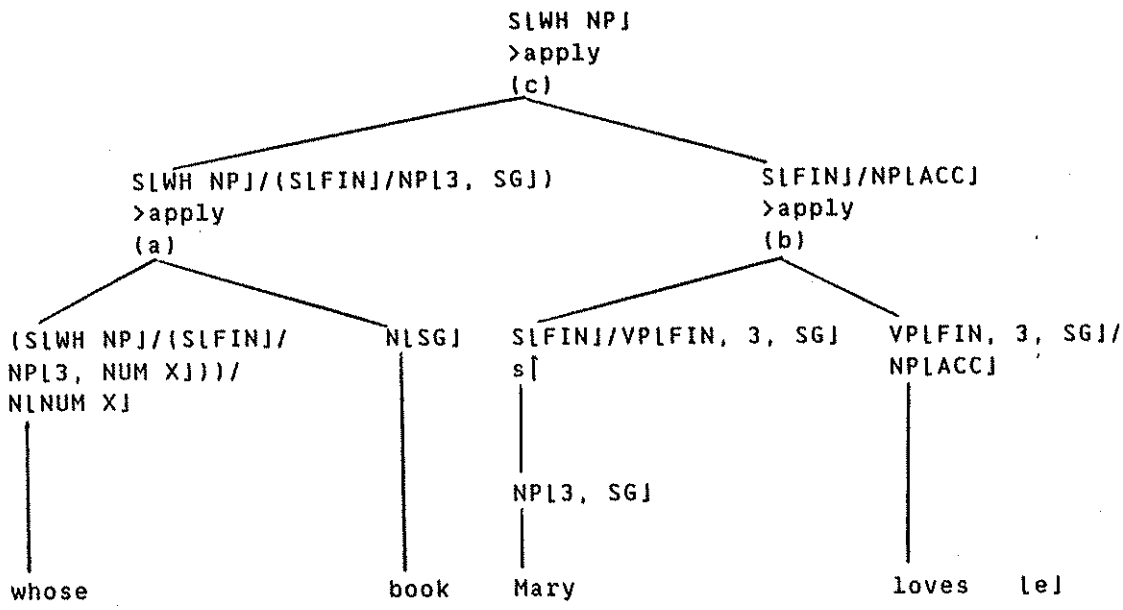

Figure 27

determiners.

3.2.4. CG: "for whom Fred works"

Pied piping such as that which occurs in this example and the example in the next section seems to be a clear instance of feature percolation and as such a challenge for the CG framework. For the examples here I use the following object relative pronoun type-raising rule[15]

(73) Object Relative Pronoun Type-raising ("or[")
S[WH NP[PER u, NUM v]]/(S[FIN]/NP[PER u, NUM v]):F =>
(S[WH NP[PER u, NUM v]]/(S[FIN]/Z))\(Z/NP[PER u, NUM v]):LGLH[F(Lx[H(Gx)])]

---

[15] This rule, and the type-raising rule in the next section, are not taken from STOMN.

The semantics have been included but will not be discussed. S\WH NP]/(S\FIN]/X) corresponds to a category X carrying the WH **FOOT** feature in GPSG. Thus the type-raising rule corresponds to

(74) $NP_{WH} \Rightarrow Z_{WH}\backslash(Z/NP)$

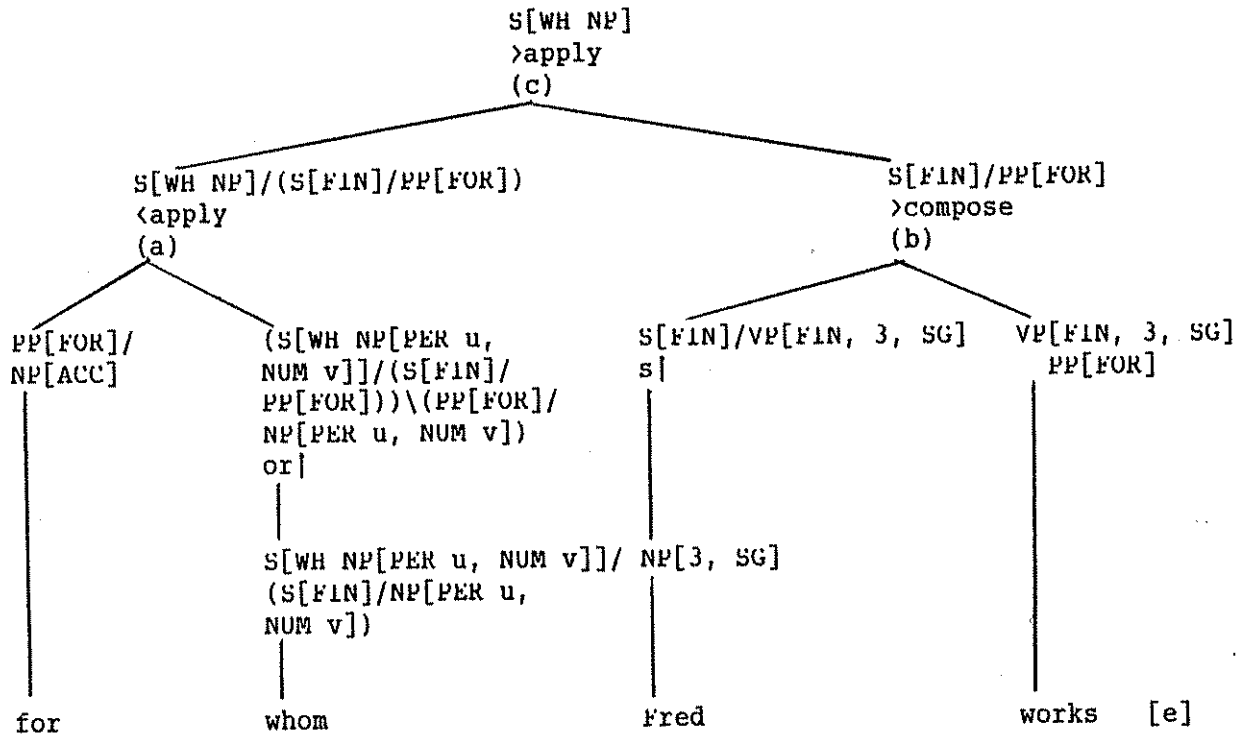

Figure 28

The analysis for the relative clause is given in fig. 28.

3.2.5. CG: "a picture of which exists"

The analysis is given in fig. 29. We need another, similar, type raising rule whereby whom or which will be able to combine backwards with NP/NP to form a constituent which can act as a subject:

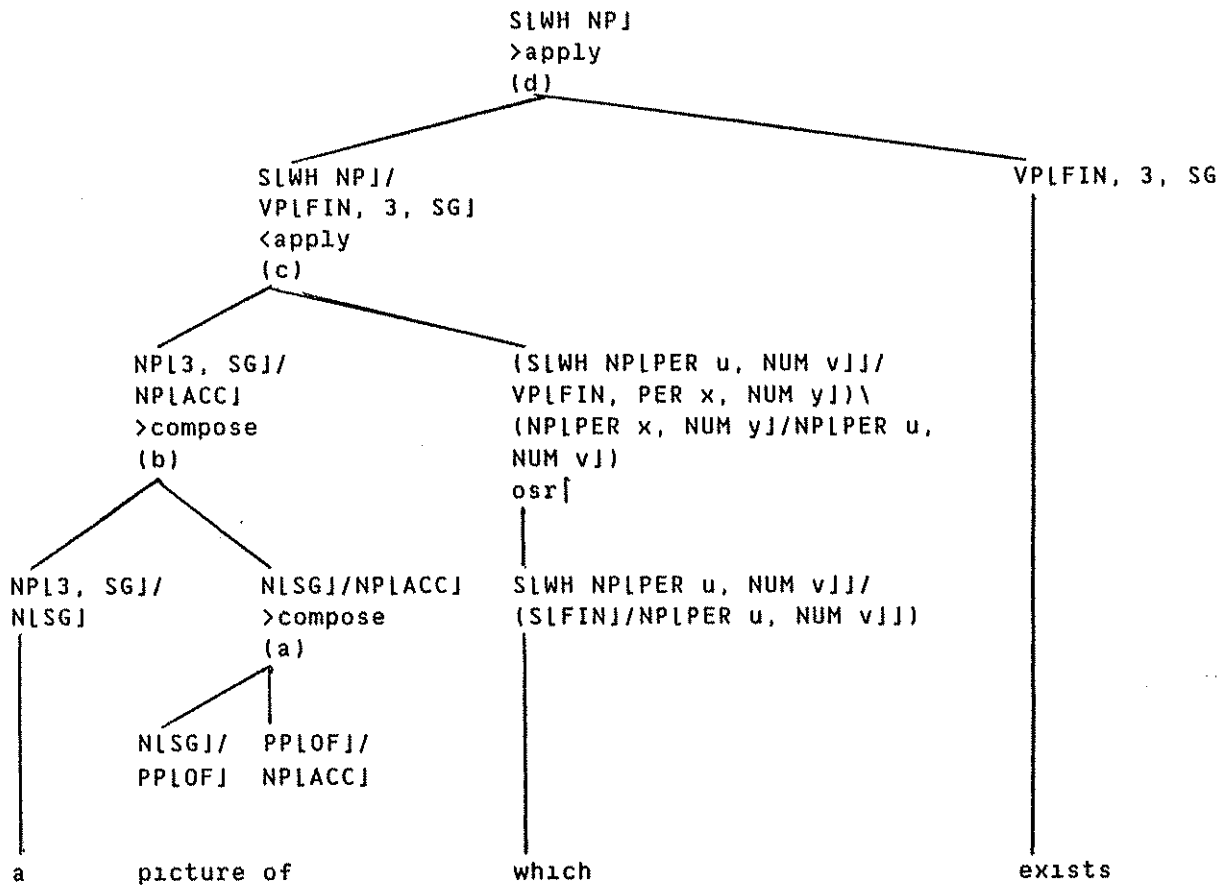

Figure 29

(75) Object-to-Subject Relative Pronoun Type-raising ("osr[")
S[WH NP[PER u, NUM v]]/(S[FIN]/NP[PER u, NUM v]) =>
(S[WH NP[PER u, NUM v]]/VP[FIN, PER x, NUM y])
\(NP[PER x, NUM y]/NP[PER u, NUM v])

3.3. Analyses under Head-driven Phrase Structure Grammar

The following LP rules suffice to prevent spurious linear ordering in the analyses given here.

(76) a. subject-before-predicate
X < V[SUBCAT <x>]
b. verb-before-complement and noun-before-complement
[SUBCAT <x, y>] < X
c. preposition-before-NP
P[SUBCAT <x>] < X
d. antecedents-left-of-relatives and leftward-extraction
X < S
e. nouns-right-of-determiners[16]
X < N[SUBCAT <x>]

The category for who is

(77) NP[PER u, NUM v, NOM, REL <NP[PER u, NUM v]>, SLASH <>]

whom is the same but accusative. which is the same except for being unspecified for case since it may take either. I have treated whose as a

---

[16] In HPSG nouns subcategorize for determiners since head features appear to percolate from the noun. In CG the converse is assumed: determiners

determiner rather than as a possessive noun phrase.

3.3.1. HPSG: "who works for John"

The analysis is shown in fig. 30. Rules for local trees (a)-(c) are

(78) a. -> H[SUBCAT <X>], C (see POLL p.6)
b. -> H[SUBCAT <X, Y>], C (see POLL p.6)
c. -> H[SUBCAT <X>], C (see POLL p.6)

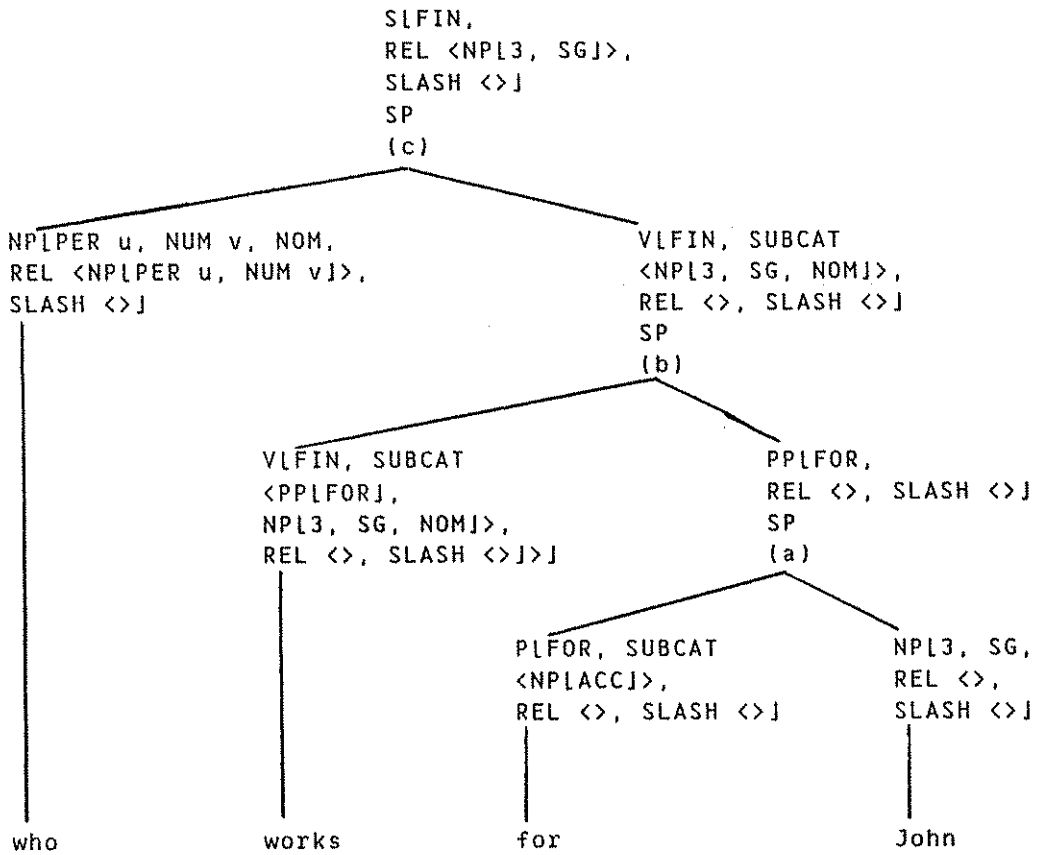

Figure 30

3.3.2. HPSG: "whom Bill works for"

The analysis is given in fig. 31. Grammatical rules for local trees (a)-(d) are

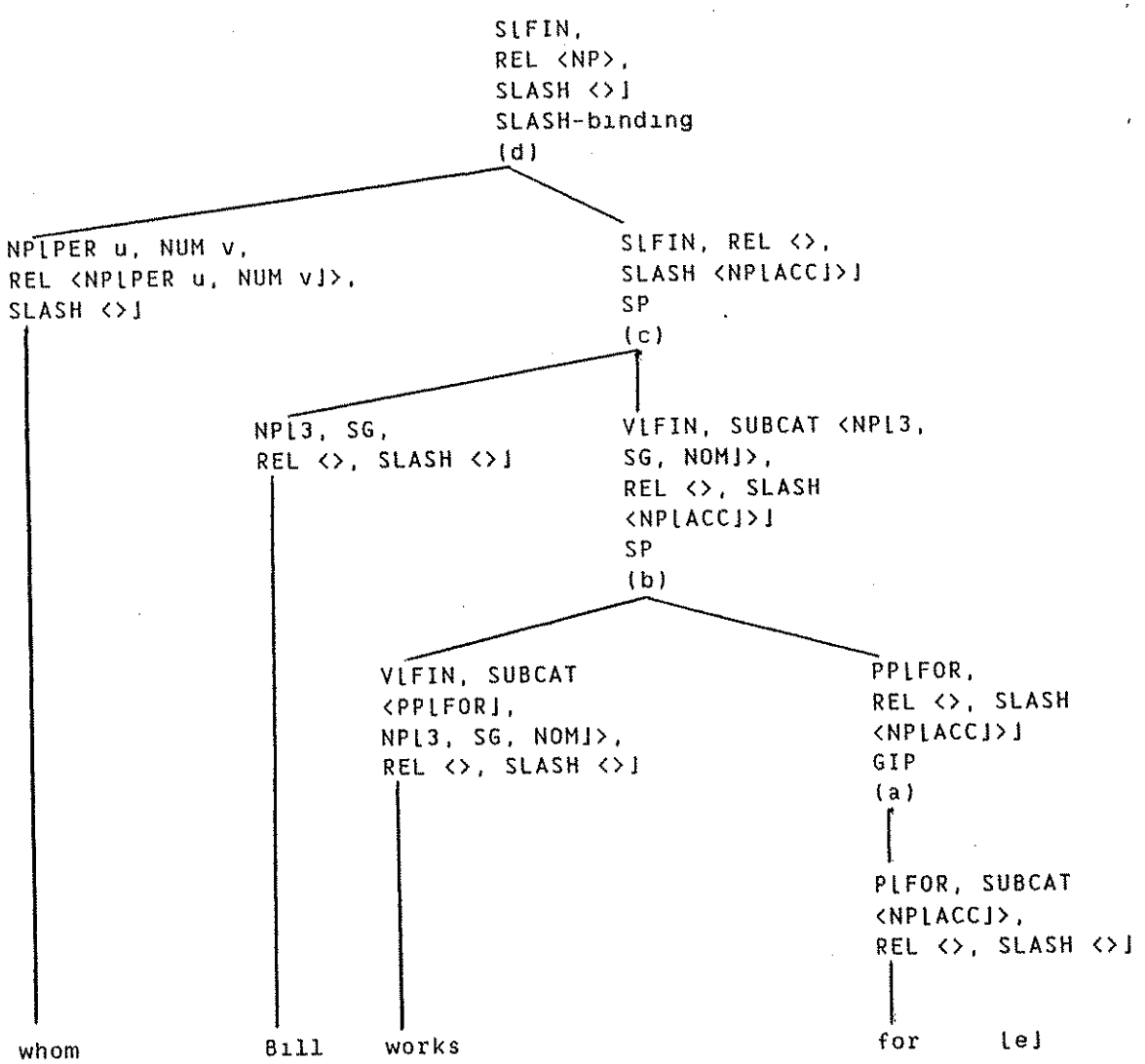

Figure 31

(79) a. -> [SUBCAT <X>], C (see POLL p.6)
b. -> [SUBCAT <X, Y>], C (see POLL p.6)
c. -> [SUBCAT <X>], C (see POLL p.6)
d. -> [MAJ V, FIN, SUBCAT <>], X [Procedure: SLASH-bind X to H] (see POLL p.25)

Note that the Gap Introduction Principle is used in (a). In (d) we are using a linking rule.

3.3.3. HPSG: "whose book Mary loves"

The analysis is given in fig. 32. Rules for (a)-(d) are

(80) a. -> [SUBCAT <X, Y>], C (see POLL p.6)
b. -> [SUBCAT <X>], C (see POLL p.6)
c. -> [SUBCAT <X, Y>], C (see POLL p.6)
d. -> [MAJ V, FIN, SUBCAT <>], X [Procedure: SLASH-bind X to H] (see POLL p.25)

3.3.4. HPSG: "for whom Fred works"

The analysis is in fig. 33. Grammatical rules for local trees (a)-(d) are

(81) a. -> [SUBCAT <X>], C (see POLL p.6)
b. -> [SUBCAT <X, Y>], C (see POLL p.6)
c. -> [SUBCAT <X>], C (see POLL p.6)
d. -> [MAJ V, FIN, SUBCAT <>], X [Procedure: SLASH-bind X to H] (see POLL p.25)

respectively.

3.3.5. HPSG: "a picture of which exists"

The analysis is shown in fig. 34. Grammatical rules for local trees (a)-(d) are

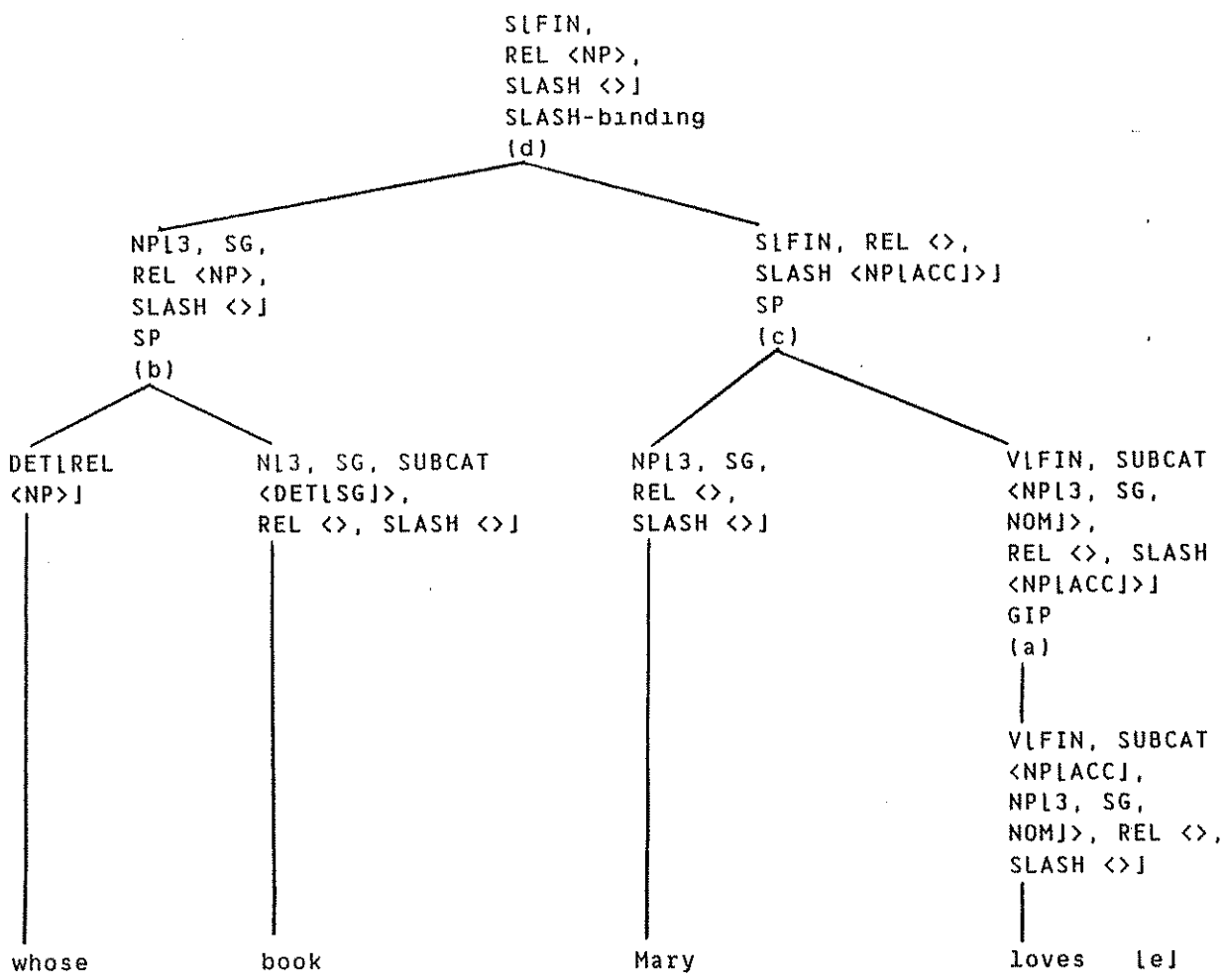
S[FIN,
REL <NP>,
SLASH <>]
SLASH-binding
(d)
NP[3, SG,
REL <NP>,
SLASH <>]
SP
(b)
S[FIN, REL <>,
SLASH <NP[ACC]>]
SP
(c)
DET[REL
<NP>]
whose
N[3, SG, SUBCAT
<DET[SG]>,
REL <>, SLASH <>]
book
NP[3, SG,
REL <>,
SLASH <>]
Mary
V[FIN, SUBCAT
<NP[3, SG,
NOM]>,
REL <>, SLASH
<NP[ACC]>]
GIP
(a)
V[FIN, SUBCAT
<NP[ACC]>,
NP[3, SG,
NOM]>, REL <>,
SLASH <>]
loves
[e]

Figure 32

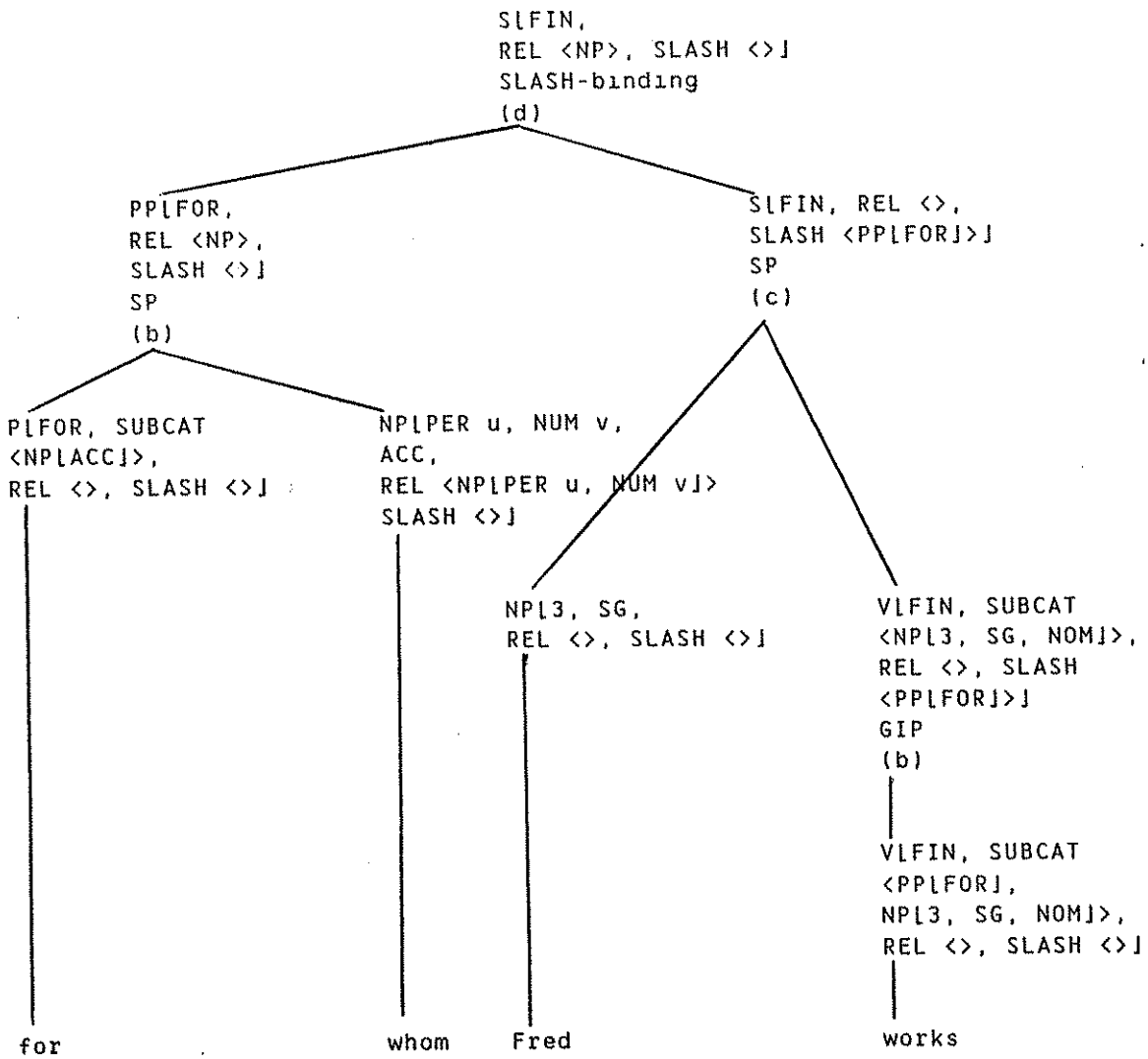
S[FIN,
REL <NP>, SLASH <>]
SLASH-binding
(d)
PP[FOR,
REL <NP>,
SLASH <>]
SP
(b)
S[FIN, REL <>,
SLASH <PP[FOR]>]
SP
(c)
P[FOR, SUBCAT
<NP[ACC]>,
REL <>, SLASH <>]
NP[PER u, NUM v,
ACC,
REL <NP[PER u, NUM v]>
SLASH <>]
NP[3, SG,
REL <>, SLASH <>]
V[FIN, SUBCAT
<NP[3, SG, NOM]>,
REL <>, SLASH
<PP[FOR]>]
GIP
(b)
V[FIN, SUBCAT
<PP[FOR],
NP[3, SG, NOM]>,
REL <>, SLASH <>]
for
whom
Fred
works

Figure 33

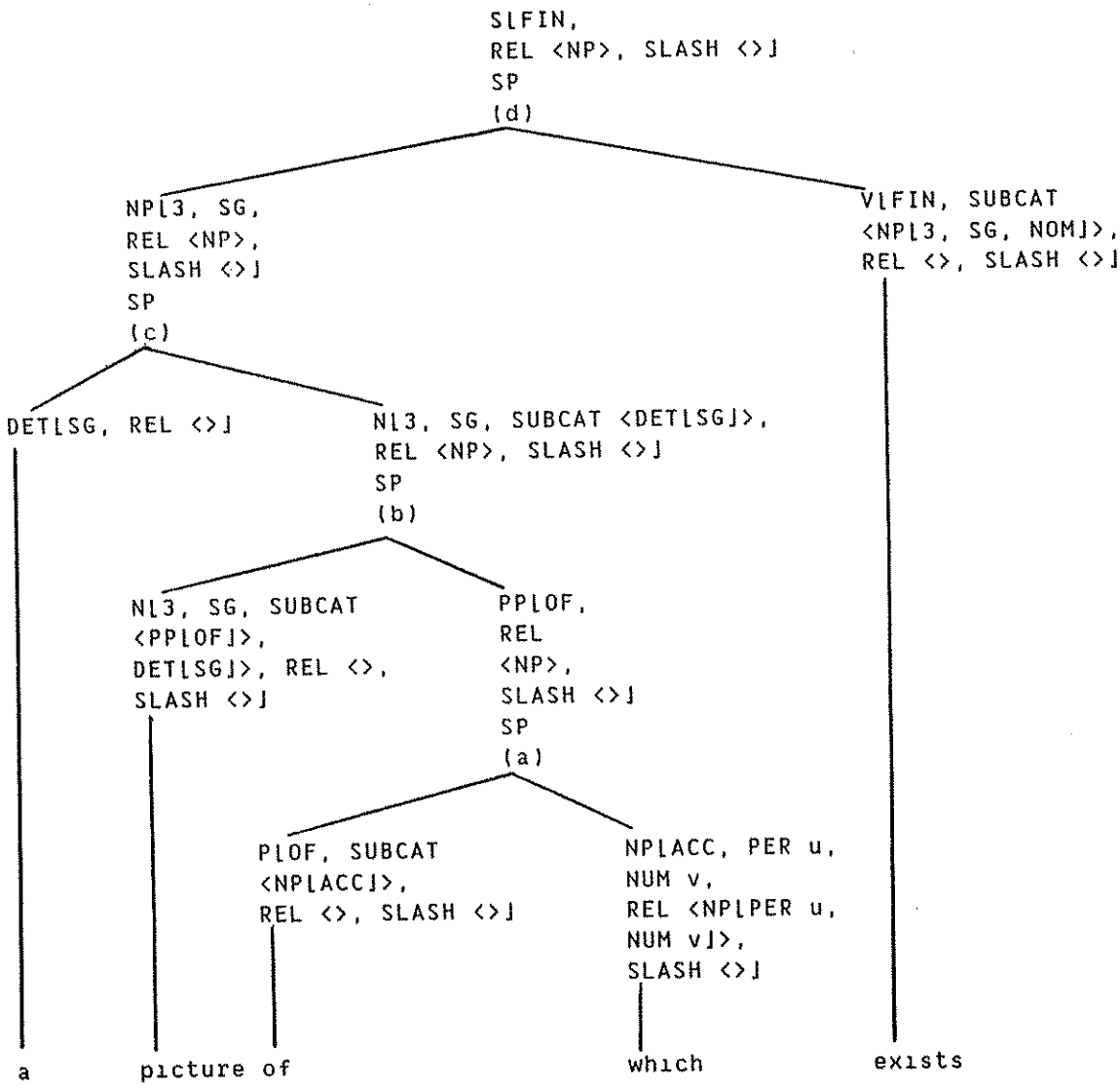

Figure 34

(82) a. -> H[SUBCAT <X>], C (see POLL p.6)
b. -> H[SUBCAT <X, Y>], C (see POLL p.6)
c. -> H[SUBCAT <X>], C (see POLL p.6)
d. -> H[SUBCAT <X>], C (see POLL p.6)

respectively.

4. Incorporating the antecedent

It was mentioned in section 1 that there is uncertainty about the category of the constituent that a relative clause combines with. To get the correct semantics of say

(83) Every man whom Bill works for [e] won the pools

we would like the determiner to join after the relative clause. Thus assuming *every* has an interpretation something like LPLQ∀x[P(x)->Q(x)], we can combine this with "man whom Bill works for" with interpretation Ly[man'(y)&Bill-works-for'(y)] to give LQ∀x[[man'(x)&Bill-works-for'(x)]->Q(x)] for the noun phrase. But it is hard to see how we can get this interpretation if "every man" is a constituent with interpretation LQ∀x[man'(x)->Q(x)].

So semantically we want the relative clause to combine with a nominal constituent lower in the X-bar hierarchy than noun phrases. Further evidence for relatives combining with lower constituents is provided by the fact that combining a restrictive relative clause to a proper name is dubious:

(84) ?Cecil whom Bill works for [e] likes apples

However with appositive intonation the string is fine:

(85) Cecil, whom Bill works for [e], likes apples

Sentences like

(86) He who tries will be rewarded

sound slightly antiquated and/or formal but do not have comma intonation and the relative clause is joined to a noun phrase (assuming the Montague tradition by which pronouns are full NPs). A strong argument for the noun phrase antecedent hypothesis is provided by cases such as

(87) [A boy and a girl] who ran away together have returned home

where the coordinated antecedent is undoubtedly a noun phrase.

If we say that restrictive relative clauses join to a Noun or Nom then we must provide extra apparatus to get appositive relatives such as (85) and coordinated cases such as (87). If we say that all relatives have noun phrase antecedents then our semantics becomes harder and we must look elsewhere than syntactic structure for an explanation of the differences in intonation between restrictive and appositive relatives. I prefer the noun phrase antecedent story and this is the line I shall follow in dealing with CG and HPSG. In GPSG relative clauses join to N1 antecedents.

A relative pronoun must agree with its antecedent in number[17]:

(88) a. The mountain which stands before us was once under sea
b. *The mountains which stands before us were once under sea
c. *The mountain which stand before us was once under sea
d. The mountains which stand before us were once under sea

Assuming we allow pronoun antecedents, it will be seen that a relative pronoun also agrees with its antecedent in person:

(89) a. He who has tried will be rewarded
b. *I who has tried will be rewarded
c. *He who have tried will be rewarded
d. I who have tried will be rewarded

In GPSG relative clauses join to N1 sisters by the ID rule

(90) N1 -> H, S[+R] (see GKPS p.155, 248)

The top part of the analysis of

(91) the man whom Bill works for [e]

is given in fig. 35. It seems to me that there is no mechanism covering agreement between the relative pronoun and its antecedent. Presumably such a link was supposed to be mediated by WH's value, and the CAP would somehow cover this feature as well as the other two category-valued features AGR and SLASH. As things are then, GPSG fails to block examples like (88b) and (88c). Because antecedents are Noms, none of (89) are obtained.

I have said that I want to treat relative clauses as NP\NP in CG. Thus the "S[WH NP[NUM u, PER v]]" used earlier means NP[NUM u, PER v]\NP[NUM u, PER v]. Given this category, joining a relative clause to its antecedent is a simple matter of backward application; see fig. 36.

---

[17] Stricly speaking because relative pronouns don't have different inflectional forms we should say that the antecedent must meet the agreement requirements of the position occupied by the relative pronoun.

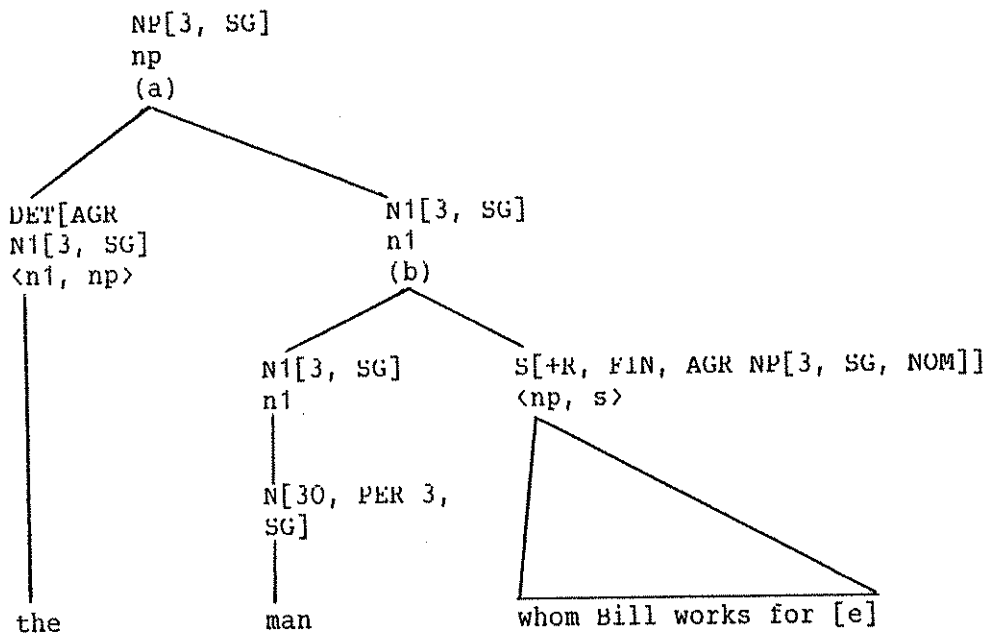

Figure 35

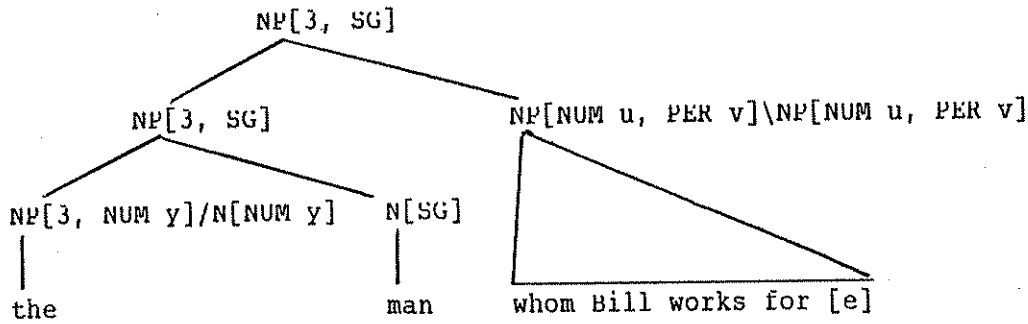

Figure 36

In HPSG we require a linking rule to do this job:

(92) M -> H, X[V, FIN, SUBCAT <>, SLASH <>] [Procedure: REL-bind H to X]

Fig. 37 illustrates its use.

We now turn to gender. It was noted in section 1 that which must have a neuter antecedent and who, whom, and whose must have non-neuter ones. To cover these facts in CG and HPSG we simply need to introduce a gender feature. So in CG the lexical category for who, say will be

(93) (NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/
(S[FIN]\NP[PER u, NUM v, GEN w])
w ∈ {MASC, FEM, NEUT}

(using S\NP instead of VP). In HPSG it will be

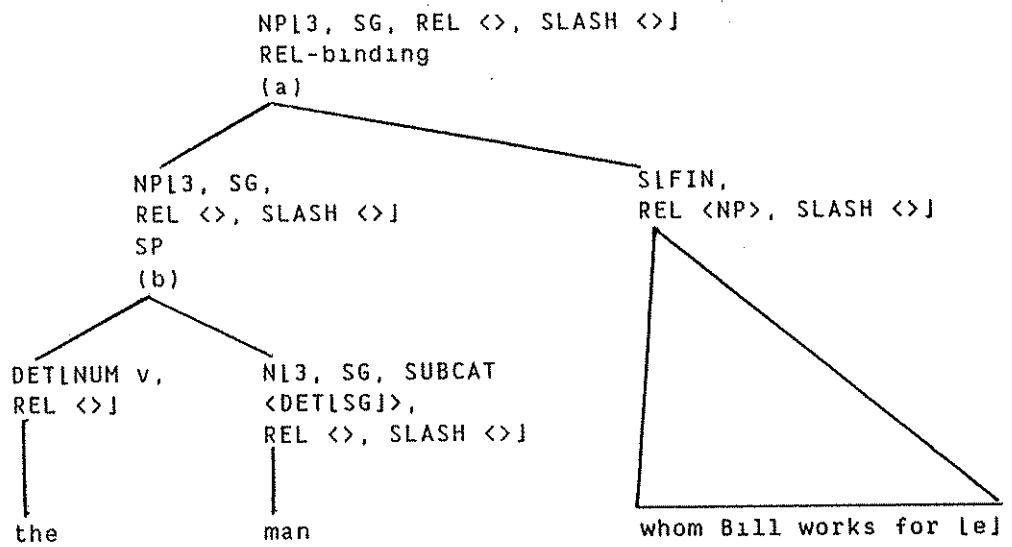

Figure 37

(94) [PER u, NUM v, GEN w, NOM, SLASH <>,
REL <[PER u, NUM v, GEN w]>]
w ∈ {MASC, FEM, NEUT}

As was mentioned at the beginning of section 3.2, with whose, which I have viewed in HPSG as a determiner rather than a possessive noun phrase, the number, gender and person of the antecedent is entirely irrelevant to agreement within the relative clause; it is the noun that whose binds to which is important there. Thus

(95) The girls whose hero sacrificed himself cried

So in HPSG whose simply has lexical entry

(96) [DET[REL <NP>]]

as was used in section 3.3.3. In CG the 'subject' whose will be ((NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/(S[FIN]\NP[3, NUM y, GEN z]))/N[NUM y, GEN z].

5. Summary of data, grammars, and implementations

In this section I summarise the data addressed in this paper by presenting ten example sentences which contain the relative clauses analysed in section 3 and which exhibit certain agreement phenomena discussed in section 4. I also give GPSG, CG and HPSG grammars intended to cover this data, and discuss the implementations of each of these grammars. The implemented grammatical rules and lexicons are listed in the appendices. There also are copies of the parsing systems' responses to each of the ten example sentences. The implemented grammars appear to operate correctly.

5.1. Summary of data

The first two example sentences are

(97) a. The man who works for John exists
b. *The men who works for John exist

The second sentence is ungrammatical because within the relative clause, who must be singular to agree with its verb, yet its antecedent is plural. The pair thus illustrate relative pronoun/antecedent agreement for number.

The sentences

(98) a. The man whom Bill works for exists
b. *The firm whom Bill works for exists

are used to illustrate the fact that whom cannot have a neuter antecedent.

The fifth and sixth examples are

(99)a. The man whose book Mary loves exists
b. The men whose book Mary loves exist

intended to show that whose may have any number antecedent (of course the person and gender are also unconstrained).

The case-sensitivity of who and whom is seen in

(100) a. The person for whom Fred works exists
b. *The person for who Fred works exists

The latter is ungrammatical because prepositions require accusative noun phrase complements and who is nominative. The relative clause is pied piped.

The last two test sentences are

(101) a. A book a picture of which exists exists
b. *A man a picture of which exists exists

Again there is pied piping; the fact that which requires a neuter antecedent is also illustrated.

5.2. Summary of grammars and implementations

The GPSG, CG and HPSG grammars required to cover the data in section 5.1 are summarised in sections 5.2.1, 5.2.2, and 5.2.3 respectively. The implemen-

tatıons of the grammars are also dıscussed ın those sectıons.

5.2.1. Generalızed Phrase Structure Grammar grammar

The GPSG ID rules, LP statements, and metarules requıred for the data consıdered ın thıs paper are summarısed ın fıg. 38. Because of the dıfficulties wıth some feature percolatıon (SLASH and the HFC, CASE and the CAP, and relatıve pronoun/antecedent agreement), and because of the uncertaınty about FSDs and the lexıcon and lexıcal admıssabılıty rules, I have not consıdered ıt worthwhıle to try to construct a complete grammar under GPSG as presented ın GKPS. However I do construct and ımplement a full grammar under GPSG as presented ın Gazdar (1982). Thıs grammar was run under a system buılt by Thompson and Phıllıps (see Thompson and Phıllıps (1984)). The grammar formalısm buılt ınto thıs system, and GPSG as defıned ın GKPS, dıffer quıte wıdely; some of these dıfferences are dıscussed below.

The Thompson/Phıllıps system uses context-free rules rather than ID/LP rules, and I made use of no default mechanısms, so the only types of grammar rules were ordınary grammatıcal rules, and metarules. The only feature percolatıon conventıon ıs a head feature percolatıon conventıon. Thıs conventıon ıs of hıghest prıorıty: even ıf dıfferences ın head feature values on mother and head are stıpulated ın a grammatıcal rule, the head feature conventıon overıdes and forces the mother to share the daughter's specıfıcatıon. It was found convenıent to make POSS and CASE head features even though they are not ın GKPS.

```
Subject-predicate ID rule
S -> X2, H[-SUBJ]                    (see GKPS p.139, 155, 248)

Lexical ID rule for verb 'work'
VP -> H[999], PP[for]                (assumed; see GKPS p.157)

P2/P1 ID rule
P2 -> H1                             (assumed; see GKPS p.132)

Lexical ID rule for NP-requiring prepositions
P1 -> H[38], NP                      (see GKPS p.134, 140, 248)

Left-extraction NP ID rule
S -> X2, H/X2                        (see GKPS p.145, 148, 155, 248)

Possessive NP ID rule
NP -> NP[+POSS], H1                  (see GKPS p.149, 248)

Lexical ID rule for complementless nouns
N1 -> H[30]                          (see GKPS p.247)

Lexical ID rule for verb 'love'
VP -> H[2], NP                       (see GKPS p.110, 247)

Determiner-noun ID rule
N2 -> Det, H1                        (assumed; see GKPS p.85, 91, 126, 209)

Lexical ID rule for noun 'picture'
N1 -> H[35], PP[of]                  (see GKPS p.128, 247)

Relative clause antecedent ID rule
N1 -> H, S[+R]                       (see GKPS p.155, 248)


LP rules:
[+N] < PP < VP                       (see GKPS p.248)
SUBCAT < -[SUBCAT]                   (see GKPS p.110, 248)
DET < [+N]                           (assumed)
POSS < -[POSS]                       (assumed)


Slash Termination Metarule 1
Z -> W, X
    ∀
Z -> W, X[+NULL]
```

Figure 38

The system allows no category valued features, so relative pronoun information was encoded in three atom-valued features RPER, RNUM and RGEN. Naturally these percolate like **FOOT** features. Because there is no counterpart to the FFP it was necessary to write these features into most grammatical rules[18]. Similarly because there is no CAP, subject-verb agreement was done explicitly.

Not even the SLASH feature is properly category-valued. All it carries is the major category and a bar level. So for example if we want to left-extract a 'for'-prepositional phrase, we lose the 'for' part. The restriction of metarules to lexical rules is lifted. This means that it was possible to use metarules to give slashed counterparts to each ordinary grammatical rule.

As I used the system, the lexical insertion rule was such that the category dominating any lexical item contained all and only the feature specifications in one of the lexical item's lexical categories. Each lexical category was written out in full; there was no mechanism for unifying feature values in the lexicon. It was possible to adhere to the 'constant domain' hypothesis of section 2.1 except it didn't seem right to assign PER, NUM, GEN or CASE values to <u>whose</u>, assumed to be NP[+POSS].

---

[18] The FFP allows **FOOT** features to percolate up from any daughter, so it might seem that we would require several grammatical rules corresponding to the different percolation permutations possible. However, for the examples here **FOOT** features only ever percolate up from non-heads, and we never need to give what amounts to the same rule twice.

The grammatical rules and lexicon are given in appendix A, along with parses for the example sentences in section 5.1.

5.2.2. Combinatory Grammar grammar

The grammatical rules required by CG to cover the data considered in this paper are listed in fig. 39. Note that I am now using S\NP rather than VP. Examples of lexical entries are shown in fig. 40.

```
>apply
X/Y  Y  =>  X

<apply
Y  X\Y  =>  X

>compose
X/Y  Y/Z  =>  X/Z

s⌈
NP⌊PER u, NUM v, NOM, GEN w⌋  =>  S⌊FIN⌋/(S⌊FIN⌋\NP⌊PER u, NUM v, GEN w⌋)

or⌈
(NP⌊PER u, NUM v, GEN w⌋\NP⌊PER u, NUM v, GEN w⌋)/
                (S⌊FIN⌋/NP⌊PER u, NUM v, GEN w⌋)  =>
((NP⌊PER u, NUM v, GEN w⌋\NP⌊PER u, NUM v, GEN w⌋)/(S⌊FIN⌋/Z))\
                (Z/NP⌊PER u, NUM v, GEN w⌋)

osr⌈
(NP⌊PER u, NUM v, GEN w⌋\NP⌊PER u, NUM v, GEN w⌋)/
                (S⌊FIN⌋/NP⌊PER u, NUM v, GEN w⌋)  =>
((NP⌊PER u, NUM v, GEN w⌋\NP⌊PER u, NUM v, GEN w⌋)/
                                (S⌊FIN⌋\NP⌊PER x, NUM y, GEN z⌋))\
        (NP⌊PER x, NUM y, GEN z⌋/NP⌊PER u, NUM v, GEN w⌋)
```

Figure 39

This Combinatory Grammar grammar was implemented under Jo Calder's PIMP system (see Calder (1985)) which is an implementation of the PATR-II grammar formalism. PATR-II (see Shieber *et al.* (1983)) is a simple but versatile grammar formalism. Categories are represented by directed acyclic graphs

---

John	NP[3, SG, MASC]
works	(S[FIN]\NP[3, SG, NOM])/PP[FOR]
of	PP[OF]/NP[ACC]
a	NP[3, SG, GEN w]/N[SG, GEN w]
person	N[SG, COMM]
who	(NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/ (S[FIN]\NP[PER u, NUM v, GEN w]) w ∈ {MASC, FEM, NOM}
whom	(NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/ (S[FIN]/NP[PER u, NUM v, GEN w]) w ∈ {MASC, FEM, NOM}
which	(NP[PER u, NUM v, NEUT]\NP[PER u, NUM v, NEUT])/ (S[FIN]\NP[PER u, NUM v, NEUT])
which	(NP[PER u, NUM v, NEUT]\NP[PER u, NUM v, NEUT])/ (S[FIN]/NP[PER u, NUM v, NEUT])
whose	((NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/ (S[FIN]\NP[3, NUM y, GEN z]))/N[NUM y, GEN z] w ∈ {MASC, FEM, NOM}
whose	((NP[PER u, NUM v, GEN w]\NP[PER u, NUM v, GEN w])/ (S[FIN]/NP[3, NUM y, GEN z]))/N[NUM y, GEN z] w ∈ {MASC, FEM, NOM}

Figure 40

(DAGs). A part of a DAG is referenced by following a *path* consisting of sequences of arc labels separated by colons. Such a path amounts to a complex feature name. The DAG so referenced is the corresponding feature value and may consist of a single node, so that the value is atomic, or a complex DAG corresponding to a category- or stack-value. Grammatical rules consist of a phrase structure part and a set of *unifications*. Each unification consists of two paths which start with a symbol occuring in the phrase structure part of the rule. The unification is interpreted as requiring that the DAGs led to by these paths in the categories picked out by their initial symbols are identical. A grammatical rule as a whole is interpreted as admitting a local tree if and only if each of its unifications is satisfied.

The formalism is such that the absence of a feature from a category, i.e. the absence of a path in a DAG, is always interpreted us unconstrainedness: the category may unify with any specification for that feature. This means that we cannot, for example, try to indicate the absence of slash from a CG category by having no slash path, because the DAG would unify with any slash specification. Rather, the absence of slash must be positively marked, e.g. by having a feature "slash" with value "no".

I do not have sufficient space to properly discuss the specific CG and HPSG grammars in the appendices. From a linguistic point of view the details of the implementation are not crucial; what is important is that the CG grammar in fig.s 40 and 39 was implemented and performed correctly over

the data listed in section 5.1.

5.2.3. Head-driven Phrase Structure Grammar grammar

The syntactic rules required by HPSG to cover the data in section 5.1 are listed in fig. 41. Some lexical entries are given in fig. 42. This HPSG grammar was also implemented using PIMP. It may seem surprising that HPSG's

---

Head-complement rules

M -> X, H[SUBCAT <x>]

M -> H[SUBCAT <x, y>], X

Linking rules

M -> X, H[MAJ V, FIN, SUBCAT <>, REL <>] [Procedure: SLASH-bind X to H]

M -> H, X[MAJ V, FIN, SUBCAT <>, SLASH <>] [Procedure: REL-bind H to X]

LP rules

X < V[SUBCAT <x>]

[SUBCAT <x, y>] < X

P[SUBCAT <x>] < X

X < S

X < N[SUBCAT <x>]

Figure 41

```
John     [N 3, SG, MASC, SLASH <>, REL <>]

works    [V FIN, SUBCAT <PP[FOR], NP 3, SG, NOM]>,
         REL <>, SLASH <>]

of       [P OF, SUBCAT <NP ACC]>, REL <>, SLASH <>]

a        [DET SG, REL <>]

person   [N 3, SG, COMM, SUBCAT <DET SG]>, REL <>, SLASH <>]

who      [NP PER u, NUM v, GEN w, NOM,
         REL <NP PER u, NUM v, GEN w]>, SLASH <>]
         w <{MASC, FEM, NEUT}

whom     [NP PER u, NUM v, GEN w, ACC,
         REL <NP PER u, NUM v, GEN w]>, SLASH <>]
         w <{MASC, FEM, NEUT}

which    [NP PER u, NUM v, NEUT,
         REL <NP PER u, NUM v, NEUT]>, SLASH <>]

whose    [DET REL <NP>]
```

Figure 42

RAA can be mimicked in such a declarative system as PATR-II. However, because we have a simplified RAA and because we are dealing with limited cases of binding (i.e. GPSG's **FOOT**) feature percolation, we do not in general require several rules doing much the same work. The exception to this is in joining determiners to nouns. Because whose is treated as a determiner, we need one rule to cover the case where the determiner passes up the REL binding feature, and another to cover cases such as "a [picture of whom]" where the noun passes up the REL binding feature.

PATR-II does not provide for LP statements - it imposes order within the grammatical rules. Therefore it was necessary to repeat head-complement rules for the different head-complement combinations we want to allow.

The implemented grammar rules and lexicon are listed along with sample output on the test sentences in appendix C.

6. Discussion

One way to compare the grammar formalisms is to look at GPSG, HPSG, and CG as a sequence of grammars in which increasing amounts of information are put in the lexicon. GPSG is a theory which concerns itself with the well-formedness of trees completely independently of the lexicon. The subcategorization properties of lexical items are specified in ID rules with which lexical items are associated by integer SUBCAT values. In HPSG lexical categories encode subcategorization properties directly by means of a SUBCAT stack and in CG the information is encoded directly in categories' slash structures.

This movement of subcategorization information from syntactic rules to lexical categories means that metarules can be dropped. Metarules in GPSG apply to lexical ID rules only, i.e. only those rules containing subcategorization information. Now once such information is shifted into categories we can have ordinary grammar rules doing the work of metarules. In the course of this paper for example, we have seen how GPSG requires a metarule, STM1, to obtain left extraction; HPSG and CG require no such metagrammatical device. GPSG's lexical constraint on metarules appears to be a strong argument against GPSG's approach and for HPSG's and CG's category-encoding of subcategorization properties.

Because HPSG incorporates a lexicon, features filter up and GPSG's FCRs, and FSDs, designed to modulate instantiation, are not required. So HPSG inherits from GPSG only non-lexical ID rules, LP rules, and the feature percolation conventions; lexical ID rules, metarules, FCRs, and FSDs are made unnecessary by HPSG's use of the lexicon.

In CG, linear precedence information is also shifted into the lexicon. CG contains no concept of head feature percolation. In effect, this idea too is fixed within each lexical entry as follows. Head-complement pairs in phrase structure grammar correlate with function-argument pairs in CG. Now in CG, a function category 'contains' its mother category left of its slash, so the head features appearing on that mother can be specified directly in the lexical category.

The major difference between the phrase structures grammars and CG is in phenomena involving **FOOT** features (called 'binding' features in HPSG). It was stated at the beginning that the basic phenomena exhibited by relatives are anaphora and movement, and these are just those phenomena most closely involved with **FOOT** features. We have seen how in the phrase structure grammars GPSG and HPSG the bridge between a relative pronoun and its antecedent is made by **FOOT** feature percolation; in CG the percolation effect is obtained by type-raising. Movement is mediated in the phrase structure grammars by the SLASH feature; in CG extracted items and extraction cites are linked by functional composition.

Phrase structure grammars' SLASH and CG's type-raising both seem untidy. For example to obtain the correct SLASH percolation it is necessary to stipulate that SLASH is both a **HEAD** and a **FOOT** feature, and in HPSG we need to give a rather *ad hoc* constraint in order to prevent subject extraction (see POLL p.27). However there is an aspect of SLASH and type-raising which seems more fundamentally suspect. This is that in both cases we posit single-daughter syntactic local trees: in GPSG the effect of STM1 on a binary ID rule is to give a rule with only the head daughter (and a null category); in HPSG using the GIP with a binary rule means there is only one daughter; in CG type-raising necessarily applies to just one category. The problem with single-daughter local trees is that their existence is tantamount to saying 'if you have category X, you can call it category Y if you want'.

These arguments seem to favour an approach in which we borrow phrase structure grammars' **FOOT** features for anaphora, and CG's functional composition for movement; thus we avoid phrase structure grammar's SLASH and CG's type-raising. I would like to make a further suggestion however. We saw that in the same way FCRs and FCDs in GPSG became constraints on lexical entries in HPSG, so phrase structure grammars' head feature convention becomes reinterpreted as a generalization about lexical categories in CG. It seems to me that it may be possible to shift **FOOT** feature percolation information into lexical categories also. The result would be a CG which does not need to invoke type-raising in order to obtain **FOOT** feature percolation effects.

However CG as presented here would still uses type-raising for say topicalization[19]. I want to suggest something which constitutes a stand against all the formalisms. GPSG, HPSG, and CG all adhere to what I view as the 'myth of canonical form'. This idea has its origins with Transformational Grammar and deep structure. The claim is that canonical sentences are syntactically more basic than their topicalized and passivized etc. 'synonomous' counterparts. Thus in the phrase structure grammars we require metarules and gaps in order to obtain non-canonical forms, but not canonical ones; and in CG we use functional composition and type-raising to get topicalization. Bresnan (1978) argues convincingly that passive is a lexical process. It is my view that topicalization should also be viewed as a lexical process, i.e. there will be seperate lexical categories yielding canonical and topicalized forms.

In GPSG, HPSG, and CG we see a trend in which increasing amounts of information are put into the lexicon. I have suggested that **FOOT** feature percolation, such as that which we see linking relative pronouns to antecedents in pied-piped constructions, can also be captured in lexical categories. I have also suggested that 'movement', such as that we see in some relative clauses, could be made a lexical process. The mechanism facilitating unbounded movement could still be functional composition. The formalism emerging seems to be one in which the syntactic component has virtually disappeared.

---

[19] I should say that STOWE does not view type-raising as essential to capturing topicalization, though I'm not sure what alternatives he is thinking of.

Appendix A: Generalized Phrase Structure Grammar implementation

Lexicon

```
(setq Dict '((a (D (Rule DETNOUN) (NUM SG)))
             (bill (N (PER THIRD) (NUM SG) (CASE NOM) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1))
                   (N (PER THIRD) (NUM SG) (CASE ACC) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1)))
             (book (N (PER THIRD) (NUM SG) (CASE NOM) (GEN NEUT)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30))
                   (N (PER THIRD) (NUM SG) (CASE NOM) (GEN NEUT)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30)))
             (exists (V (Rule VP1) (VFORM FIN) (PER THIRD) (NUM SG)))
             (exist (V (Rule VP1) (VFORM FIN) (PER THIRD) (NUM PL)))
             (firm (N (PER THIRD) (NUM SG) (CASE NOM) (GEN NEUT)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30))
                   (N (PER THIRD) (NUM SG) (CASE ACC) (GEN NEUT)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30)))
             (for (P (Rule P38) (PFORM FOR)))
             (fred (N (PER THIRD) (NUM SG) (CASE NOM) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1))
                   (N (PER THIRD) (NUM SG) (CASE ACC) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1)))
             (john (N (PER THIRD) (NUM SG) (CASE NOM) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1))
                   (N (PER THIRD) (NUM SG) (CASE ACC) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1)))
             (loves (V (VFORM FIN) (PER THIRD) (NUM SG) (Rule VP2)))
             (man (N (PER THIRD) (NUM SG) (CASE NOM) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30))
                  (N (PER THIRD) (NUM SG) (CASE ACC) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30)))
             (mary (N (PER THIRD) (NUM SG) (CASE NOM) (GEN FEM)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1))
                   (N (PER THIRD) (NUM SG) (CASE ACC) (GEN FEM)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP1)))
             (men (N (PER THIRD) (NUM PL) (CASE NOM) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30))
                  (N (PER THIRD) (NUM PL) (CASE ACC) (GEN MASC)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30)))
             (of (P (Rule P38) (PFORM OF)))
             (person (N (PER THIRD) (NUM SG) (CASE NOM) (GEN COMM)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30))
                     (N (PER THIRD) (NUM SG) (CASE ACC) (GEN COMM)
                      (RPER -) (RNUM -) (RGEN -) (POSS -) (Rule NP30)))
```

```
(picture (N (PER THIRD) (NUM SG) (CASE NOM)
   (RPER -) (RNUM -) (RGEN -) (GEN NEUT) (POSS -) (Rule NP35))
         (N (PER THIRD) (NUM SG) (CASE ACC)
   (RPER -) (RNUM -) (RGEN -) (GEN NEUT) (POSS -) (Rule NP35)))
(the (D (Rule DETNOUN) (NUM SG))
     (D (Rule DETNOUN) (NUM PL)))
(which (N (PER THIRD) (NUM SG) (GEN NEUT) (CASE NOM) (RPER THIRD)
          (RNUM SG) (RGEN NEUT) (POSS -) (Rule NP1))
       (N (PER THIRD) (NUM SG) (GEN NEUT) (CASE ACC) (RPER THIRD)
          (RNUM SG) (RGEN NEUT) (POSS -) (Rule NP1))
       (N (PER THIRD) (NUM PL) (GEN NEUT) (CASE NOM) (RPER THIRD)
          (RNUM PL) (RGEN NEUT) (POSS -) (Rule NP1))
       (N (PER THIRD) (NUM PL) (GEN NEUT) (CASE ACC) (RPER THIRD)
          (RNUM SG) (RGEN NEUT) (POSS -) (Rule NP1)))
(who (N (PER FIRST) (NUM SG) (GEN MASC) (CASE NOM)
    (RPER FIRST) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM SG) (GEN MASC) (CASE NOM)
    (RPER SECOND) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM SG) (GEN MASC) (CASE NOM)
    (RPER THIRD) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER FIRST) (NUM PL) (GEN MASC) (CASE NOM)
            (RPER FIRST) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM PL) (GEN MASC) (CASE NOM)
    (RPER SECOND) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM PL) (GEN MASC) (CASE NOM)
    (RPER THIRD) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER FIRST) (NUM SG) (GEN FEM) (CASE NOM)
            (RPER FIRST) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM SG) (GEN FEM) (CASE NOM)
    (RPER SECOND) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM SG) (GEN FEM) (CASE NOM)
    (RPER THIRD) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER FIRST) (NUM PL) (GEN FEM) (CASE NOM)
            (RPER FIRST) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM PL) (GEN FEM) (CASE NOM)
    (RPER SECOND) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM PL) (GEN FEM) (CASE NOM)
    (RPER THIRD) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
     (N (PER FIRST) (NUM SG) (GEN COMM) (CASE NOM)
    (RPER FIRST) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM SG) (GEN COMM) (CASE NOM)
    (RPER SECOND) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM SG) (GEN COMM) (CASE NOM)
    (RPER THIRD) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
     (N (PER FIRST) (NUM PL) (GEN COMM) (CASE NOM)
            (RPER FIRST) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM PL) (GEN COMM) (CASE NOM)
    (RPER SECOND) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1))
     (N (PER THIRD) (NUM PL) (GEN COMM) (CASE NOM)
    (RPER THIRD) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1)))
(whom (N (PER FIRST) (NUM SG) (GEN MASC) (CASE ACC)
    (RPER FIRST) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
     (N (PER SECOND) (NUM SG) (GEN MASC) (CASE ACC)
```

```
                (RPER SECOND) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM SG) (GEN MASC) (CASE ACC)
                (RPER THIRD) (RNUM SG) (RGEN MASC) (POSS -) (Rule NP1))
                 (N (PER FIRST) (NUM PL) (GEN MASC) (CASE ACC)
                        (RPER FIRST) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
                 (N (PER SECOND) (NUM PL) (GEN MASC) (CASE ACC)
                (RPER SECOND) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM PL) (GEN MASC) (CASE ACC)
                (RPER THIRD) (RNUM PL) (RGEN MASC) (POSS -) (Rule NP1))
                 (N (PER FIRST) (NUM SG) (GEN FEM) (CASE ACC)
                        (RPER FIRST) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER SECOND) (NUM SG) (GEN FEM) (CASE ACC)
                (RPER SECOND) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM SG) (GEN FEM) (CASE ACC)
                (RPER THIRD) (RNUM SG) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER FIRST) (NUM PL) (GEN FEM) (CASE ACC)
                        (RPER FIRST) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER SECOND) (NUM PL) (GEN FEM) (CASE ACC)
                (RPER SECOND) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM PL) (GEN FEM) (CASE ACC)
                (RPER THIRD) (RNUM PL) (RGEN FEM) (POSS -) (Rule NP1))
                 (N (PER FIRST) (NUM SG) (GEN COMM) (CASE ACC)
                (RPER FIRST) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
                 (N (PER SECOND) (NUM SG) (GEN COMM) (CASE ACC)
                (RPER SECOND) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM SG) (GEN COMM) (CASE ACC)
                (RPER THIRD) (RNUM SG) (RGEN COMM) (POSS -) (Rule NP1))
                 (N (PER FIRST) (NUM PL) (GEN COMM) (CASE ACC)
                        (RPER FIRST) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1))
                 (N (PER SECOND) (NUM PL) (GEN COMM) (CASE ACC)
                (RPER SECOND) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1))
                 (N (PER THIRD) (NUM PL) (GEN COMM) (CASE ACC)
                (RPER THIRD) (RNUM PL) (RGEN COMM) (POSS -) (Rule NP1)))
            (whose (N (RPER THIRD) (RNUM SG) (RGEN MASC) (POSS +)
                                                          (Rule NP1))
                   (N (RPER THIRD) (RNUM SG) (RGEN NEUT) (POSS +)
                                                          (Rule NP1))
                   (N (RPER THIRD) (RNUM PL) (RGEN MASC) (POSS +)
                                                          (Rule NP1))
                   (N (RPER THIRD) (RNUM PL) (RGEN NEUT) (POSS +)
                                                          (Rule NP1)))
            (works (V (Rule VP999) (SUBJ -) (VFORM FIN) (PER THIRD)
                                                        (NUM SG)))))

(setq Redun nil)
```

Syntactic rules

```
(ReadGram nil)

BaseCat N
BaseCat V
BaseCat P
BaseCat D

PropFeat V {VFORM PER NUM}
PropFeat N {PER NUM GEN CASE POSS}
PropFeat P {PFORM}
PropFeat D {NUM}

FeatureRangeVariable x {}
FeatureRangeVariable y {}
FeatureRangeVariable z {}
FeatureRangeVariable c {}
FeatureRangeVariable u {}
FeatureRangeVariable v {}
FeatureRangeVariable w {}

Variable Z {V|| V| P||}
Variable W {V|| V| P||}
Variable X {}
Variable Y {P|| N||L(POSS -)]}
Variable Top {V|| N|| P|| V| N| Z/Y}

Feature VFORM {FIN BAS PRP PSP PAS INF}
Feature POSS {+ -}
Feature PER {FIRST SECOND THIRD}
Feature NUM {SG PL}
Feature GEN {MASC FEM COMM NEUT}
Feature CASE {NOM ACC}
Feature RPER {FIRST SECOND THIRD -}
Feature RNUM {SG PL -}
Feature RGEN {MASC FEM COMM NEUT -}
Feature PFORM {FOR OF}

<NP1: N||L(RPER u) (RNUM v) (RGEN w)] ->
                        NL(RPER u) (RNUM v) (RGEN w)]>
<NP30: N|L(RPER u) (RNUM v) (RGEN w)] ->
                        NL(RPER u) (RNUM v) (RGEN w)]>
<NP35: N|L(RPER u) (RNUM v) (RGEN w)] -> N
                    P||L(PFORM OF) (RPER u) (RNUM v) (RGEN w)]>
<VP1: V| -> V>
<VP2: V| -> V N||L(CASE ACC) (POSS -)]>
<VP999: V| -> V P||L(PFORM FOR)]>
<P38: P||L(RPER u) (RNUM v) (RGEN w)] -> P
                        N||L(CASE ACC) (RPER u) (RNUM v) (RGEN w) (POSS -)]>

<DETNOUN: N||L(RPER u) (RNUM v) (RGEN w)] ->
        DL(NUM y)] N|L(NUM y) (RPER u) (RNUM v) (RGEN w)]>
```

```
<SBJPRD: V||L(RPER u) (RNUM v) (RGEN w)] ->
N||L(PER x) (NUM y) (CASE NOM) (RPER u) (RNUM v) (RGEN w) (POSS -)]
                                                    V|L(PER x) (NUM y)]>
<LFTEXT: V||L(RPER u) (RNUM v) (RGEN w)] -> YL(RPER u) (RNUM v) (RGEN w)]
                                                                  V||/Y>
<RELANT: N|L(RPER -) (RNUM -) (RGEN -)] -> N|L(PER x) (NUM y) (GEN z)]
                                          V||L(RPER x) (RNUM y) (RGEN z)]>
<POSSNP: N||L(RPER u) (RNUM v) (RGEN w)] ->
                        N||L+POSS (RPER u) (RNUM v) (RGEN w)] N|>

{SLTERM: <Z -> ... X Y ...>
                  =>
         <Z/Y -> ... X ...>}
{SLPROP: <W -> ... Z ...>
              =>
         <W/Y -> ... Z/Y ...>}

EndGram

(NewGram)
```

Sample Output

```
Script started on Fri Sep 27 04:20:56 1985
% ~jdp/gpsgp

Parse as t: l

file? RCG

is /mnt/glyn/MCHARTGPSG the right directory? y

Parse as t: l

file? RCD

is /mnt/glyn/MCHARTGPSG the right directory? y

Parse as t: the man who works for john exists

1420 msec CPU, 4000 msec clock, 0 conses.
179 active edges, 95 inactive edges, 79 redundancies.
1 Parse

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)(CASE NOM)
                                                              (POSS -)
             (D the(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                    (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                    (CASE NOM)(POSS -)
```

```
                    (N man(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                               (GEN MASC)(CASE NOM)(NUM SG)(PER THIRD)))
               (V|| (RGEN MASC)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                    (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                (GEN MASC)(CASE NOM)(POSS -)
                         (N who(Rule NP1)(POSS -)(RGEN MASC)(RNUM SG)
                                (RPER THIRD)(CASE NOM)(GEN MASC)
                                (NUM SG)(PER THIRD)))
                    (V| (VFORM FIN)(PER THIRD)(NUM SG)
                        (V works(NUM SG)(PER THIRD)(VFORM FIN)(SUBJ -)
                                                   (Rule VP999))
                        (P|| (RGEN -)(RNUM -)(RPER -)(PFORM FOR)
                             (P for(PFORM FOR)(Rule P38))
                             (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                                  (NUM SG)(GEN MASC)(CASE ACC)
                                                  (POSS -)
                                  (N john(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN MASC)(CASE ACC)
                                                  (NUM SG)(PER THIRD)))))))))
          (V| (VFORM FIN)(PER THIRD)(NUM SG)
              (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: the men who works for john exist

1420 msec CPU, 4000 msec clock, 0 conses.
163 active edges, 86 inactive edges, 73 redundancies.
No Parses


Parse as t: the man whom bill works for exists

1420 msec CPU, 4000 msec clock, 0 conses.
99 active edges, 92 inactive edges, 45 redundancies.
1 Parse

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)(CASE NOM)
                                                          (POSS -)
             (D the(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                     (N man(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                               (GEN MASC)(CASE NOM)(NUM SG)(PER THIRD)))
                 (V|| (RGEN MASC)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                      (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                (GEN MASC)(CASE ACC)(POSS -)
                           (N whom(Rule NP1)(POSS -)(RGEN MASC)(RNUM SG)
                                (RPER THIRD)(CASE ACC)(GEN MASC)
                                (NUM SG)(PER THIRD)))
```

```
                    (V||/N|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                             (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                                  (NUM SG)(GEN MASC)(CASE NOM)
                                                  (POSS -)
                                  (N bill(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN MASC)(CASE NOM)
                                                  (NUM SG)(PER THIRD)))
                             (V|/N|| (VFORM FIN)(PER THIRD)(NUM SG)
                                     (V works(NUM SG)(PER THIRD)(VFORM FIN)
                                                  (SUBJ -)(Rule VP999))
                                     (P||/N|| (RGEN . w)(RNUM . v)(RPER . u)
                                                  (PFORM FOR)
                                              (P for(PFORM FOR)
                                                  (Rule P38)))))))).
        (V| (VFORM FIN)(PER THIRD)(NUM SG)
            (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: the firm whom bill works for exists

1420 msec CPU, 4000 msec clock, 0 conses.
83 active edges, 83 inactive edges, 39 redundancies.
No Parses


Parse as t: the man whose book mary loves exists

1420 msec CPU, 4000 msec clock, 0 conses.
139 active edges, 70 inactive edges, 51 redundancies.
2 Parses

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)(CASE NOM)
                                                          (POSS -)
             (D the(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                     (N man(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                                 (GEN MASC)(CASE NOM)(NUM SG)(PER THIRD)))
                 (V|| (RGEN MASC)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                      (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                 (GEN NEUT)(CASE NOM)(POSS -)
                           (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(POSS +)
                                (N whose(Rule NP1)(POSS +)(RGEN MASC)(RNUM SG)
                                                  (RPER THIRD)))
                           (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)
                                                  (GEN NEUT)(CASE NOM)(POSS -)
                               (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN NEUT)(CASE NOM)(NUM SG)
                                                  (PER THIRD))))
```

```
                    (V||/N|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                              (NUM SG)
                             (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                          (NUM SG)(GEN FEM)(CASE NOM)
                                          (POSS -)
                                  (N mary(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                          (RPER -)(GEN FEM)(CASE NOM)
                                          (NUM SG)(PER THIRD)))
                             (V|/N|| (VFORM FIN)(PER THIRD)(NUM SG)
                                     (V loves(Rule VP2)(NUM SG)(PER THIRD)
                                          (VFORM FIN)))))))
        (V| (VFORM FIN)(PER THIRD)(NUM SG)
            (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))
(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)(CASE NOM)
                                                              (POSS -)
             (D the(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                   (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN MASC)
                                                   (CASE NOM)(POSS -)
                     (N man(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                            (GEN MASC)(CASE NOM)(NUM SG)(PER THIRD)))
                 (V|| (RGEN MASC)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                              (NUM SG)
                      (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                   (GEN NEUT)(CASE NOM)(POSS -)
                           (N|| (RGEN MASC)(RNUM SG)(RPER THIRD)(POSS +)
                                (N whose(Rule NP1)(POSS +)(RGEN MASC)(RNUM SG)
                                                   (RPER THIRD)))
                           (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)
                                   (GEN NEUT)(CASE NOM)(POSS -)
                               (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)
                                   (RPER -)(GEN NEUT)(CASE NOM)(NUM SG)
                                   (PER THIRD))))
                      (V||/N|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                              (NUM SG)
                               (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                          (NUM SG)(GEN FEM)(CASE NOM)
                                          (POSS -)
                                    (N mary(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                          (RPER -)(GEN FEM)(CASE NOM)
                                          (NUM SG)(PER THIRD)))
                               (V|/N|| (VFORM FIN)(PER THIRD)(NUM SG)
                                       (V loves(Rule VP2)(NUM SG)(PER THIRD)
                                          (VFORM FIN)))))))
        (V| (VFORM FIN)(PER THIRD)(NUM SG)
            (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: the men whose book mary loves exist

1420 msec CPU, 4000 msec clock, 0 conses.
139 active edges, 70 inactive edges, 51 redundancies.
```

```
2 Parses

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM PL)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)(CASE NOM)
                                                          (POSS -)
             (D the(NUM PL)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                     (N men(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                                 (GEN MASC)(CASE NOM)(NUM PL)(PER THIRD)))
                 (V|| (RGEN MASC)(RNUM PL)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                      (N|| (RGEN MASC)(RNUM PL)(RPER THIRD)(PER THIRD)(NUM SG)
                                   (GEN NEUT)(CASE NOM)(POSS -)
                           (N|| (RGEN MASC)(RNUM PL)(RPER THIRD)(POSS +)
                                (N whose(Rule NP1)(POSS +)(RGEN MASC)(RNUM PL)
                                                  (RPER THIRD)))
                           (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)
                                   (GEN NEUT)(CASE NOM)(POSS -)
                               (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)
                                   (RPER -)(GEN NEUT)(CASE NOM)(NUM SG)
                                   (PER THIRD))))
                      (V||/N|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                               (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                                  (NUM SG)(GEN FEM)(CASE NOM)
                                                  (POSS -)
                                    (N mary(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN FEM)(CASE NOM)
                                                  (NUM SG)(PER THIRD)))
                               (V|/N|| (VFORM FIN)(PER THIRD)(NUM SG)
                                       (V loves(Rule VP2)(NUM SG)(PER THIRD)
                                                  (VFORM FIN)))))))
        (V| (VFORM FIN)(PER THIRD)(NUM PL)
            (V exist(NUM PL)(PER THIRD)(VFORM FIN)(Rule VP1)))))
(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM PL)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)(CASE NOM)
                                                          (POSS -)
             (D the(NUM PL)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM PL)(GEN MASC)
                                                  (CASE NOM)(POSS -)
                     (N men(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                                 (GEN MASC)(CASE NOM)(NUM PL)(PER THIRD)))
                 (V|| (RGEN MASC)(RNUM PL)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                          (NUM SG)
                      (N|| (RGEN MASC)(RNUM PL)(RPER THIRD)(PER THIRD)(NUM SG)
                                   (GEN NEUT)(CASE NOM)(POSS -)
                           (N|| (RGEN MASC)(RNUM PL)(RPER THIRD)(POSS +)
                                (N whose(Rule NP1)(POSS +)(RGEN MASC)(RNUM PL)
```

```
                                                    (RPER THIRD)))
                                  (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)
                                              (GEN NEUT)(CASE NOM)(POSS -)
                                      (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN NEUT)(CASE NOM)
                                                  (NUM SG)(PER THIRD))))
                              (V||/N|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                                  (NUM SG)
                                      (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                                  (NUM SG)(GEN FEM)(CASE NOM)
                                                  (POSS -)
                                          (N mary(Rule NP1)(POSS -)(RGEN -)(RNUM -)
                                                  (RPER -)(GEN FEM)(CASE NOM)
                                                  (NUM SG)(PER THIRD)))
                                      (V|/N|| (VFORM FIN)(PER THIRD)(NUM SG)
                                              (V loves(Rule VP2)(NUM SG)(PER THIRD)
                                                    (VFORM FIN)))))))
                (V| (VFORM FIN)(PER THIRD)(NUM PL)
                    (V exist(NUM PL)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: the person for whom fred works exists

1420 msec CPU, 4000 msec clock, 0 conses.
100 active edges, 107 inactive edges, 45 redundancies.
1 Parse

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN COMM)(CASE NOM)
                                                              (POSS -)
             (D the(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN COMM)
                                                    (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN COMM)
                                                    (CASE NOM)(POSS -)
                     (N person(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                               (GEN COMM)(CASE NOM)(NUM SG)(PER THIRD)))
                 (V|| (RGEN COMM)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                              (NUM SG)
                      (P|| (RGEN COMM)(RNUM SG)(RPER THIRD)(PFORM FOR)
                           (P for(PFORM FOR)(Rule P38))
                           (N|| (RGEN COMM)(RNUM SG)(RPER THIRD)(PER THIRD)
                                                (NUM SG)(GEN COMM)(CASE ACC)
                                                (POSS -)
                                (N whom(Rule NP1)(POSS -)(RGEN COMM)(RNUM SG)
                                                (RPER THIRD)(CASE ACC)
                                                (GEN COMM)(NUM SG)
                                                (PER THIRD))))
                      (V||/P|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)
                                                              (NUM SG)
                               (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)
                                                (NUM SG)(GEN MASC)(CASE NOM)
                                                (POSS -)
                                   (N fred(Rule NP1)(POSS -)(RGEN -)(RNUM -)
```

```
                                                (RPER -)(GEN MASC)(CASE NOM)
                                                (NUM SG)(PER THIRD)))
                                    (V|/P|| (VFORM FIN)(PER THIRD)(NUM SG)
                                            (V works(NUM SG)(PER THIRD)(VFORM FIN)
                                                (SUBJ -)(Rule VP999)))))))
            (V| (VFORM FIN)(PER THIRD)(NUM SG)
                (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: the person for who fred works exists

1420 msec CPU, 4000 msec clock, 0 conses.
41 active edges, 58 inactive edges, 21 redundancies.
No Parses


Parse as t: a book a picture of which exists exists

1420 msec CPU, 4000 msec clock, 0 conses.
101 active edges, 65 inactive edges, 33 redundancies.
4 Parses

(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)(CASE NOM)
                                                        (POSS -)
             (D a(NUM SG)(Rule DETNOUN))
             (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                (CASE NOM)(POSS -)
                 (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                (CASE NOM)(POSS -)
                     (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                                (GEN NEUT)(CASE NOM)(NUM SG)(PER THIRD)))
                 (V|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                        (NUM SG)
                      (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                    (GEN NEUT)(CASE NOM)(POSS -)
                           (D a(NUM SG)(Rule DETNOUN))
                           (N| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)
                                (NUM SG)(GEN NEUT)(CASE NOM)(POSS -)
                               (N picture(Rule NP35)(POSS -)(GEN NEUT)(RGEN -)
                                                (RNUM -)(RPER -)(CASE NOM)
                                                (NUM SG)(PER THIRD))
                               (P|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                                        (PFORM OF)
                                    (P of(PFORM OF)(Rule P38))
                                    (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                                (PER THIRD)(NUM SG)(GEN NEUT)
                                                (CASE ACC)(POSS -)
                                         (N which(Rule NP1)(POSS -)(RGEN NEUT)
                                                        (RNUM SG)(RPER THIRD)
                                                        (CASE ACC)(GEN NEUT)
                                                        (NUM SG)
                                                        (PER THIRD))))))
                      (V| (VFORM FIN)(PER THIRD)(NUM SG)
```

```
                        (V exists(NUM SG)(PER THIRD)(VFORM FIN)
                                                    (Rule VP1))))))
        (V| (VFORM FIN)(PER THIRD)(NUM SG)
            (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))
(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)(CASE NOM)
                                                        (POSS -)
            (D a(NUM SG)(Rule DETNOUN))
            (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                    (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                            (GEN NEUT)(CASE NOM)(NUM SG)(PER THIRD)))
                (V|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                        (NUM SG)
                    (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                (GEN NEUT)(CASE NOM)(POSS -)
                        (D a(NUM SG)(Rule DETNOUN))
                        (N| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)
                            (NUM SG)(GEN NEUT)(CASE NOM)(POSS -)
                            (N picture(Rule NP35)(POSS -)(GEN NEUT)(RGEN -)
                                        (RNUM -)(RPER -)(CASE NOM)
                                        (NUM SG)(PER THIRD))
                            (P|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                                        (PFORM OF)
                                (P of(PFORM OF)(Rule P38))
                                (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                        (PER THIRD)(NUM SG)(GEN NEUT)
                                        (CASE ACC)(POSS -)
                                    (N which(Rule NP1)(POSS -)(RGEN NEUT)
                                                (RNUM SG)(RPER THIRD)
                                                (CASE ACC)(GEN NEUT)
                                                (NUM SG)
                                                (PER THIRD))))))
                        (V| (VFORM FIN)(PER THIRD)(NUM SG)
                            (V exists(NUM SG)(PER THIRD)(VFORM FIN)
                                                    (Rule VP1))))))
        (V| (VFORM FIN)(PER THIRD)(NUM SG)
            (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))
(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)(CASE NOM)
                                                        (POSS -)
            (D a(NUM SG)(Rule DETNOUN))
            (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                    (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                            (GEN NEUT)(CASE NOM)(NUM SG)(PER THIRD)))
                (V|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
                                                        (NUM SG)
                    (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
```

```
                                        (GEN NEUT)(CASE NOM)(POSS -)
                                (D a(NUM SG)(Rule DETNOUN))
                                (N| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)
                                    (NUM SG)(GEN NEUT)(CASE NOM)(POSS -)
                                    (N picture(Rule NP35)(POSS -)(GEN NEUT)(RGEN -)
                                                (RNUM -)(RPER -)(CASE NOM)
                                                (NUM SG)(PER THIRD))
                                    (P|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                                                (PFORM OF)
                                        (P of(PFORM OF)(Rule P38))
                                        (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD),
                                                (PER THIRD)(NUM PL)(GEN NEUT)
                                                (CASE ACC)(POSS -)
                                            (N which(Rule NP1)(POSS -)(RGEN NEUT)
                                                        (RNUM SG)(RPER THIRD)
                                                        (CASE ACC)(GEN NEUT)
                                                        (NUM PL)
                                                        (PER THIRD))))))
                            (V| (VFORM FIN)(PER THIRD)(NUM SG)
                                (V exists(NUM SG)(PER THIRD)(VFORM FIN)
                                                        (Rule VP1))))))
                    (V| (VFORM FIN)(PER THIRD)(NUM SG)
                        (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))
(t (V|| (RGEN -)(RNUM -)(RPER -)(VFORM FIN)(PER THIRD)(NUM SG)
        (N|| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)(CASE NOM)
                                                            (POSS -)
            (D a(NUM SG)(Rule DETNOUN))
            (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                (N| (RGEN -)(RNUM -)(RPER -)(PER THIRD)(NUM SG)(GEN NEUT)
                                                    (CASE NOM)(POSS -)
                    (N book(Rule NP30)(POSS -)(RGEN -)(RNUM -)(RPER -)
                            (GEN NEUT)(CASE NOM)(NUM SG)(PER THIRD)))
                (V|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(VFORM FIN)(PER THIRD)
(NUM SG)
                    (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)(NUM SG)
                                (GEN NEUT)(CASE NOM)(POSS -)
                        (D a(NUM SG)(Rule DETNOUN))
                        (N| (RGEN NEUT)(RNUM SG)(RPER THIRD)(PER THIRD)
                            (NUM SG)(GEN NEUT)(CASE NOM)(POSS -)
                            (N picture(Rule NP35)(POSS -)(GEN NEUT)(RGEN -)
                                        (RNUM -)(RPER -)(CASE NOM)
                                        (NUM SG)(PER THIRD))
                            (P|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                                        (PFORM OF)
                                (P of(PFORM OF)(Rule P38))
                                (N|| (RGEN NEUT)(RNUM SG)(RPER THIRD)
                                        (PER THIRD)(NUM PL)(GEN NEUT)
                                        (CASE ACC)(POSS -)
                                    (N which(Rule NP1)(POSS -)(RGEN NEUT)
                                                (RNUM SG)(RPER THIRD)
                                                (CASE ACC)(GEN NEUT)
                                                (NUM PL)
```

```
                                                        (PER THIRD))))))
                        (V| (VFORM FIN)(PER THIRD)(NUM SG)
                            (V exists(NUM SG)(PER THIRD)(VFORM FIN)
                                                        (Rule VP1))))))
            (V| (VFORM FIN)(PER THIRD)(NUM SG)
                (V exists(NUM SG)(PER THIRD)(VFORM FIN)(Rule VP1)))))

Parse as t: a man a picture of which exists exists

1420 msec CPU, 4000 msec clock, 0 conses.
56 active edges, 44 inactive edges, 15 redundancies.
No Parses


Parse as t: q

Finished - type 'cont' after the '.' if you want to go on
% ^Z
script done on Fri Sep 27 04:32:09 1985
```

Appendix B: Combinatory Grammar implementation

Lexicon

```
a:      [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr:num = sg,
        cat:forming:slash = no,
        cat:forming:cat:maj = np,
        cat:forming:cat:agr:per = third,
        cat:forming:cat:agr:num = sg,
        cat:forming:cat:agr:gen = cat:eating:cat:agr:gen].

bill:   [slash = no,
        cat:maj = np,
        cat:agr:per = third,
        cat:agr:num = sg,
        cat:agr:gen = masc].

book:   [slash = no,
        cat:maj = n,
        cat:agr:num = sg,
        cat:agr:gen = neut].

exists: [slash = backwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = np,
        cat:eating:cat:agr:per = third,
        cat:eating:cat:agr:num = sg,
        cat:eating:cat:case = nom,
        cat:forming:slash = no,
        cat:forming:cat:maj = s,
        cat:forming:cat:vform = fin].

% plural "exist"

exist:  [slash = backwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = np,
        cat:eating:cat:agr:num = pl,
        cat:eating:cat:case = nom,
        cat:forming:slash = no,
        cat:forming:cat:maj = s,
        cat:forming:cat:vform = fin].
```

```
firm:   [slash = no,
         cat:maj = n,
         cat:agr:num = sg,
         cat:agr:gen = neut].

for:    [slash = forwards,
         cat:eating:slash = no,
         cat:eating:cat:maj = np,
         cat:eating:cat:case = acc,
         cat:forming:slash = no,
         cat:forming:cat:maj = pp,
         cat:forming:cat:pform = for].

fred:   [slash = no,
         cat:maj = np,
         cat:agr:per = third,
         cat:agr:num = sg,
         cat:agr:gen = masc].

john:   [slash = no,
         cat:maj = np,
         cat:agr:per = third,
         cat:agr:num = sg,
         cat:agr:gen = masc].

loves:  [slash = forwards,
         cat:eating:slash = no,
         cat:eating:cat:maj = np,
         cat:eating:cat:case = acc,
         cat:forming:slash = backwards,
         cat:forming:cat:eating:slash = no,
         cat:forming:cat:eating:cat:maj = np,
         cat:forming:cat:eating:cat:agr:per = third,
         cat:forming:cat:eating:cat:agr:num = sg,
         cat:forming:cat:eating:cat:case = nom,
         cat:forming:cat:forming:slash = no,
         cat:forming:cat:forming:cat:maj = s,
         cat:forming:cat:forming:cat:vform = fin].

man:    [slash = no,
         cat:maj = n,
         cat:agr:num = sg,
         cat:agr:gen = masc].

men:    [slash = no,
         cat:maj = n,
         cat:agr:num = pl,
         cat:agr:gen = masc].

mary:   [slash = no,
         cat:maj = np,
         cat:agr:per = third,
         cat:agr:num = sg,
```

```
        cat:agr:gen = fem].

of:     [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = np,
        cat:eating:cat:case = acc,
        cat:forming:slash = no,
        cat:forming:cat:maj = pp,
        cat:forming:cat:pform = of].

person: [slash = no,
        cat:maj = n,
        cat:agr:num = sg,
        cat:agr:gen = comm].

picture:[slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = pp,
        cat:eating:cat:pform = of,
        cat:forming:slash = no,
        cat:forming:cat:maj = n,
        cat:forming:cat:agr:num = sg,
        cat:forming:cat:agr:gen = neut].

the:    [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:forming:slash = no,
        cat:forming:cat:maj = np,
        cat:forming:cat:agr:per = third,
        cat:forming:cat:agr:num = cat:eating:cat:agr:num,
        cat:forming:cat:agr:gen = cat:eating:cat:agr:gen].

% A 'subject' "which" and an 'object' which

which:  [slash = forwards,
        cat:eating:slash = backwards,
        cat:eating:cat:eating:slash = no,
        cat:eating:cat:eating:cat:maj = np,
        cat:eating:cat:eating:cat:agr:gen = neut,
        cat:eating:cat:forming:slash = no,
        cat:eating:cat:forming:cat:maj = s,
        cat:eating:cat:forming:cat:vform = fin,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
        cat:forming:cat:forming = cat:forming:cat:eating].

which:  [slash = forwards,
        cat:eating:slash = forwards,
        cat:eating:cat:eating:slash = no,
        cat:eating:cat:eating:cat:maj = np,
```

```
        cat:eating:cat:eating:cat:agr:gen = neut,
        cat:eating:cat:forming:slash = no,
        cat:eating:cat:forming:cat:maj = s,
        cat:eating:cat:forming:cat:vform = fin,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
        cat:forming:cat:forming = cat:forming:cat:eating].

% One "who" for each non-neuter gender

who:    [slash = forwards,
        cat:eating:slash = backwards,
        cat:eating:cat:eating:slash = no,
        cat:eating:cat:eating:cat:maj = np,
        cat:eating:cat:eating:cat:agr:gen = masc,
        cat:eating:cat:forming:slash = no,
        cat:eating:cat:forming:cat:maj = s,
        cat:eating:cat:forming:cat:vform = fin,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
        cat:forming:cat:forming = cat:forming:cat:eating].

who:    [slash = forwards,
        cat:eating:slash = backwards,
        cat:eating:cat:eating:slash = no,
        cat:eating:cat:eating:cat:maj = np,
        cat:eating:cat:eating:cat:agr:gen = fem,
        cat:eating:cat:forming:slash = no,
        cat:eating:cat:forming:cat:maj = s,
        cat:eating:cat:forming:cat:vform = fin,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
        cat:forming:cat:forming = cat:forming:cat:eating].

who:    [slash = forwards,
        cat:eating:slash = backwards,
        cat:eating:cat:eating:slash = no,
        cat:eating:cat:eating:cat:maj = np,
        cat:eating:cat:eating:cat:agr:gen = comm,
        cat:eating:cat:forming:slash = no,
        cat:eating:cat:forming:cat:maj = s,
        cat:eating:cat:forming:cat:vform = fin,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
        cat:forming:cat:forming = cat:forming:cat:eating].
```

```
% One "whom" for each non-neuter gender

whom:   [slash = forwards,
         cat:eating:slash = forwards,
         cat:eating:cat:eating:slash = no,
         cat:eating:cat:eating:cat:maj = np,
         cat:eating:cat:eating:cat:agr:gen = masc,
         cat:eating:cat:forming:slash = no,
         cat:eating:cat:forming:cat:maj = s,
         cat:eating:cat:forming:cat:vform = fin,
         cat:forming:slash = backwards,
         cat:forming:cat:eating:slash = no,
         cat:forming:cat:eating:cat:maj = np,
         cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
         cat:forming:cat:forming = cat:forming:cat:eating].

whom:   [slash = forwards,
         cat:eating:slash = forwards,
         cat:eating:cat:eating:slash = no,
         cat:eating:cat:eating:cat:maj = np,
         cat:eating:cat:eating:cat:agr:gen = fem,
         cat:eating:cat:forming:slash = no,
         cat:eating:cat:forming:cat:maj = s,
         cat:eating:cat:forming:cat:vform = fin,
         cat:forming:slash = backwards,
         cat:forming:cat:eating:slash = no,
         cat:forming:cat:eating:cat:maj = np,
         cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
         cat:forming:cat:forming = cat:forming:cat:eating].

whom:   [slash = forwards,
         cat:eating:slash = forwards,
         cat:eating:cat:eating:slash = no,
         cat:eating:cat:eating:cat:maj = np,
         cat:eating:cat:eating:cat:agr:gen = comm,
         cat:eating:cat:forming:slash = no,
         cat:eating:cat:forming:cat:maj = s,
         cat:eating:cat:forming:cat:vform = fin,
         cat:forming:slash = backwards,
         cat:forming:cat:eating:slash = no,
         cat:forming:cat:eating:cat:maj = np,
         cat:forming:cat:eating:cat:agr = cat:eating:cat:eating:cat:agr,
         cat:forming:cat:forming = cat:forming:cat:eating].

% One 'subject' "whose" for each non-neuter gender

whose:  [slash = forwards,
         cat:eating:slash = no,
         cat:eating:cat:maj = n,
         cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
         cat:forming:slash = forwards,
         cat:forming:cat:eating:slash = backwards,
         cat:forming:cat:eating:cat:eating:slash = no,
```

```
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = masc,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

whose:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
        cat:forming:slash = forwards,
        cat:forming:cat:eating:slash = backwards,
        cat:forming:cat:eating:cat:eating:slash = no,
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = fem,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

whose:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
        cat:forming:slash = forwards,
        cat:forming:cat:eating:slash = backwards,
        cat:forming:cat:eating:cat:eating:slash = no,
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = comm,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

% One 'object' "whose" for each non-neuter gender

whose:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
```

```
        cat:forming:slash = forwards,
        cat:forming:cat:eating:slash = forwards,
        cat:forming:cat:eating:cat:eating:slash = no,
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = masc,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

whose:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
        cat:forming:slash = forwards,
        cat:forming:cat:eating:slash = forwards,
        cat:forming:cat:eating:cat:eating:slash = no,
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = fem,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

whose:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = n,
        cat:eating:cat:agr = cat:forming:cat:eating:cat:eating:cat:agr,
        cat:forming:slash = forwards,
        cat:forming:cat:eating:slash = forwards,
        cat:forming:cat:eating:cat:eating:slash = no,
        cat:forming:cat:eating:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:forming:slash = no,
        cat:forming:cat:eating:cat:forming:cat:maj = s,
        cat:forming:cat:eating:cat:forming:cat:vform = fin,
        cat:forming:cat:forming:slash = backwards,
        cat:forming:cat:forming:cat:eating:slash = no,
        cat:forming:cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:forming:cat:eating:cat:agr:gen = comm,
        cat:forming:cat:forming:cat:eating = cat:forming:cat:forming:cat:forming].

works:  [slash = forwards,
        cat:eating:slash = no,
        cat:eating:cat:maj = pp,
```

```
        cat:eating:cat:pform = for,
        cat:forming:slash = backwards,
        cat:forming:cat:eating:slash = no,
        cat:forming:cat:eating:cat:maj = np,
        cat:forming:cat:eating:cat:agr:per = third,
        cat:forming:cat:eating:cat:agr:num = sg,
        cat:forming:cat:eating:cat:case = nom,
        cat:forming:cat:forming:slash = no,
        cat:forming:cat:forming:cat:maj = s,
        cat:forming:cat:forming:cat:vform = fin].
```

Syntactic rules

```
% forward application

m --> f, a,     {f:slash = forwards,
                 a = f:cat:eating,
                 m = f:cat:forming}.

% backward application

m --> a, f,     {f:slash = backwards,
                 a = f:cat:eating,
                 m = f:cat:forming}.

% forward composition

m --> c, d,     {c:slash = forwards, d:slash = forwards,
                 c:cat:eating = d:cat:forming,
                 m:slash = forwards,
                 m:cat:eating = d:cat:eating,
                 m:cat:forming = c:cat:forming}.

% subject type-raising

m --> d,        {d:slash = no,
                 d:cat:maj = np,
                 d:cat:case = nom,
                 m:slash = forwards,
                 m:cat:eating:slash = backwards,
                 d = m:cat:eating:cat:eating,
                 m:cat:eating:cat:forming:slash = no,
                 m:cat:eating:cat:forming:cat:maj = s,
                 m:cat:eating:cat:forming:cat:vform = fin,
                 m:cat:forming = m:cat:eating:cat:forming}.

% object relative pronoun type raising

m --> d,        {d:slash = forwards,
                 d:cat:eating:slash = forwards,
                 d:cat:eating:cat:eating:slash = no,
                 d:cat:eating:cat:eating:cat:maj = np,
```

```
                d:cat:eating:cat:forming:slash = no,
                d:cat:eating:cat:forming:cat:maj = s,
                d:cat:eating:cat:forming:cat:vform = fin,
                d:cat:forming:slash = backwards,
                d:cat:forming:cat:eating = d:cat:eating:cat:eating,
                d:cat:forming:cat:forming = d:cat:eating:cat:eating,
                m:slash = backwards,
                m:cat:eating:slash = forwards,
                m:cat:eating:cat:eating = d:cat:eating:cat:eating,
                m:cat:forming:slash = forwards,
                m:cat:forming:cat:eating:slash = forwards,
                m:cat:forming:cat:eating:cat:eating =
                                              m:cat:eating:cat:forming,
                m:cat:forming:cat:eating:cat:forming =
                                              d:cat:eating:cat:forming,
                m:cat:forming:cat:forming:slash = backwards,
                m:cat:forming:cat:forming:cat:eating:slash = no,
                m:cat:forming:cat:forming:cat:eating:cat:maj = np,
                m:cat:forming:cat:forming:cat:eating:cat:agr =
                                    m:cat:eating:cat:eating:cat:agr,
                m:cat:forming:cat:forming:cat:forming =
                                    m:cat:forming:cat:forming:cat:eating}.

% object-subject relative pronoun type raising

m --> d,        {d:slash = forwards,
                d:cat:eating:slash = forwards,
                d:cat:eating:cat:eating:slash = no,
                d:cat:eating:cat:eating:cat:maj = np,
                d:cat:eating:cat:forming:slash = no,
                d:cat:eating:cat:forming:cat:maj = s,
                d:cat:eating:cat:forming:cat:vform = fin,
                d:cat:forming:slash = backwards,
                d:cat:forming:cat:eating = d:cat:eating:cat:eating,
                d:cat:forming:cat:forming = d:cat:eating:cat:eating,
                m:slash = backwards,
                m:cat:eating:slash = forwards,
                m:cat:eating:cat:eating = d:cat:eating:cat:eating,
                m:cat:eating:cat:forming:slash = no,
                m:cat:eating:cat:forming:cat:maj = np,
                m:cat:eating:cat:forming:cat:case = nom,
                m:cat:forming:slash = forwards,
                m:cat:forming:cat:eating:slash = backwards,
                m:cat:forming:cat:eating:cat:eating =
                                              m:cat:eating:cat:forming,
                m:cat:forming:cat:eating:cat:forming =
                                              d:cat:eating:cat:forming,
                m:cat:forming:cat:forming:slash = backwards,
                m:cat:forming:cat:forming:cat:eating:slash = no,
                m:cat:forming:cat:forming:cat:eating:cat:maj = np,
                m:cat:forming:cat:forming:cat:eating:cat:agr =
                        m:cat:eating:cat:eating:cat:agr,
                m:cat:forming:cat:forming:cat:forming =
```

```
                    m:cat:forming:cat:forming:cat:eating}.
```

Sample output

```
Script started on Sun Sep 22 17:20:21 1985
% prolog sv1
C Prolog version 1.4e.edai
PIMP - Prolog Implementation of PATR
$Revision: 0.6 $$Date: 85/09/03 12:21:04 $


File: def.pimp loaded

File: lex.pimp loaded

File gram.pimp loaded
Sentence ("halt" to quit): t

tracing is off
Sentence ("halt" to quit): the man who works for john exists

The string [the,man,who,works,for,john,exists] parses as an unknown with dag:

slash = no
cat:maj = s
    vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the men who works for john exist
Sentence ("halt" to quit): the man whom bill works for exists

The string [the,man,whom,bill,works,for,exists] parses as an unknown with dag:

slash = no
cat:maj = s
    vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the firm whom bill works for exists
Sentence ("halt" to quit): the man whose book mary loves exists

The string [the,man,whose,book,mary,loves,exists] parses as an unknown with da
g:
slash = no
cat:maj = s
    vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the men whose book mary loves exist

The string [the,men,whose,book,mary,loves,exist] parses as an unknown with dag
:
slash = no
cat:maj = s
    vform = fin
```

```
Return to continue (h for help):
Sentence ("halt" to quit): the man for whom fred works exists

The string [the,man,for,whom,fred,works,exists] parses as an unknown with dag:

slash = no
cat:maj = s
    vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the man for who fred works exists
Sentence ("halt" to quit): a book a picture of which exists exists

The string [a,book,a,picture,of,which,exists,exists] parses as an unknown with
 dag:
slash = no
cat:maj = s
    vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): a man a picture of which exists exists
Sentence ("halt" to quit): halt
| ?- ^Z
% Prolog execution halted
% ^Z
script done on Sun Sep 22 17:57:31 1985
```

Appendix C: Head-driven Phrase Structure Grammar implementation

Lexicon

```
a:      [head:maj = det,
        head:num = sg,
        subcat = [],
        foot:rel = [],
        foot:slash = [[]].

bill:   [head:maj = n, head:per = third, head:num = sg, head:gen = masc,
        subcat = [], foot:rel = [], foot:slash = [[]].

book:   [head:maj = n, head:per = third, head:num = sg, head:gen = neut,
        subcat:first:head:maj = det,
        subcat:first:head:num = sg,
        subcat:rest = [], foot:rel = [], foot:slash =[[]].

exists: [head:maj = v, head:vform = fin,
        subcat:first:head:maj = n,
        subcat:first:head:per = third,
        subcat:first:head:num = sg,
        subcat:first:case = nom,
        subcat:first:subcat = [],
        subcat:rest = [],
        foot:rel = [], foot:slash = [[]].

% plural "exist"

exist:  [head:maj = v, head:vform = fin,
        subcat:first:head:maj = n,
        subcat:first:head:num = pl,
        subcat:first:case = nom,
        subcat:first:subcat = [],
        subcat:rest = [],
        foot:rel = [], foot:slash = [[]].

firm:   [head:maj = n, head:per = third, head:num = sg, head:gen = neut,
        subcat:first:head:maj = det,
        subcat:first:head:num = sg,
        subcat:rest = [], foot:rel = [], foot:slash =[[]].

for:    [head:maj = p, head:pform = for,
        subcat:first:head:maj = n,
        subcat:first:case = acc,
        subcat:first:subcat = [],
```

```
        subcat:rest = [],
        foot:rel = [], foot:slash = [[]].

fred:   [head:maj = n, head:per = third, head:num = sg, head:gen = masc,
        subcat = [], foot:rel = [], foot:slash = [[]].

john:   [head:maj = n, head:per = third, head:num = sg, head:gen = masc,
        subcat = [], foot:rel = [], foot:slash = [[]].

loves:  [head:maj = v, head:vform = fin,
        subcat:first:head:maj = n,
        subcat:first:case = acc,
        subcat:rest:first:head:maj = n,
        subcat:rest:first:head:per = third,
        subcat:rest:first:head:num = sg,
        subcat:rest:first:case = nom,
        subcat:rest:first:subcat = [],
        subcat:rest:rest = [],
        foot:rel = [], foot:slash = [[]].

man:    [head:maj = n, head:per = third, head:num = sg, head:gen = masc,
        subcat:first:head:maj = det,
        subcat:first:head:num = sg,
        subcat:rest = [], foot:rel = [], foot:slash =[[]].

men:    [head:maj = n, head:per = third, head:num = pl, head:gen = masc,
        subcat:first:head:maj = det,
        subcat:first:head:num = pl,
        subcat:rest = [], foot:rel = [], foot:slash =[[]].

mary:   [head:maj = n, head:per = third, head:num = sg, head:gen = fem,
        subcat = [], foot:rel = [], foot:slash = [[]].

of:     [head:maj = p, head:pform = of,
        subcat:first:head:maj = n,
        subcat:first:case = acc,
        subcat:first:subcat = [],
        subcat:rest = [],
        foot:rel = [], foot:slash = [[]].

person: [head:maj = n, head:per = third, head:num = sg, head:gen = comm,
        subcat:first:head:maj = det,
        subcat:first:head:num = sg,
        subcat:rest = [], foot:rel = [], foot:slash =[[]].

picture:[head:maj = n, head:per = third, head:num = sg, head:gen = neut,
        subcat:first:head:maj = p,
        subcat:first:head:pform = of,
        subcat:first:subcat = [],
        subcat:rest:first:head:maj = det,
        subcat:rest:first:head:num = sg,
        subcat:rest:rest = [], foot:rel = [], foot:slash =[[]].
```

```
the:    [head:maj = det,
        subcat = [],
        foot:rel = [],
        foot:slash = []].

which:  [head:maj = n, head:gen = neut, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

% One "who" for each non-neuter gender

who:    [head:maj = n, head:gen = masc, case = nom, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

who:    [head:maj = n, head:gen = fem, case = nom, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

who:    [head:maj = n, head:gen = comm, case = nom, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

% One "whom" for each non-neuter gender

whom:   [head:maj = n, head:gen = masc, case = acc, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

whom:   [head:maj = n, head:gen = fem, case = acc, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

whom:   [head:maj = n, head:gen = comm, case = acc, subcat = [],
        head = foot:rel:first:head, foot:rel:first:subcat = [],
                foot:rel:rest = [],
        foot:slash = []].

whose: [head:maj = det,
        subcat = [],
        foot:rel:first:head:maj = n,
        foot:rel:first:subcat = [],
        foot:rel:rest = [],
        foot:slash = []].

works:  [head:maj = v, head:vform = fin,
        subcat:first:head:maj = p,
```

```
        subcat:first:head:pform = for,
        subcat:first:subcat = [],
        subcat:rest:first:head:maj = n,
        subcat:rest:first:head:per = third,
        subcat:rest:first:head:num = sg,
        subcat:rest:first:case = nom,
        subcat:rest:first:subcat = [],
        subcat:rest:rest = [],
        foot:rel = [],
        foot:slash = [[]].
```

Syntactic rules

```
% subject-predicate rule

m --> c, h,     {h:head:maj = v, h:subcat:rest = [],
                 c = h:subcat:first,
                 m:subcat = h:subcat:rest,
                 m:foot:rel = c:foot:rel,
                 m:foot:slash = h:foot:slash,
                 m:head = h:head}.

% preposition-NP rule

m --> h, c,     {h:head:maj = p, h:subcat:rest = [],
                 c = h:subcat:first,
                 m:subcat = h:subcat:rest,
                 m:foot = c:foot,
                 m:head = h:head}.

% verb-complement and noun-complement rule

m --> h, c,     {h:subcat:rest:rest = [],
                 c = h:subcat:first,
                 m:subcat = h:subcat:rest,
                 m:foot = c:foot,
                 m:head = h:head}.

% determiner-noun rule; foot features from determiner ("whose book")

m --> c, h,     {h:head:maj = n, h:subcat:rest = [],
                 c = h:subcat:first,
                 m:subcat = h:subcat:rest,
                 h:foot:rel = [],
                 h:foot:slash = [],
                 m:foot = c:foot,
                 m:head = h:head}.

% determiner-noun rule; foot features from noun ("a [picture of whom]")

m --> c, h,     {h:head:maj = n, h:subcat:rest = [],
                 c = h:subcat:first,
```

```
                m:subcat = h:subcat:rest,
                c:foot:rel = [],
                c:foot:slash = [],
                m:foot = h:foot,
                m:head = h:head}.

% antecedent-relative rule

m --> h, x,     {x:head:maj = v, x:head:vform = fin,
                x:subcat = [], x:foot:slash = [],
                h = x:foot:rel:first,
                m:foot:rel = x:foot:rel:rest,
                m:foot:slash = x:foot:slash,
                m:subcat = h:subcat,
                m:head = h:head}.

% leftward-extraction gap-filling

m --> x, h;     {h:head:maj = v, h:head:vform = fin,
                h:subcat = [], h:foot:rel = [],
                x = h:foot:slash,
                m:foot = x:foot,
                m:subcat = h:subcat, m:head = h:head}.

% gap-introducing rule

m --> x,        {m:head = x:head,
                m:foot:rel = x:foot:rel,
                m:foot:slash = x:subcat:first,
                m:subcat = x:subcat:rest}.
```

Sample Output

```
Script started on Sun Sep 22 15:10:35 1985
% prolog sv1
C Prolog version 1.4e.edai
PIMP - Prolog Implementation of PATR
$Revision: 0.6 $$Date: 85/09/03 12:21:04 $


File: def.pimp loaded

File: lex.pimp loaded

File gram.pimp loaded
Sentence ("halt" to quit): t

tracing is off
Sentence ("halt" to quit): the man who works for john exists

The string [the,man,who,works,for,john,exists] parses as an unknown with dag:
```

```
subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the men who works for john exist
Sentence ("halt" to quit): the man whom bill works for exists

The string [the,man,whom,bill,works,for,exists] parses as an unknown with dag:

subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the firm whom bill works for exists
Sentence ("halt" to quit): the man whose book mary loves exists

The string [the,man,whose,book,mary,loves,exists] parses as an unknown with da
g:
subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):the men whose book mary loves exist
Sentence ("halt" to quit): the men whose book mary loves exist

The string [the,men,whose,book,mary,loves,exist] parses as an unknown with dag
:
subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the man for whom fred works exists

The string [the,man,for,whom,fred,works,exists] parses as an unknown with dag:

subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): the man for who fred works exists
Sentence ("halt" to quit): a book a picture of which exists exists

The string [a,book,a,picture,of,which,exists,exists] parses as an unknown with
 dag:
```

```
subcat = []
foot:rel = []
     slash = []
head:maj = v
     vform = fin
Return to continue (h for help):
Sentence ("halt" to quit): a man a picture of which exists exists
Sentence ("halt" to quit): halt
| ?- ^Z
% Prolog execution halted
% ^Z
script done on Sun Sep 22 15:50:09 1985
```

References

Ajdukiewicz (1935) Die syntaktische Konnexität, *Studia Philosophica*, 3.1-27. English translation in *Polish Logic 1920-1939*, 207-231, edited by Storrs McCall, Oxford University Press.

Bresnan, Joan W. (1982) The Passive in Lexical Theory. In *The Mental Representation of Grammatical Relations*, edited by Joan W. Bresnan. Cambridge, Mass.:MIT Press.

Calder, Jonathan (1985) *PIMP, A Prolog Implementation of PATR-II.* Unpublished paper, The Centre for Cognitive Science (formerly School of Epistemics), University of Edinburgh.

Gazdar, Gerald (1982) Phrase Structure Grammar. In Pauline Jacobson and Geoffrey K. Pullum (eds.), *The Nature of Syntactic Representation*. Dordrecht: D. Reidel, 131-186.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum and Ivan A. Sag (1985) *Generalized Phrase Structure Grammar*, Oxford, Blackwell.

Pollard, Carl (1985) *Lectures on HPSG.* Unpublished paper, Stanford University.

Shieber, Stuart M., Hans Uszkoreit, Fernando C. N. Pereira, Jane Robinson and Mabry Tyson (1983) The Formalism and Implementation of PATR-II. In B. Grosz and M. Stickel (eds.) *Research on the Interactive Acquisition and Use of Knowledge*, SRI Final Report 1894, SRI International, Menlo Park.

Steedman, Mark J. (forthcoming) Combinatory Grammars and Parasitic Gaps.

Thompson, Henry and John Phillips (1984) *An Implementation of GPSG within the MCHART Chart Parsing Framework.* Unpublished paper, Department of Artificial Intelligence, University of Edinburgh.