A Computer Model of a Grammar for English Questions

Chris Callison-Burch

June 2000

Contents

Chapter 1

Introduction

This document describes my senior honors project, which is an implementation of a grammar for English questions. I have created a computer model of Ginzburg and Sag's theory of English interrogative constructions using the parsing software developed at the Center for Study of Language and Information (CSLI). In this chapter I describe the LKB parsing software, give instructions on downloading the system, and comment on the process of grammar engineering. The next chapter gives a summary of Ginzburg and Sag (2000). Chapter 3 details the discrepancies between the Ginzburg and Sag theory and my implementation. Chapter 4 provides a detailed discussion of a set of key example sentences. The appendices contain tables describing all the grammar constructions, lexical rules, types, and example lexical entries used in my implementation.

1.1 "Grammar Engineering" Explained

The process of modeling linguistic theories or "grammar engineering" is useful for two reasons. First, and more obviously, the building of a large scale grammar is useful for integration into natural language technologies. Because computers are formal systems that require precise, logical instructions in order to function, they are unable to use language, which is such a seamlessly intuitive process for humans. Grammars can be used to create a system of representation for utterances in a language, which allows a mapping to be specified between syntax and semantics. Therefore the the surface form of sentences, which has wide variation, can be represented in a more abstract, logical form. Logical representations are useful for such applications as machine translation, for sophisticated information retrieval, and for interfaces to various kinds of computer systems.

One possible approach to machine transltion is to build a logical representation of the meaning of an input string using a grammar for the source language. That representation then is fed to a grammar for the target language which generates a sentence in that language which has an identical meaning as the input sentence. Indeed, the LinGO grammar being developed at CSLI using the same tools that

I use, is integrated into the Verbmobil system, which performs speech to speech translations between German, English and Japanese.

The second reason that prototyping grammars is useful is that a computational model of a formal linguistic theory allows for practical testing of that theory on a wide range of data. This is of great use to linguists because the interaction of numerous, subtle constraints is often difficult to anticipate precisely. When an additional grammatical construction is posited to broaden the coverage of a grammar, it can have unexpected interactions with other pieces of the grammar and cause over-generation. Building a computational model of a grammar allows a linguist to quickly and accurately test the predictions made by that grammar.

1.2 The LKB

The LKB (Linguistic Knowledge Building) system is a grammar and lexicon development environment for use with constraint-based linguistic formalisms. It is specifically designed for the use of typed feature structures. It is intended to be used for Natural Language Processing research involving unification-based linguistic formalisms for parsing and generation. It has been most extensively tested with grammars based on Head-driven Phrase Structure Grammars (HPSG), such as the one described here, but it is intended to be compatible with most constraint-based grammatical formalisms.

The best way to think about the LKB system is as a development environment for a very high-level specialized programming language. Typed feature structure languages are essentially based on one data structure – the typed feature structure, and one operation – unification. This combination is powerful enough to allow the grammar developer to write grammars and lexicons that can be used to parse and generate natural language utterances. In effect, the grammar developer is a programmer, and the grammars and lexicons comprise code to be run by the system. The LKB system is therefore a software package for writing linguistic programs, i.e., grammars. (Copestake in preparation).

It's important not to confuse the LKB system and the grammars which run on it. To be clear, I have had no part in writing the LKB software – it was developed by others in the LinGO group at the Center for Study of Language and Information, including Ann Copestake, John Carroll, Rob Malouf, and Stephan Oepen – I have used the program to prototype a grammar based on the Ginzburg and Sag text.

1.3 Downloading the Software

The LKB grammar development environment is available as a UNIX executable file, and can be obtained from `http://www-csli.stanford.edu/~aac/lkb.html` as gzipped tar files. Use Netscape to download the file appropriate to your flavor of UNIX. At the command prompt type:

```
gunzip climimage.tar.gz
```

or

```
gunzip linuximage.tar.gz
```

depending on whether you are installing the Solaris or Linux version. This should give you a .tar file which you can extract the LKB image from by typing:

```
tar xf climimage.tar
```

or

```
tar xf linuximage.tar
```

This will create the new directory called `lkb/` containing several files including the `lkb` executable. If you have trouble installing the LKB please refer to its comprehensive documentation also available from the LKB homepage.

The grammar files need to be downloaded separately. They are linked from `http://www-csli.stanford.edu/~ccb`.

Chapter 2

Summary of Ginzburg and Sag (2000)

Ginzburg and Sag provide a comprehensive account of a wide range of syntactic and semantic facts involved with English interrogative constructions. Their theory is formulated in the constraint based linguistic formalism HPSG, and influenced by situation theory semantics. The constructions that they examine include:

- Subject and Non-subject *Wh*-Interrogative Clauses
- Polar interrogatives

as well as a treatment of more exotic IN-SITU *wh*-interrogatives such as "echo"and "reference" questions, which are generally not treated by other analyses and instead deemed extra-grammatical. I have modeled the theory presented in Ginzburg and Sag (2000) as closely as possible; I provide a detailed discussion about where my implementation deviates from the theory in the next chapter.

2.1 Overview of HPSG

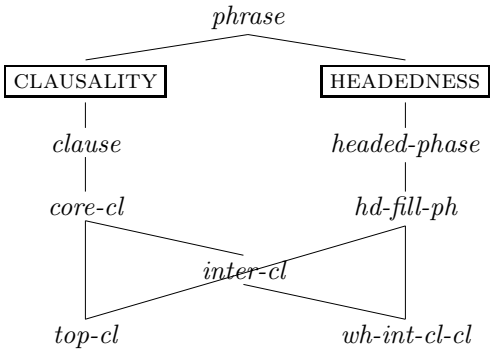
HPSG is a constraint-based linguistic formalism that uses typed feature structures to define a fine-grained typology of language. Utterances in HPSG are modeled as feature structures of type *sign*. The features associated with structures of this type include PHONOLOGY[1], and SYNSEM, the latter specifying both syntactic and semantic information. The two fundamental subtypes of *sign* are *word* and *phrase*.

Previous versions of HPSG have used type hierarchies to express generalizations about classes of words, and have included multiple inheritance as a way of generalizing across categories. For example, lexical types inherit from both a part of speech (such as verb-lexeme, prep-lexeme, or adj-lexeme) and an argument selection type

[1]My implementation uses the feature ORTHOGRAPHY rather than PHONOLOGY because I represent words with their conventional spellings rather than in a phonological form.

(such as strict-intransitive). This allows information which is common across different categories of words to be abstracted away from the individual lexical types. Beginning with Sag (1997), this multidimensional approach has recently been used to define construction hierarchies, which delineate the set of grammatical constructions, and is further developed by Ginzburg and Sag. Phrases are modeled in terms of their headedness and clausality. In this way generalizations across phrases are expressed - for example the *wh-interrogative-clause* and *topicalized-clause* constructions both inherit from *head-filler-phrase* gaining the constrains associated with that type, and inherit individually from the *core-clause* and *interrogative-clause* types which distinguish their behavior.

(1)



Perhaps the biggest change in the formulation of HPSG from Sag (1997) to Ginzburg and Sag (2000) is the adoption of the Generalized Head Feature Principle (shown in 2), which uses default unification to replace a number of other independent principles, including the The HEAD Feature Principle, the Valence Principle, the SLASH Inheritance Principle, and the STORE Inheritance Principle, which detail how information is passed from the head daughter to the mother phrase.

(2) Generalized Head Feature Principle (GHFP):

$$\textit{hd-ph} \quad \Rightarrow \quad \begin{bmatrix} \text{SS} \;/\; \boxed{1} \\ \text{HD-DTR} \left[\text{SS} \;/\; \boxed{1} \right] \end{bmatrix}$$

The '/' notation indicates a default unification. The default unification operation is formally defined in Lascarides and Copestake (1999), and affords an extremely elegant way of formulating constraints. It essentially means that all features are unified unless specified to have different values by a more specific type. Thus, rather than having to specify the identification of values for each feature through principles to factor out their application to specific types, the Generalized Head Feature Principle states that all syntactic and semantic information is shared by default between the mother and the head daughter and allows subtypes of *headed-phrase* to override the general constraint for specific features. That is, unless a construction specifies otherwise, then all features of a phrase are inherited from its head.

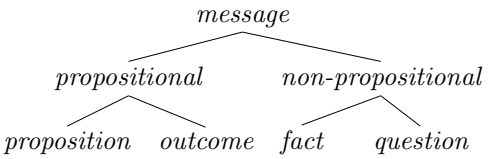
The Generalized Head Feature Principle is arguably a stronger claim about headedness than a set of independent inheritance principles. Rather than headedness being distributed over half a dozen independent constraints which only have the appropriate effect as a result of their interaction, the GHFP condenses the constraints into a single claim about headed phrases.

The mechanics of the version of HPSG used by Ginzburg and Sag are otherwise the same as in previous versions. I presuppose in the rest of this document that the reader has a familiarity with HPSG.

2.2 Semantic Argument Selection

‘ Following the work of Hamblin and Karttunen, Ginzburg and Sag develop a semantic typology which distinguishes between questions and propositions, but further posit a type *fact* and a type *outcome*, and motivate these types by showing that certain predicates select for clausal arguments which are describable neither as propositions nor as questions. The authors represent their semantic ontology HPSG's type system:

(3)



The type hierarchy above describes the abstract semantic entities which can be selected by predicates. The subtypes of the general type *message* are the semantic distinctions that are necessary to explain the argument selection for QUESTION EMBEDDING, TRUE/FALSE, FACTIVE, and MANDATIVE predicates.

One class of predicates embeds interrogative complements but not declarative complements:

(4) a. Kim asked/wondered/investigated who left.

b. Kim asked/wondered/investigated #that Sandy left.

This class of QUESTION EMBEDDING predicates helps motivate the ontological distinction between questions and other types, because members of this class select only for questions and ‘question–denoting nominals’ as their arguments.

A separate class of predicates, called TRUE/FALSE (TF) predicates which includes *believe, assert, deny, prove*, is incompatible with interrogative complements:

(5) a. # Bo supposes/assumes the question/issue of which pitcher will play tomorrow.

b. # Bo supposes/assumes which pitcher will do what tomorrow.

c. # Carrie claimed/argued who came yesterday.

d. # Carrie denies/doubts who stole Mo's key.

e. # Tony believes/suggests whether Bo stole Mo's key.

Furthermore, TF predicates can only cooccur with propositions and nominal complements of which truth can be predicated:

(6) a. #Jackie believed/doubted/assumed/proved Bo's weight/my phone number.

b. Jackie knows/discovered Bo's weight./my phone number.

c. #Billie's weight./my phone number is true/false

d. Jo believed/doubted/assumed Billie's story/the claim/the hypothesis/the charges/the forecast.

e. Billie's story/the claim/the hypothesis/the charges/the forecast is true/false.

Truth is predictable only of propositions, not of facts/possibilities:

(7) a. #The fact that Tony was ruthless is true.

b. #Tony's being ruthless is true.

c. The possibility that Glyn might get elected is true.

d. The claim/hypothesis/proposition that Tony was ruthless is true/false.

Therefore the type *fact* is included in the hierarchy, and is distinct and incompatible with the the type *proposition*. Predicates such as *know, discover, reveal, forget* are described as FACTIVE because their complements must semantically be facts.

The argumentation for the existence of *outcome* as a distinct type in the ontology is parallel to the one used to motivate the existence of the type *question*. There exists a semantically coherent class of MANDATIVE predicates, including predicates such as *demand, require, prefer, instruct*, which are incompatible with indicative declaratives:

(8) a. #Bo demanded that Mo is released.

b. #Kjell demanded that Billie leaves the party.

c. #The citizens of Worms pray that Solomon's book is found.

d. #The regulations require that Luca resigns next week.

e. #Glen orders that Billie is brought to see her.

but are acceptable with a distinct type of finite clause, subjunctives:

(9) a. Bo demanded that Mo be released.

b. Kjell demanded to leave the party.

c. The regulations require that Luca resign next week.

d. The citizens of Worms prayed that Solomon's book be found.

e. Glen ordered that we be brought to see her.

f. Glen ordered Billie be brought to see her.

Here is the lexical entry for the verb "wonder". Notice that it selects for an argument with semantic type *question*, and therefore makes the correct predictions about the data in (4).

(10)
$$\begin{bmatrix} \textsc{orth} \langle \text{ "wonder" } \rangle & \\ \textsc{ss|loc|cat} & \begin{bmatrix} \textsc{head} & \textit{verb} \\ \textsc{arg-st} & \langle [\,], \left[\textsc{loc|cont } \textit{question}\right] \rangle \end{bmatrix} \end{bmatrix}$$

Similarly, the lexical entry for *demand* or *require* would specify an argument structure where the second argument is type *outcome* which only gets assigned to subjunctive verbs or infinitives, and will therefore only allow arguments like those in (9), and rule out those in (8).

2.3 The Content of Questions

The contribution of *wh*-phrases in questions is captured by stipulating that questions have a feature call PARAMS with a value that is a set of parameters, which act as unbound variables – the semantic elements that are being asked about. Each *wh*-word introduces one parameter. The PARAMS feature performs two of the tasks associated with *wh*-phrases: (1) it links the parameter for each *wh*-word to the argument position within the *proposition* and (2) it introduces restrictions that the referent of the parameter must satisfy. Here are a few illustrations of the CONTENT for questions:

(11) a. *Who left?* $\mapsto$

$$\begin{bmatrix} \textit{question} \\ \text{PARAMS} & \left\{ \begin{bmatrix} \textit{param} \\ \text{INDEX} & \boxed{1} \\ \text{RESTR} & \left\{ \textit{person}(\boxed{1}) \right\} \end{bmatrix} \right\} \\ \text{PROP} & \begin{bmatrix} \textit{proposition} \\ \text{SIT} & s \\ \text{SOA} & \begin{bmatrix} \text{QUANTS} & \langle\,\rangle \\ \text{NUCL} & \begin{bmatrix} \textit{leave-rel} \\ \text{LEAVER} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

b. *Who greeted who?* $\mapsto$

$$\begin{bmatrix} \textit{question} \\ \text{PARAMS} & \left\{ \begin{bmatrix} \textit{param} \\ \text{INDEX} & \boxed{1} \\ \text{RESTR} & \left\{ \textit{person}(\boxed{1}) \right\} \end{bmatrix}, \begin{bmatrix} \textit{param} \\ \text{INDEX} & \boxed{2} \\ \text{RESTR} & \left\{ \textit{person}\boxed{2}) \right\} \end{bmatrix} \right\} \\ \text{PROP} & \begin{bmatrix} \textit{proposition} \\ \text{SIT} & s \\ \text{SOA} & \begin{bmatrix} \text{QUANTS} & \langle\,\rangle \\ \text{NUCL} & \begin{bmatrix} \textit{greet-rel} \\ \text{GREETER} & \boxed{1} \\ \text{GREETED} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that polar questions are treated uniformly in terms of an empty PARAMS value:

(12) *Did someone leave?* ↦

$$\begin{bmatrix} \textit{question} & & & \\ \text{PARAMS} & \{\} & & \\ \text{PROP} & \begin{bmatrix} \textit{proposition} & & \\ \text{SIT} & s & \\ \text{SOA} & \begin{bmatrix} \text{QUANTS} & \left\langle \begin{bmatrix} \textit{existential-qf} & \\ \text{INDEX} & \boxed{1} \\ \text{RESTR} & \left\{ \textit{person}(\boxed{1}) \right\} \end{bmatrix} \right\rangle \\ \text{NUCL} & \begin{bmatrix} \textit{leave-rel} & \\ \text{LEAVER} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Therefore, the generalization for the semantics for polar interrogative constructions is:

(13) $$\begin{bmatrix} \textit{question} & & \\ \text{PARAMS} & \{\ \} & \\ \text{PROP} & \begin{bmatrix} \textit{proposition} & \\ \text{SIT} & s \\ \text{SOA} & \textit{soa} \end{bmatrix} \end{bmatrix}$$

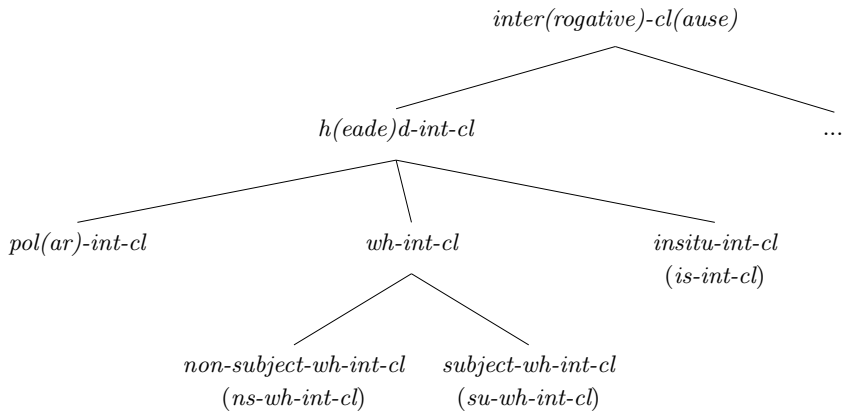
Wh-questions are similarly structured, but have a nonempty set of parameters:

(14) $$\begin{bmatrix} \textit{question} & & \\ \text{PARAMS} & \{\pi, ...\} & \\ \text{PROP} & \begin{bmatrix} \textit{proposition} & \\ \text{SIT} & s \\ \text{SOA} & \textit{soa} \end{bmatrix} \end{bmatrix}$$

2.4 Polar Interrogative Constructions

The type which introduces *question* as the value of the CONTENT of a sign is *interrogative-clause*. Each type of English question construction that Ginzburg and Sag examine is defined to be a subtype of that general type. The inheritance scheme for question types that they define is given in (15). Since interrogative constructions all inherit from the type *inter-cl* which constrains its CONTENT value, and thereby the CONTENT of all its subtypes, to be of type *question*.

(15)



Maximal subtypes of *inter-cl* inherit from the HEADEDNESS dimension of *phrase*, as well as from the CLAUSALITY dimension shown here. For instance, *polar-interrogative-clause*, which all 'yes-no' questions are instances of, is a subtype of *hd-int-cl* and *subject-auxiliary-inversion-phrase* (*sai-ph*).

(16) *sai-ph*:

$$\left[\text{SUBJ} \ \langle\ \rangle\right] \rightarrow \mathbf{H}\begin{bmatrix} \textit{word} & \\ \text{INV} & + \\ \text{AUX} & + \\ \text{SUBJ} & \langle \boxed{0} \rangle \\ \text{COMPS} & \boxed{A} \end{bmatrix}, \boxed{0}, \boxed{A}$$

The constraints contributed by *sai-ph* indicate that the phrase is an inverted (INV +), auxialary (AUX +) construction in which the subject argument follows the head, and the complements follow the subject. Thus the only construction-specific constraint that needs to be stated is the following:

(17) *pol-int-cl*:

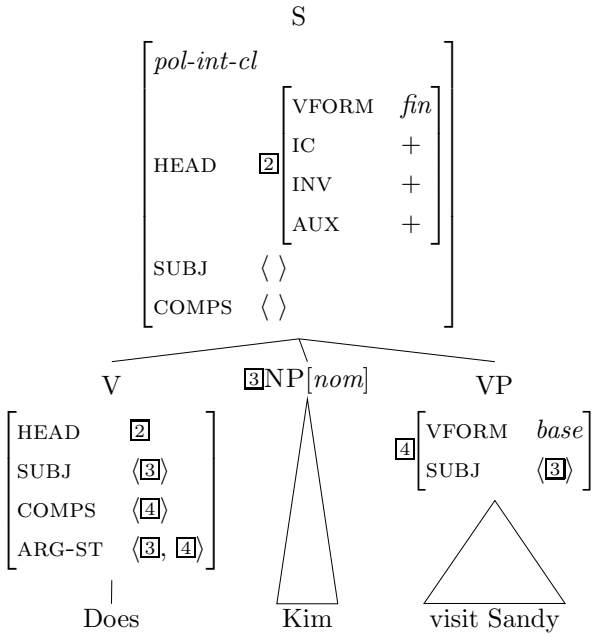
$$\begin{bmatrix} \text{CONT} & \begin{bmatrix} \textit{question} & \\ \text{PARAMS} & \{\} \\ \text{PROP} & \begin{bmatrix} \textit{proposition} \\ \text{SOA} \quad \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H}\begin{bmatrix} \text{IC} & + \\ \text{CONT} & \boxed{1} \end{bmatrix}, \ldots$$

That is, the only things that are specific to this construction are that its content is a question with no parapmeters to be filled (i.e. it is a question which can be answered with "yes" or "no") about the proposition formed by the content of the head daughter. Furthermore, because the head daughter is constrained to be an

independant clause (IC +), this is inherited up to the mother and allows polar interrogatives to stand alone as sentence – that is, they are not required to appear in an embedded context as would be the case if they were IC -.

A typical 'inverted' polar interrogative clause, satisfying all the inherited constraints, is sketched in (18).

(18)



"Does Kim visit Sandy?" is inverted – the subject follows the verb "does". It is an auxilary construction (see Section 4.2 for details), and it is an independant clause.

2.5 *Wh*-Questions

2.5.1 Unbounded Dependencies

The term "unbounded dependency" or "long-distance dependency" applies to a range of syntactic phenomena, such as topicalization, *wh*-questions, and relative clause constructions, and is used to describe a constituent's being dislocated from the place that it is normally realized, and occurring at a potentially unbounded distance from that location. Commonly these dependencies occur in filler-gap constructions:

(19) a. [These bagels]$_i$, I like __ $_i$. (topicalization)

b. [These bagels]$_i$, they say they like __ $_i$. (topicalization)

c. [Whose bagels]$_i$ do you like __ $_i$? (*wh*-interrogative)

d. [From whom]$_i$ did you buy these bagels __ $_i$? (*wh*-interrogative)

e. [What great bagels]$_i$ they bought __ $_i$! (*wh*-exclamative)

All of these phenomena are treated using the type *head-filler-phrase* which obeys the following constraint:

(20) *hd-fill-ph*:

$$\begin{bmatrix} \text{SUBJ} & \langle\ \rangle \\ \text{SLASH} & \boxed{\Sigma_2} \end{bmatrix} \rightarrow \begin{bmatrix} \text{LOC} & \boxed{1} \end{bmatrix},\ \text{H}\begin{bmatrix} \textit{phrase} & \\ \text{HEAD} & \textit{verb} \\ \text{SLASH} & \left\{\boxed{1}\right\} \uplus \boxed{\Sigma_2} \end{bmatrix}$$

(20) creates a phrase from two daughters, a non-head daughter whose LOCAL information is identified with an element from a head daughter's SLASH value.

Elements are added to the SLASH set if not canonically realized. "Canonically realized" constituents are those that participate in the standard *head-complement* and *head-subject* constructions. Any item which normally appears as a complement of a verb is treated as non-canonically realized when it appears in a fronted position through a *filler-head* construction instead of its normal position following that verb. All canonically realized arguments (i.e. items on the ARG-ST list) of a word are added to the word's SUBJ, SPR, and COMPS lists through the Argument Realization Principle (21). Arguments which are not canonically realized, and which participate in filler-gap dependencies, are marked as type *gap-synsem* and are subtracted from the COMPS list.

(21) Argument Realization Principle:

$$\textit{word} \Rightarrow \begin{bmatrix} \text{SS}|\text{LOC}|\text{CAT} & \begin{bmatrix} \text{SUBJ} & \boxed{A} & & \\ \text{SPR} & \boxed{B} & & \\ \text{COMPS} & \boxed{C} & \ominus & \text{list}(\textit{gap-ss}) \\ \text{ARG-ST} & \boxed{A} \oplus \boxed{B} \oplus \boxed{C} & & \end{bmatrix} \end{bmatrix}$$

The type *synsem* is divided into two subtypes – *canonical-ss* and *non-canonical-ss*, and *non-canonical-ss* is further divided into *gap-ss* and *pronominal-ss*. This distinction allows non-canonically realized *gap-ss* arguments to be marked for removal from the COMPS list. The type distinction also allows *gap-ss* to be further constrained:

(22) *gap-ss* $\Rightarrow$ $\begin{bmatrix} \text{LOC} & \boxed{1} \\ \text{SLASH} & \left\{\boxed{1}\right\} \end{bmatrix}$

Thus whenever a non-canonically realized element participates in a filler-gap dependency, its local information is put into SLASH. This information is passed up through the tree via the SLASH Amalgamation Constraint on words:

(23) SLASH Amalgamation Constraint

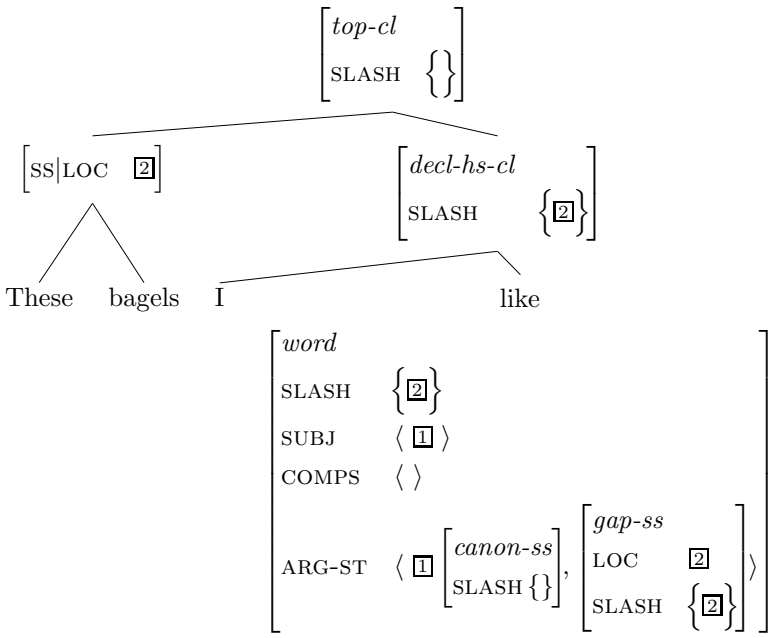
$$word \Rightarrow \left[\begin{bmatrix} \text{LOC}|\text{CAT} & \left[\text{ARG-ST} \quad \left\langle [\text{SLASH } \boxed{\Sigma_1}], \ldots, [\text{SLASH } \boxed{\Sigma_n}] \right\rangle\right] \\ \text{SLASH} & (\boxed{\Sigma_1} \cup \ldots \cup \boxed{\Sigma_n}) \end{bmatrix}\right]$$

This amalgamation constraint causes the SLASH value of a word to be the sum of the SLASH values of all of its arguments. Amalgamation is not specific to the SLASH value. It is a general mechanism which is harnessed for a number of other features, including STORE and WH. Whenever the value of a feature is dependent on all of a word's arguments the set union operation of amalgamation can be employed.

In the case of SLASH amalgamation whenever a word's argument is not canonically realized, it is added to the word's SLASH list. The value of a SLASH list is passed up to a phrase from its head daughter. An element is removed from the SLASH in a head-filler phrase construction, following the constraints in (20). The filler daughter of such constructions is identified with the item being removed from the SLASH list thus guaranteeing the dependency between the topicalized element or *wh*-word, and the "gapped" constituent.

The process of SLASH amalgamation is detailed in the following tree for the topicalized sentence "These bagels, I like."

(24)



"Like" contains two items on its ARG-ST list, which correspond to its subject and object. In this case the object is type *gap-ss*, which means that it is sbubtracted from the COMPS list in the ARP. Since the object is a gap-ss its SLASH list contains one elemnt which is coindexed with the object's LOC value. This SLASH list is amalgamated with the subject's empty SLASH list to the word's SLASH value. *Like*'s singleton SLASH is passed up to the head subject clause, and then discharged in the topicalze clause construction. Since the LOCAL value is coindexed with the object of *like*, "these bagels" is associated the semantic role that the object plays.

2.5.2 Nonsubject *Wh*-Interrogatives

In contrast to *pol-int-cl*, *ns-wh-int-cl* inherits from the CLAUSALITY dimension through *wh-int-cl* (shown in 25) and from HEADEDNESS through the *hd-filler-ph* (shown above in 20).

(25) *wh-int-cl*:

$$\left[\text{CONT} \quad \left[\text{PROP} \quad \boxed{2}\right]\right] \rightarrow [\ ]\ \mathbf{H}\left[\text{CONT} \quad \boxed{2}\right]$$

Therefore fronted *wh*-words are treated as extracted, just as topicalized elements are, and associated with their canonical argument position through the constraints on the *hd-filler* type. Furthermore, the [CONT *question*] of nonsubject *wh*-interrogatives is associated with a [CONT *proposition*] contained in the head daughter, because of the constraints of the *wh-int-cl* type. The elements of

the PARAMS of a question associate a *wh*-word with the semantic indices of that proposition via the 'Filler Inclusion Constraint':

(26) Filler Inclusion Constraint (FIC):

wh-int-cl:

$$\left[\text{CONT}\ \left[\text{PARAMS}\ \{\boxed{1}\} \uplus \mathit{set}\right]\right] \rightarrow \left[\text{WH}\ \left\{\boxed{1}\right\}\right],\ \mathbf{H}$$

The lexical entries of *wh*-words are distinguished from other words in that they have a non-empty set of elements for the feature STORE that is identified with the feature WH. Non-interrogative lexical entries are specified to have an empty set for the WH feature. The lexical entry for interrogative *who* may thus be formulated as in (27).

(27) Interrogative *who*:

$$\left[\begin{array}{ll}\text{ORTH} & \langle\ \text{“who”}\ \rangle \\ \text{SS} & \left[\begin{array}{ll}\text{LOC} & \left[\begin{array}{ll}\text{CAT} & \text{NP} \\ \text{CONT} & \left[\begin{array}{ll}\mathit{param} & \\ \text{IND} & i \\ \text{RESTR} & \{\ \}\end{array}\right] \\ \text{STORE} & \left\{\boxed{1}\left[\begin{array}{ll}\mathit{param} & \\ \text{IND} & i \\ \text{RESTR} & \{\mathit{person}(i)\}\end{array}\right]\right\}\end{array}\right] \\ \text{WH} & \left\{\boxed{1}\right\}\end{array}\right]\end{array}\right]$$

Because *wh*-words can be properly contained within the filler daughter (as opposed to the filler daughter being a single *wh*-word), additional constraints are employed to ensure that such phrases are allowed to participate in *wh-int-cl* constructions, and are not excluded by the Filler Inclusion Constraint. For example, in so-called "pied piping" dependencies, the *wh*-word is contained within the filler daughter at an arbitrary depth:

(28) a. I wonder [[*whose* cousin] they like __ best].

b. I wonder [[...[*whose* cousin]'s friend's ...dog] ate the pastry].

To guarantee that phrases which contain a *wh*-word occur in *wh-int-cl* constructions, Ginzburg and Sag posit the WH-Amalgamation Constraint, which behaves similarly to the SLASH Amalgamation Constraint in that the value of a word's WH feature is the amalgamation of its arguments WH values:

(29) WH Amalgamation Constraint

$$word \Rightarrow \left[\text{SS} \left[\begin{array}{ll} \text{LOC|CAT|ARG-ST} & \left\langle [\text{WH } \boxed{\Sigma_1}], \ldots, [\text{WH } \boxed{\Sigma_n}] \right\rangle \\ \text{WH } (\boxed{\Sigma_1} \cup \ldots \cup \boxed{\Sigma_n}) & \end{array} \right] \right]$$

which ensures that any word which has an argument with a non-empty WH value will itself have a non-empty WH value. This allows *wh*-words to be properly contained within phrases, and thus allows for the pied piping examples.

Other construction specific constraints are also formulated. For instance, there are two environments in which *ns-wh-int-cl* behaves differently – independent clauses and embedded clauses. In independent nonsubject *wh*-interrogatives there is an inverted auxiliary verb, whereas in embedded nonsubject *wh*-interrogatives are there isn't:

(30) a. [Who [will Sandy visit __]]?

b. *[Who [Sandy will visit __]]?

c. They wonder [who [Sandy will visit __]].

d. *They wonder [who [will Sandy visit __]].

Therefore *ns-wh-int-cl* is itself further constrained:

(31) Inversion Constraint (INVC):

$$[\ ] \rightarrow [\ ] \ \mathbf{H}\begin{bmatrix} \text{IC} & \boxed{1} \\ \text{INV} & \boxed{1} \end{bmatrix}$$

This constraint guarantees that main-clause [IC +] (IC represents the notion of INDEPENDENT CLAUSE), questions are inverted and that embedded [IC −] questions are uninverted.

2.5.3 Subject *Wh*-Interrogatives

Ginzburg and Sag treat subject *wh*-interrogatives similarly to nonsubject *wh*-interrogatives in that they treat the *wh*-constituent as extracted. In English subject questions, the *wh*-phrase may appear to be in the subject position when compared to analagous declarative construction:

(32) a. *Who* visits Merle?

b. *Whose friends* left?

c. Kim visits Merle.

d. Kim's friends left.

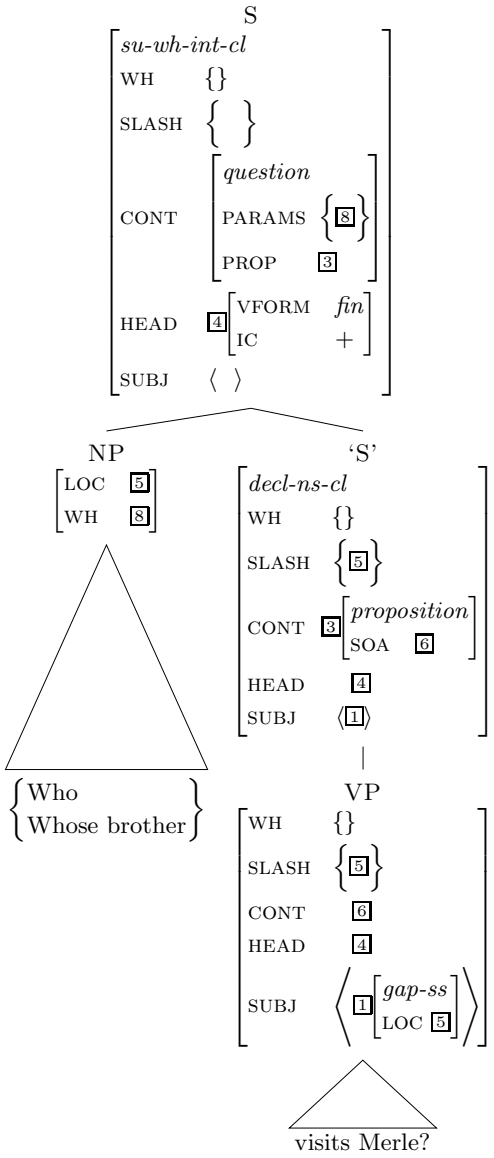
However, Hukari and Levine (1995) show cross-linguistic evidence that languages which exhibit extraction sensitive phenomena through morpological marking also exhibit the same behavior for subject *wh*-interrogative constructions. This suggests the generalization that all *wh*-questions should be treated in terms of selection of a slashed element by a lexical head. Therefore Ginzburg and Sag treat this kind of interrogative as a head-filler construction where the subject is extracted:

(33) *su-wh-int-cl*:

$$[\ ] \rightarrow \left[\text{LOC } \boxed{1}\right], \quad \mathbf{H}\left[\text{SUBJ } \left\langle \begin{bmatrix} \textit{gap-ss} \\ \text{LOC } \boxed{1} \end{bmatrix} \right\rangle\right]$$

A *declarative-nonsubject-clause* non-branching construction is used to the extract the subject, and create the appropriate [CONTENT *proposition*] semantics which the question will take as its PROP value. A typical example of a subject *wh*-interrogative clause is the following:

(34)



Thus in all subject *wh*-questions, the subject NP is in fact extracted and the verb heading such a question is slashed. This treatment is thus consistent with the generalizations of Hukari and Levine.

2.6 Multiple *Wh*-Questions

Recall that each interrogative *wh*-word (like 27) has a parameter which is associated with a semantic argument of the verb. Other than the filler daughter of subject

and non-subject *wh*-interrogative constructions, which are required by the Filler Inclusion Constraint (26) to contribute their parameter to the questions PARAMS feature, other *wh*-words are not constrained as to when they contribute their parameter to the set. Therefore if one *wh*-interrogative clause is embedded within another, the parameter of the embedded *wh*-interrogative word may be retrieved from storage at the level of the embedded clause or the highest clause. This leads to ambiguities in multiple *wh*-questions like

(35) Who wondered who saw what?

The parameter of "what" can thus become part of the meaning of either interrogative clause, leading to ambiguous interpretations. The question can either-be interpreted as asking *who is the x such that x wondered: 'who saw what'* or interpreted as asking *who is the x and what is the y such that x wondered: 'who saw y'*.

The mechanism for this parameter retrieval is treated similarly to the technique of quantifier storage pioneered by Cooper (1975, 1983). In 'Cooper storage' stored quantifiers are gathered up and passed up to successively higher levels of structure until an appropriate scope assignment locus is reached.

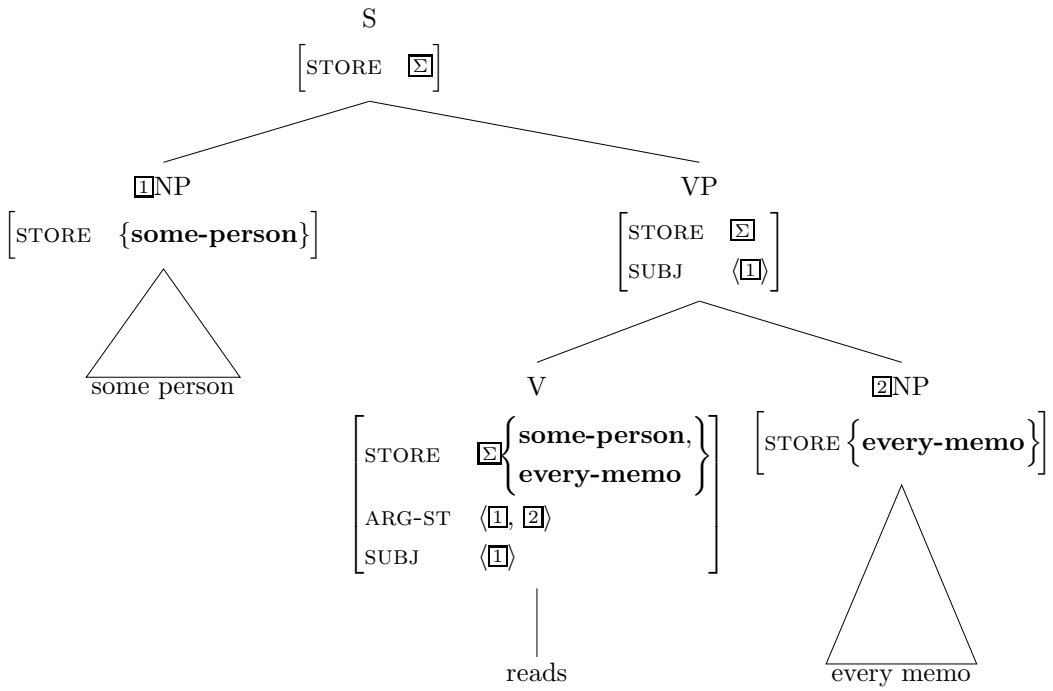
Pollard and Yoo propose a slight modification to Cooper's storage method, so unscoped quantifiers are instead passed up to the mother in a headed structure not from all the daughters, but only from the semantic head daughter. To achieve this, they constrain the STORE value of a verb, requiring that it be the set union of the STORE values of the verb's ARG-ST members. We may adapt this proposal in terms of the STORE Amalgamation Constraint formulated in ((36)).

(36) STORE Amalgamation Constraint (STAC):

$$\textit{word} \quad \Rightarrow \quad \left[\text{SS}|\text{LOC} \begin{bmatrix} \text{ARG-ST} & \langle\, [\text{STORE } \boxed{\Sigma_1}], \ldots, [\text{STORE } \boxed{\Sigma_n}] \,\rangle \\ \text{STORE} & \boxed{\Sigma_1} \cup \ \ldots \ \cup \boxed{\Sigma_n} \end{bmatrix} \right]$$

On this approach, the STORE of the verb in (37) is nonempty and is passed up the tree from head daughter to mother.

(37)



This same STORE Amalgamation Constraint is adapted for use with the parameters contributed by *wh*-words. It now contains two kinds of *scope-objects* – *quantifiers*, contributed by words like "some" and "every", and *parameters*, contributed by *wh*-words. The retrieval of *parameters* from the STORE list is constrained by the *Wh*-Question Retrieval principle (38) which requires that the STORE value of a *hd-int-cl* be the head daughter's STORE value, minus a set of parameters that become the clause's PARAMS set.

(38) *Wh*-Question Retrieval (WHQR)

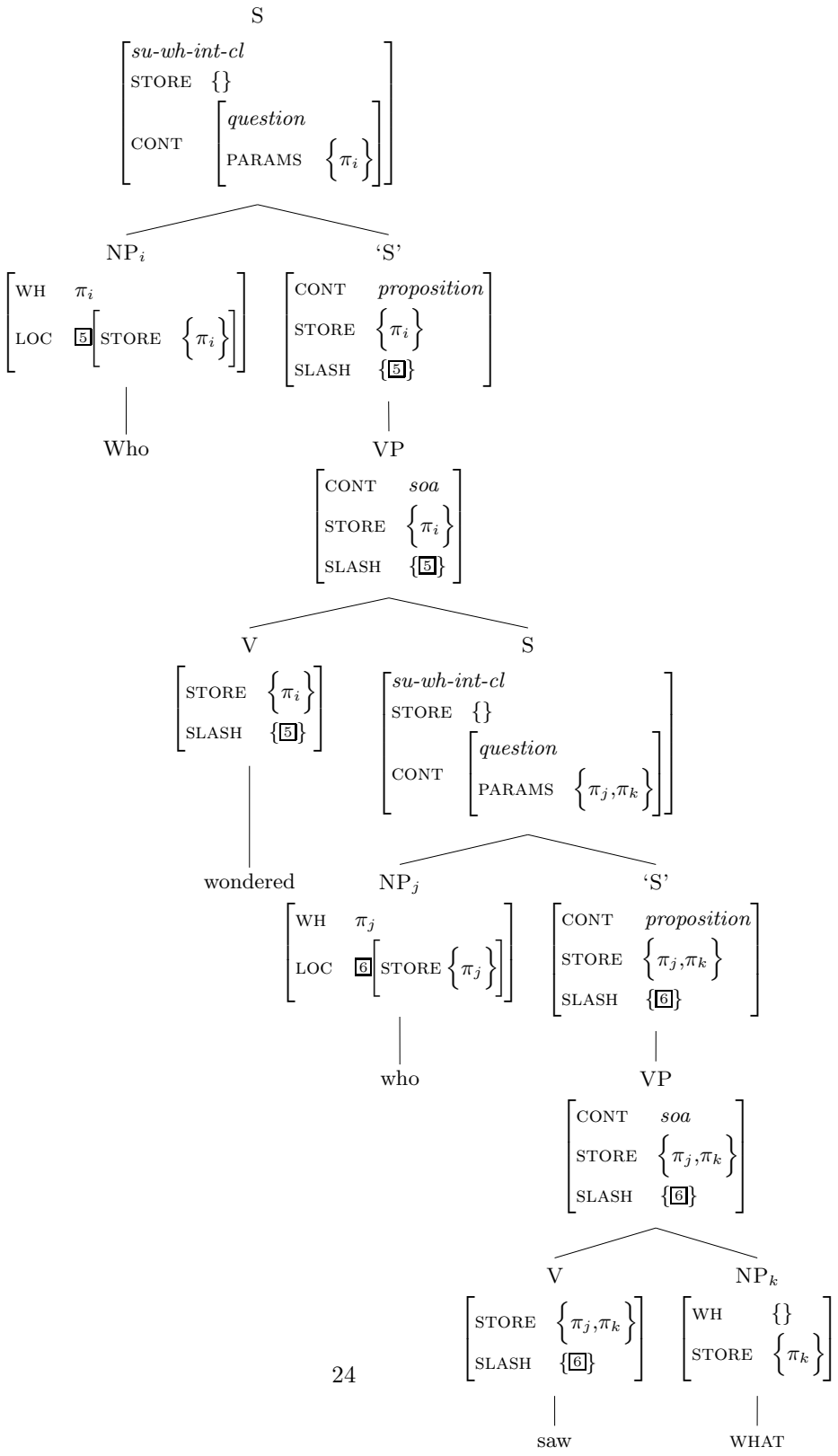
hd-int-cl:

$$\begin{bmatrix} \text{STORE} & \boxed{\Sigma_1} \\ \text{CONT} & \begin{bmatrix} \text{PARAMS} & \boxed{\Sigma_2} \end{bmatrix} \end{bmatrix} \rightarrow \ldots \mathbf{H}\begin{bmatrix} \text{STORE} & \boxed{\Sigma_1} \uplus \boxed{\Sigma_2} \end{bmatrix} \ldots$$

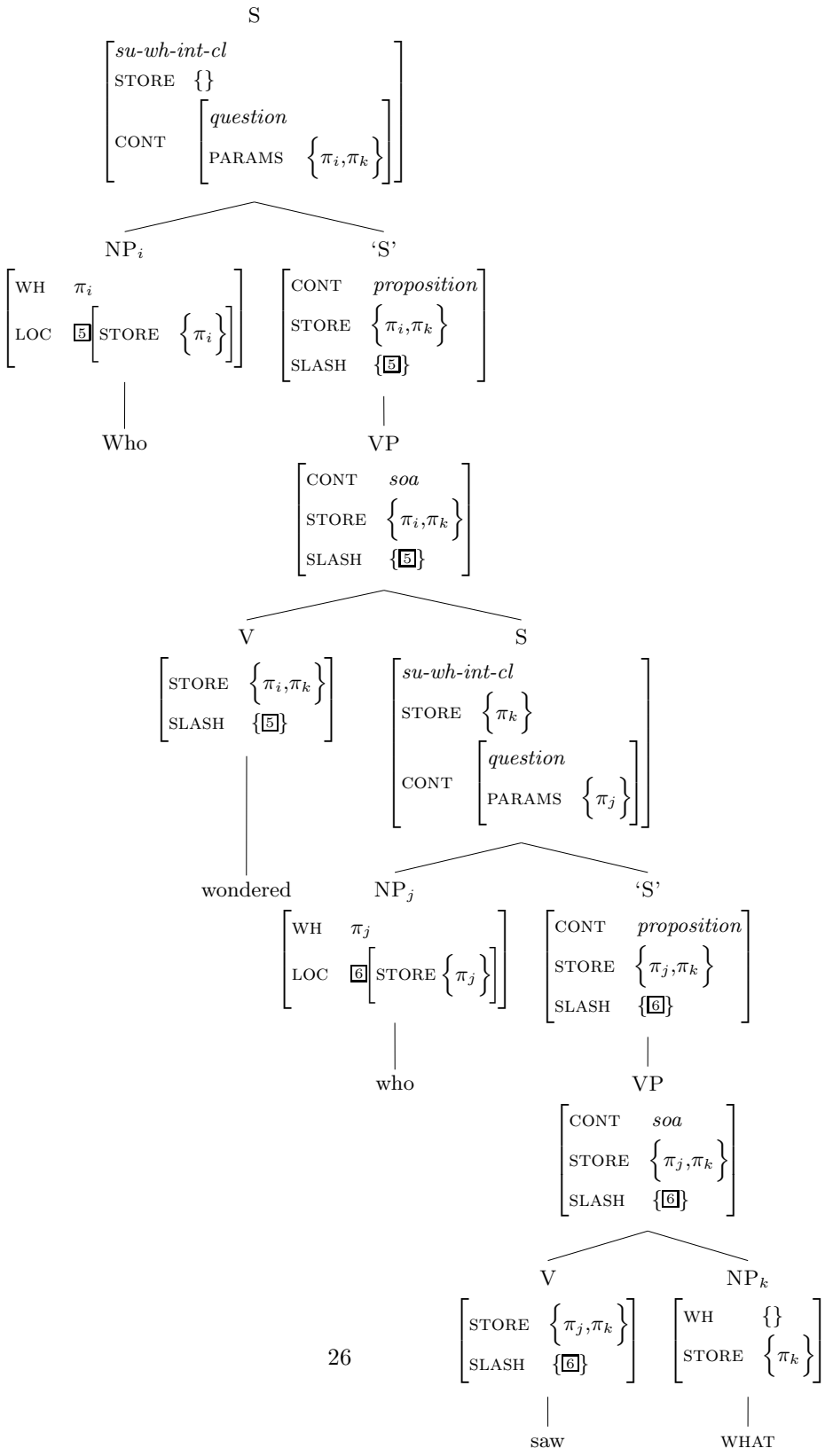
This constraint along with the Filler Inclusion Constraint (26) allows for exactly the distribution of parameters in multiple *wh*-quesions, as with the ambiguous interpretations of (35) above.

Nothing requires that the parameter of "what" be retrieved in the lowest clause. If it is not, it continues being passed up in STORE and is retrieved in the higher interrogative clause. The two alternative readings are depicted below:

(39)


S
su-wh-int-cl
STORE {}
CONT question
PARAMS $\{\pi_i\}$
NP$_i$
'S'
WH π_i
LOC 5 STORE $\{\pi_i\}$
CONT proposition
STORE $\{\pi_i\}$
SLASH {5}
Who
VP
CONT soa
STORE $\{\pi_i\}$
SLASH {5}
V
S
STORE $\{\pi_i\}$
SLASH {5}
su-wh-int-cl
STORE {}
CONT question
PARAMS $\{\pi_j,\pi_k\}$
wondered
NP$_j$
'S'
WH π_j
LOC 6 STORE $\{\pi_j\}$
CONT proposition
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
who
VP
CONT soa
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
V
NP$_k$
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
WH {}
STORE $\{\pi_k\}$
saw
WHAT

(40)


S
su-wh-int-cl
STORE {}
CONT question
PARAMS $\{\pi_i,\pi_k\}$
NP$_i$
'S'
WH π_i
LOC 5 STORE $\{\pi_i\}$
CONT proposition
STORE $\{\pi_i,\pi_k\}$
SLASH {5}
Who
VP
CONT soa
STORE $\{\pi_i,\pi_k\}$
SLASH {5}
V
S
STORE $\{\pi_i,\pi_k\}$
SLASH {5}
su-wh-int-cl
STORE $\{\pi_k\}$
CONT question
PARAMS $\{\pi_j\}$
wondered
NP$_j$
'S'
WH π_j
LOC 6 STORE $\{\pi_j\}$
CONT proposition
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
who
VP
CONT soa
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
V
NP$_k$
STORE $\{\pi_j,\pi_k\}$
SLASH {6}
WH {}
STORE $\{\pi_k\}$
saw
WHAT

Thus the parameter storage mechanism allows for multiple *wh*-questions. Ginzburg and Sag give further analysis of IN SITU questions, but the discussion of that analysis will be postponed to the Key Examples Chapter.

Chapter 3

Theory vs. Implementation

In Chapter 1, I compared typed feature structures to programming languages. Since linguists working within constraint-based frameworks like HPSG use these computationally amenable descriptions as a formal way of specifying linguistic behavior, for the most part the grammars written by linguists may be straightforwardly encoded into computational systems. That being said, there are a number of points at which my implementation departs from the theory outlined by Ginzburg and Sag. In this chapter, I highlight those points of departure.

3.1 Determinacy in Grammar Rule Length

A simple point that illustrates how the implementation differs from the theory is that the implementation requires all grammar rules to be a determinate length (because of the requirements of the LKB's chart parser), whereas the theory uses an abbreviation like the following to represent *head-complement* phrases with any number of complements[1]:

(41) *hd-comp-ph*:

$$[\ ] \rightarrow \mathbf{H}\begin{bmatrix} word \\ \text{COMPS} \ \boxed{A} \end{bmatrix}, \boxed{A}$$

However, my implementation requires an individual *hd-comp-ph* grammar rule for each COMPS list length. Therefore, I have limited the number of complements

[1] Ginzburg and Sag's notation in (41) is intended to be equivalent to the following:

(i)
$$\textit{hd-comp-ph} \ \Rightarrow \begin{bmatrix} \text{HD-DTR} & \begin{bmatrix} word \\ \text{COMPS} & \text{signs-to-synsems}(\boxed{A}) \end{bmatrix} \\ \text{NON-HD-DTRS} & \boxed{A} \end{bmatrix}$$

where the 'signs-to-synsems' is a function which relates a list of signs to the corresponding list of the synsems of those signs. I'm neglecting the fact that this sort of function is not definable within the LKB, and instead trying to illustrate the simpler point that the LKB requires grammar rules to be of a determinate length.

to two, and created a *hd-comp-ph* construction for each list length:

```
hd-comp-ph := hd-ph &
[ HD-DTR #1,
  DTRS < word & #1, ... > ].

hd-comp-ph-0 := hd-comp-ph &
[ HD-DTR.SS.LOC.CAT.COMPS < >,
  DTRS < sign > ].

hd-comp-ph-1 := hd-comp-ph &
[ HD-DTR.SS.LOC.CAT.COMPS < #1 >,
  DTRS < sign, phrase & [ SS #1 ] > ].

hd-comp-ph-2 := hd-comp-ph &
[ HD-DTR.SS.LOC.CAT.COMPS < #1, #2 >,
  DTRS < sign, phrase & [ SS #1 ], phrase & [ SS #2 ] > ].
```

Thus for each construction in Ginzburg and Sag (2000) which uses such a shorthand for the number of constructions, in my implementation balloons accordingly.

Notice that there is an additional deviation in the rules shown above – I have replaced the feature NH-DTRS (NON-HEAD-DAUGHTERS) with the feature DTRS and I coindex the HD-DTR with an item on that list. This allows me to state simply to the LKB that the order of daughters in any construction is simply the order of the DTRS list. Rather than complicate the implementation by defining a set of linear ordering constraints such as those described in Pollard and Sag (1987) (which would be theoretically possible within the LKB), I have hand-defined the ordering of the daughters of each construction using this generalized feature.

This also allows the composition of ORTHOGRAPHY to be simplified. I use a standard DIFFERENCE LIST appending technique to build up the spellings on phrases.

3.2 Difference Lists

Difference lists are a special type of list where a pointer is maintained to the end of the list (the LAST feature). The are defined as follows:

```
*diff-list* := *top* &
 [ LIST *list*,
   LAST *list* ].
```

and specified to be the type for ORTH:

```
sign := feat-struc &
[ ORTH *diff-list*,
  SS synsem,
  DTRS list-of-signs ].
```

which allows a phrase's orthography to be composed from its daughter's orthographies:

```
unary-construction := phrase &
[ ORTH #orth,
  DTRS < [ ORTH #orth ] > ].


binary-construction := phrase &
[ ORTH [ LIST #first,
         LAST #last ],
  DTRS < [ ORTH [ LIST #first,
                  LAST #second ] ], [ ORTH [ LIST #second,
                                             LAST #last ] ] > ].

ternary-construction := phrase &
[ ORTH [ LIST #first,
         LAST #last ],
  DTRS < [ ORTH [ LIST #first,
                  LAST #second ] ], [ ORTH [ LIST #second,
                                             LAST #third ] ],
         [ ORTH [ LIST #third,
                  LAST #last ] ] > ].
```

Difference lists can be used in grammars as a general way of appending lists simply using unification. In fact, difference list unification is the only method of appending lists available in the LKB.

It's also important to note that there is no list subtraction operation in the LKB. Therefore it's not possible to formulate the Argument Realization Principle (given in 21) in the implementation the same way as formulated in the Ginzburg and Sag theory. The theory's definition of the ARP gives the COMPS list in terms of the subtraction of a list of *gap-synsems*. While it's straightforward to define the ARG-ST as the append of the SUBJ, SPR, and COMPS using difference lists (as below), it is impossible to remove the *gap-synsems* from the COMPS using the same operation.

```
word := lex-sign &
[ SS /l #ss & [ LOC.CAT [ SUBJ [ LIST #first,
                                 LAST #second ],
                          SPR [ LIST #second,
                                LAST #third ],
                          COMPS [ LIST #third,
                                  LAST #last ],
                          ARG-ST [ LIST #first,
                                   LAST #last ] ] ],
  DTRS < [ SS /l #ss ] > ].
```

If the SLASH were a *list-of-synsems* rather than a *list-of-locals* and if it were the case that extracted arguments always came from the beginning of the COMPS list or the end of it, then it would be possible to use the difference list append to model the ARP. Since this is not the case, there needs to be some other mechanism removes items of type *gap-ss* from the COMPS list so that extraction will work correctly.

The removal of *gap-synsems* could be implemented though constraining type hd-comp constructions to be binary branching. This would allow a non-branching rule access to any item on the COMPS list, and any item could therefore be removed and typed as *gap-ss* for extraction. However, while this is a viable option (indeed, it is the solution used in the LinGO English Resource Grammar), I choose not to use it since it would dramatically change the appearance of the trees, and I wanted to try to match the Ginzburg and Sag text in appearance as well as theory. Instead, I use a set of non-branching pumping rules which apply to lexemes to remove arguments from the COMPS list. I'll explain this in more detail in the next section.

3.3 Pumping Rules and Intermediate Types

For the implementation of the Sag and Wasow (1999) textbook which I did two years ago, I developed a novel technique for implementing Principle A of the binding theory. The textbook's binding theory is formulated in English as:

- Principle A: An [ANA +] synsem-struc must be outranked by a coindexed synsem-struc.
- Principle B: An [ANA –] synsem-struc must not be outranked by a coindexed synsem-struc.
- The Anaphoric Agreement Principle further states that coindexed elements share the same AGR value.

It is not immediately clear how to treat this within the typed feature structure formalism, unless one has the massive additional power (and resulting inefficiency) of a framework like Richter's (1999, 2000) RSRL. It is difficult to require that a synsem be coindexed with another synsem that outranks it (that is, precedes it on an ARG-ST list), because "outranking" or "preceding" is a difficult notion to define when unification is the only operation available. One possible solution is to bound the length of argument structures, breaking the length into specific instances, and then describe each valid argument structure explicitly.

For example, the following would be examples of valid ARG-ST lists:

(42) a. $\langle \left[\text{SYN}|\text{HEAD}|\text{ANA} -\right], \left[\text{SYN}|\text{HEAD}|\text{ANA} -\right] \rangle$

b. $\langle \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} - \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} + \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix} \rangle$

c. $\langle \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} - \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} + \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD}|\text{ANA} - \end{bmatrix} \rangle$

d. $\langle \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} - \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD}|\text{ANA} - \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} + \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix} \rangle$

e. $\langle \begin{bmatrix} \text{SYN}|\text{HEAD}|\text{ANA} - \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} - \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix}, \begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{ANA} + \\ \text{AGR } \boxed{1} \end{bmatrix} \\ \text{SEM}|\text{INDEX } \boxed{2} \end{bmatrix} \rangle$

f. et cetera

One can clearly see that, though perhaps tedious, it is possible to enumerate argument structures for a bounded list length. Furthermore, since argument structures are generally no longer than three items long (possibly four in the case of the verb *trade*), it's actually manageable to do this enumeration.

It would be nice if enforcing the binding principles held was as simple as defining each valid argument-structure as a subtype of the type *arg-st*. However, creating subtypes of *arg-st* is not enough to constrain the ARG-ST list, because there is no mechanism within the LKB for forcing a general type to be a more specific subtype. This is available within the model theory governing the textbook's theory because valid feature structures can be described as being composed only of maximal subtypes for certain values. That is, the Chapter 6 of the textbook describes "resolved feature structures" as having the value for each feature be specified and maximally specific. Therefore if the subtypes of the type for ARG-ST were constrained as in (42), then a resolved feature structure containing the feature ARG-ST would have to have a maximal specific subtype as its value. However, this constraint on the well-formedness of feature structures is not available within the LKB, and does not seem feasible within a computationally efficient system.

The problem then becomes one of mandating a type change, or forcing a set of constraints on a structure. The strategy that I developed was to have a set of non-branching pumping rules which applied to an (arbitrarily created) intermediate type between *lexeme* and *word*. In order to move from this intermediate type to *word*, a sign's argument structure was forced to undergo a non-branching grammar rule which caused its argument structure to unify with one of the valid ARG-ST types. For example:

```
binding-rule-5 := word &
[ SYN #syn,
  SEM #sem,
  ARG-ST #arg-st & < [ SYN [ HEAD [ ANA false, AGR #agr ] ],
                       SEM [ INDEX #ind ] ],
```

```
                        [ SYN [ HEAD [ ANA true, AGR #agr ] ],
                          SEM [ INDEX #ind ] ] >,

DTRS < inflected-lexeme & [ SYN #syn,
                            SEM #sem,
                            ARG-ST #arg-st ] > ].
```

The argument structure specification in the rule above corresponds to the argument structure in (42b). The rule applies to a synsem-struc of the intermediate type *inflected-lexeme*, and produces a word which is identical to the inflected-lexeme except that it's argument structure has been constrained in accordance with the binding restrictions. This pumping rule is similar to the zero-complement application of the *hd-comp* construction which vacuously applies to a word which has no complements, and changes it into a phrase. In the Sag and Wasow (1999) textbook grammar the pumping of the *hd-comp* rule is mandatory because all other grammar rules apply to feature structures of type *phrase*.

So in the theory outlined in the textbook, each lexeme must pass through an inflectional rule before it becomes a word, which the head complement rule applies to, creating a phrase which all other grammar rules apply to. In my implementation of the textbook grammar, each lexeme must pass through an inflectional rule, becoming an inflected lexeme which must go through a binding rule in order to become a word which the head complement rule then applies to.

This strategy of having an intermediate type through which lexemes must pass, which constrains them in some fashion, is a general one. It can be applied to a number of circumstances, often without any efficiency loss.[2] Indeed, in the implementation of Ginzburg and Sag I use this strategy to ensure that the ARG-ST is the append of the SPR, SUBJ and COMPS and that all *gap-synsems* are removed from the COMPS list as per the Argument Realization Principle. Furthermore, I use it to do STORE, WH and SLASH Amalgamation.

The non-branching rules that I use for the above principles are ordered to avoid ambiguous parses which would result from applying the rules in varying orders. I force an ordering by creating an intermediate type for each rule, and causing each rule to apply to a different intermediate type and to produce the next intermediate type, eventually producing the type *word*, which the rest of the grammar then applies to. Each of the intermediate types correspond to the application of the constraints that apply using these pumping rules. Therefore there is one type *inflected-lexeme+arp*, which applies the Argument Realization Principle, another *inflected-lexeme+amalg* which applies the STORE, SLASH, and WH Amalgamation Constraints, and *inflected-lexeme+gap* which marks elements on the ARG-ST as type *gap-ss* and removes them from the COMPS list.

[2]The application of the pumping rule for binding does cause a efficiency problem because for each inflected lexemes it creates a number of words – one for each valid argument structure of that lexeme's ARG-ST length. The parse chart is then instantiated with these extra edges.

3.4 Sets Treated as Lists

Because set theoretic operations are not conducive to efficient computation, they have not been implemented in the LKB. As such, sets were not available to me as a type of object that could be part of the implementation. Therefore, where the theory uses sets as feature values, I instead substituted difference lists, and simulated set union with difference list appends. In places where the non-deterministic retrieval of elements of a set was required (as with *Wh*-Question Retrieval described in (38)), I have simulated the retrieval by assuming a bounded set length and then creating a number of rules which retrieved from every possible positions on the list.

Interestingly, treating the value of SLASH as a list rather than as a set makes a valuable linguistic prediction – causing SLASH to be a list, and limiting the retrieval of filler elements to the start of that list rules out cross serial dependencies in cases of multiple extraction. The Bouma *et al.* (in press) analysis of extraction, which the Ginzburg and Sag work builds on, fails to make this prediction and instead rules the unacceptability of such sentences to be linked to processing factors, and not the the grammar itself. The Bouma *et al.* (in press) does harness the set union operation to make correct predictions about parasitic gaps, relying on the collapsing of values within the set. I am unable to simulate this behavior with lists, without the implementation becoming incredibly cumbersome. I instead take the cross serial predictions made by lists as a positive tradeoff for the loss of the parasitic gap predictions, and have therefore not simulated the set behavior for SLASH.

Chapter 4

Key Examples

In this chapter I go through the analysis for a set of key examples in great detail. I start by walking through the parses for a set of simple declarative sentences, then describe my treatment of the auxiliary "do", then show examples of *wh*-interrogative constructions, then illustrate some phenomena sensitive to the presence of *wh*-words, then show examples of multiple *wh*-questions, and finally describe some of the analysis of IN SITU questions.

4.1 Simple Declaratives

(43) a. Kim left.

b. *Kim leave.

The parse tree for the simple declarative statement "Kim left" is shown in Figure 4.1. Notice that the nodes on the parse trees that my implementation produces show the specific type for each phrase, rather than using the standard (and less informative) abbreviations NP, VP, PP, etc. The top node in this tree represents the application of the *declarative head-subject clause* construction, and is applied to a *non-clausal head-complement phrase* and a *non-clausal verb phrase* both of which take zero complements and simply pump the words into phrases.

The verb "leave" inherits from the lexeme type *strict-intransitive-verb* which inherits from the PART OF SPEECH partition of *lexeme* through *non-auxiliary-verb-lexeme* and from ARGUMENT SELECTION through the *strict-intransitive* type. The information which each of these types contributes through inheritance is shown in (44).

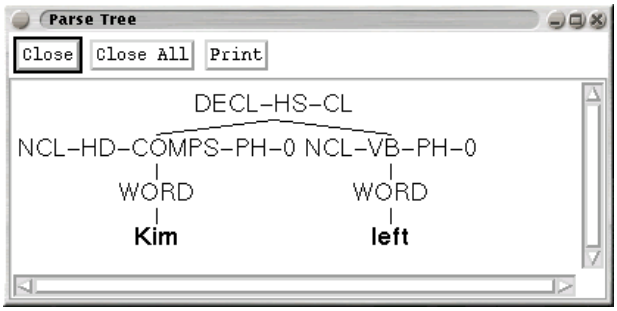
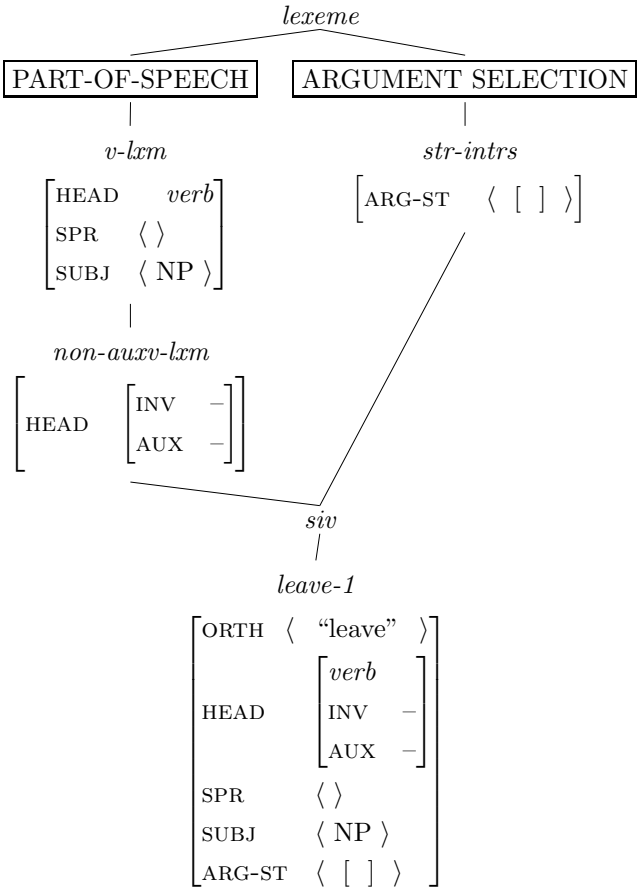


Figure 4.1: The parse for "Kim left."

(44)



The Past Tense Verb Inflectional Rule applies to the lexical entry, inflecting the verb[1] and adding the information that it is a finite, non-predicative verb which describes a *realis* state of affairs:

(45) Past Verb Tense Inflectional Rule:

$$\begin{bmatrix} \textit{v-lxm} \\ \text{ORTH } \boxed{\textit{orth}} \\ \text{SS } \boxed{\textit{ss}} \end{bmatrix} \Rightarrow \begin{bmatrix} \text{ORTH } f_{past}(\boxed{\textit{orth}}) \\ \text{SS } \boxed{\textit{ss}} \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{FORM} \quad \text{fin} \\ \text{PRED} \quad - \end{bmatrix} \\ \text{SUBJ } \langle \begin{bmatrix} \text{HEAD}|\text{CASE nom} \end{bmatrix} \rangle \end{bmatrix} \\ \text{CONT r-soa} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The Argument Realization Principle (which is simulated through a set of non-branching rules in my implementation) then applies to the inflected form "left", and coindexes the SUBJ value with the ARG-ST value, and assigns the COMPS to be empty. Therefore the zero complement case of the *non-clausal verb phrase* construction applies, turning "left" into a phrase. Since "left" is a finite, non-inverted phrase with a non-empty SUBJ list the *declarative head-subject clause* then applies, binding the phrase "Kim" as the subject. The resulting phrase unifies with the Root Symbol, and therefore parses. The Root Symbol has the following specification in my implementation:

(46) The Root Symbol:

$$\begin{bmatrix} \text{SS} \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT} \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{IC} \ + \end{bmatrix} \\ \text{SPR} & \langle \ \rangle \\ \text{SUBJ} & \langle \ \rangle \\ \text{COMPS} & \langle \ \rangle \end{bmatrix} \\ \text{CONT} \quad \text{message} \\ \text{STORE} \quad \langle \ \rangle \end{bmatrix} \\ \text{SLASH} \quad \langle \ \rangle \\ \text{WH} \quad \{\} \end{bmatrix} \end{bmatrix}$$

The reason that (43b) "Kim leave" fails to parse is that it does not unify with this specification. The parse chart in Figure 4.2 shows the inflectional rules which apply to "leave". The Base , Subjunctive/Imperative, Infinitival, and Non-3rd-Singular verb inflectional rules all apply to the lexical entry *leave-1* producing the

[1]The LKB's system for inflection is similar in spirit to the function on ORTH shown in (45). It adds a suffix to a string following a set of regular expressions. In cases of words like "leave" which is irregular in its spelling for the past tense, the LKB looks up the correct spelling in a file called irregs.lisp, which must be created by the grammar writer to avoid errors in overregularization.

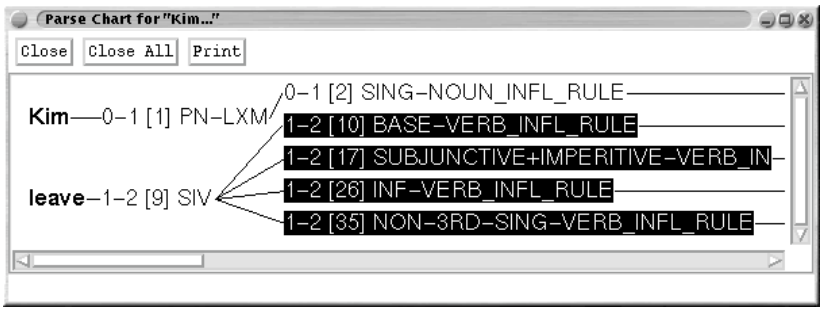


Figure 4.2: The parse chart for "Kim leave."

unchanged output "leave". The word created by the application of the Non-3rd-Singular verb Inflectional Rule blocks a sentence parsing because the application of the *decl-hs-cl* with the 3rd person subject "Kim" fails. The Base Verb and Infinitival Verb Inflectional Rules also block the *decl-hs-cl* construction, because their FORM values are not compatible with *fin*.

The structure for "leave" when Subjunctive/Imperative Verb Inflectional Rule (47) is applied successfully participates with "Kim" in the *decl-hs-cl* construction, but fails to unify with the Root Symbol because the subjuntive adds the stipulation that the verb be [IC –].

(47) Subjunctive/Imperative Verb Inflectional Rule:

$$\begin{bmatrix} \textit{v-lxm} \\ \text{ORTH } \boxed{\textit{orth}} \\ \text{SS } \boxed{\textit{ss}} \end{bmatrix} \Rightarrow \begin{bmatrix} \text{ORTH } \boxed{\textit{orth}} \\ \text{SS } \boxed{\textit{ss}} \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{FORM} \quad \text{fin} \\ \text{IC} \quad - \\ \text{PRED} \quad - \end{bmatrix} \\ \text{SUBJ} \left\langle \left[\text{HEAD}|\text{CASE nom} \right] \right\rangle \end{bmatrix} \\ \text{CONT r-soa} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

4.1.1 Semantic Subcategorization

In this section I will show an example of how the semantic argument selection described in Section 2.2 integrates into parsing. I'll show how the constraints on the MANDATIVE predicate "insist" correctly predict the following data:

(48) a. I insist that Kim leave.

b. Kim did not leave.

c. *I insist that Kim does not leave.

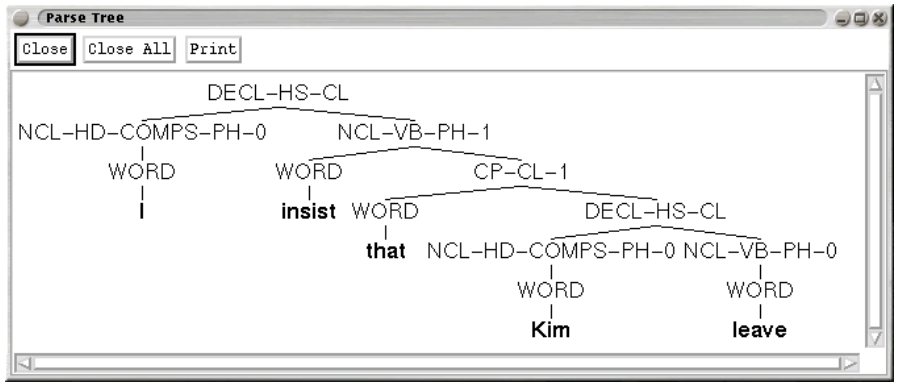


Figure 4.3: The parse for "I insit that Kim leave."

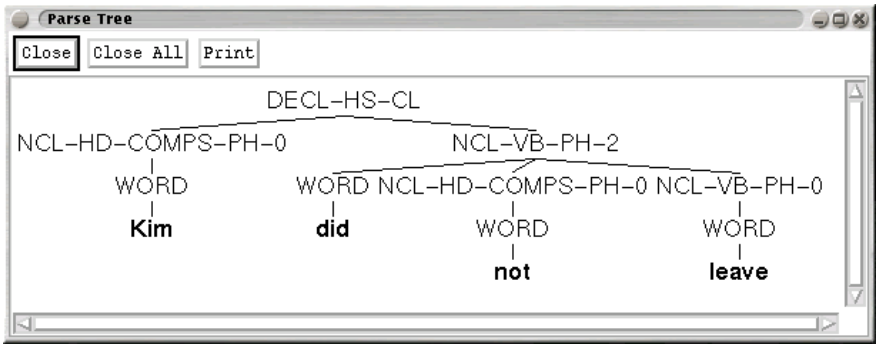


Figure 4.4: The parse for "Kim did not leave."

As with other mandatives, the CONTENT of the complement of "insist" is constrained to be of type *outcome*:

(49)
$$\begin{bmatrix} \textit{v-lxm} \\ \text{ORTH} \langle \text{ "insist" } \rangle \\ \text{SS|LOC|CAT} \begin{bmatrix} \text{ARG-ST} \langle [\ ], \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT} \begin{bmatrix} \text{SUBJ} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \\ \text{CONT outcome} \end{bmatrix} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix}$$

The reason why (48a, shown in Figure 4.3) parses and (43b) does not is because the complementizer phrase headed by "that" allows for the [IC –] complement "Kim leave" formed by the Subjunctive/Imperative Verb Inflectional Rule, whereas the Root Symbol blocks it from standing alone.

The distinction between (48b - shown in Figure 4.4) successfully standing alone as an independent clause and failing to be the complement of (48c) comes about because of the restrictions on the semantic type *outcome*, which constrains its STATE OF AFFAIRS to be an *irrealis-soa*. The inflectional rules that form "did" and "does" in the examples mark them as *realis-soas* and thus the fail to unify with the constraints of the *outcome* required by "insist".

4.2 Focused and Unfocused "Do"

I've incorporated the analysis of English auxiliaries presented in Sag (2000), which presents an approach to the auxiliary system including a treatment of the distribution of unfocused "do":

(50) a. *Kim did leave.

b. Kim DID leave.

c. Did/DID/will Kim leave?

The most significant change in the Sag (2000) analysis from previous HPSG accounts of English auxiliaries is that the feature AUX is treated as an indicator of auxiliary constructions (including finite negation, inversion, and contraction), whereas in the past it had been previously used to distinguish between auxiliary and non-auxiliary verbs. In this analysis non-auxiliary verbs remain [AUX –], but finite auxiliary verbs are left underspecified and instead are realized as [AUX +] or [AUX -] depending on the grammatical construction that they appear in.

Such an analysis allows the classical problem of the distribution of focussed "DO" and unfocused "do" to be solved simply have having two separate lexical entries, with the one for "do" lexically specified as [AUX +]. The interaction of the constraints on auxiliary construction with the auxiliary specification of "do" and the underspecification of "DO" thus models the idiosyncratic distribution.

The constraints on auxiliary constructions are as follow:

(51) a. Finite Verb Phrase (*fin-vb*):

$$\textit{fin-vb} \rightarrow \begin{bmatrix} \text{HD-DTR}|\text{SS}|\text{LOC}|\text{CAT}|\text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AUX } \boxed{1} \\ \text{NEG } \boxed{1} \end{bmatrix} \end{bmatrix}$$

b. Elliptical Verb Phrase (*ellip-vp*):

$$\textit{ellip-vp} \rightarrow \begin{bmatrix} \text{HD-DTR}|\text{SS}|\text{LOC}|\text{CAT}|\text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AUX } + \end{bmatrix} \end{bmatrix}$$

c. Subject Auxiliary Inversion Phrase (*sai-ph*):

$$\textit{sia-ph} \rightarrow \begin{bmatrix} \text{HD-DTR}|\text{SS}|\text{LOC}|\text{CAT}|\text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AUX } + \\ \text{INV } + \end{bmatrix} \end{bmatrix}$$

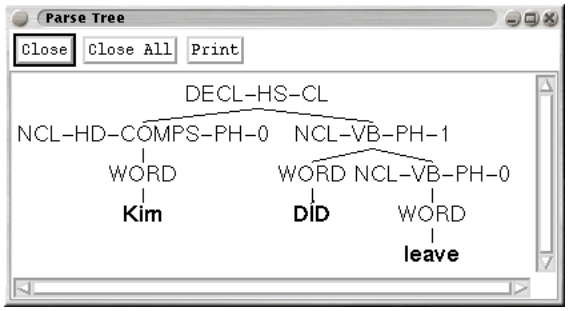


Figure 4.5: The parse for "Kim DID leave."

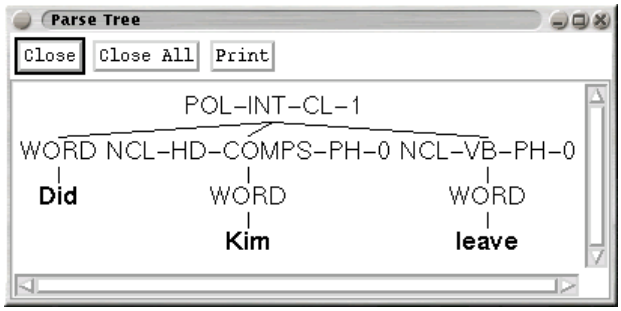


Figure 4.6: The parse for "Did Kim leave?"

Verb phrase (*vb-ph*) constructions are treated as auxiliary constructions if and only if their head is negated. "DO" can participate in either a standard ([AUX –]) VP construction (as in 50b, which is shown in Figure 4.5) or a finite negation auxiliary construction, whereas "do" must only participate in the auxiliary construction, and is blocked otherwise (thus predicting the ungrammaticality of 50a).

Inverted constructions like (50c - shown in Figure 4.6) are always auxiliary constructions, and thus both the focused and unfocused lexical entries can participate in it. Because of the [AUX +] constraint on the construction, no unintended parses will be generated by the non-auxiliary use of "DO" which is used in (50b). Constructions which use the *wh*-complementizer "whether" block inverted constructions as its complement:

(52) a. I wonder if/whether Kim left.

b. *I wonder if/whether did Kim leave.

This prediction is made simply by specifying in the lexical entry for "whether" that its complement be [INV –]:

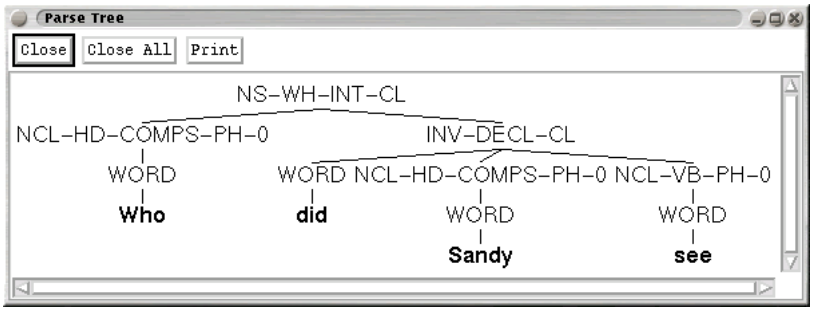


Figure 4.7: The parse for "Who did Sandy see?"

(53)
$$\begin{bmatrix} \text{ORTH} & \langle \text{ "whether" } \rangle \\ \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{comp} \\ \text{IC} & - \\ \text{VFORM} & \boxed{2} \end{bmatrix} \\ \text{ARG-ST} & \left\langle \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{INV} & - \\ \text{VFORM} & \boxed{2} \end{bmatrix} \\ \text{CONT} & \boxed{1} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \textit{question} \\ \text{PARAMS} & \{\ \} \\ \text{PROP} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

4.3 Non-subject *Wh*-Interrogatives

(54) a. Who did/will Sandy see?

b. Who did/will Sandy think she saw?

The parse for "Who did Sandy see?" is shown in Figure 4.7. The complement of "see" is marked as type *gap-ss*, and therefore not canonically realized in a head complement phrase. Instead of being realized by that construction, its LOCAL value is added to its SLASH list, and amalgamated onto the SLASH list of the verb "see" because it appears on the verb's argument structure. The SLASH of "see" is then amalgamated into the SLASH of "did". Note that this passing up of SLASH values can happen for at an arbitrary depth of embedding. Figure 4.8 shows a single level of embedding.

The Non-subject *Wh*-interrogative Clause (*ns-wh-int-cl*) construction inherits from the Head-Filler Phrase (*hd-fill-ph*). Therefore the filler daughter "who" has

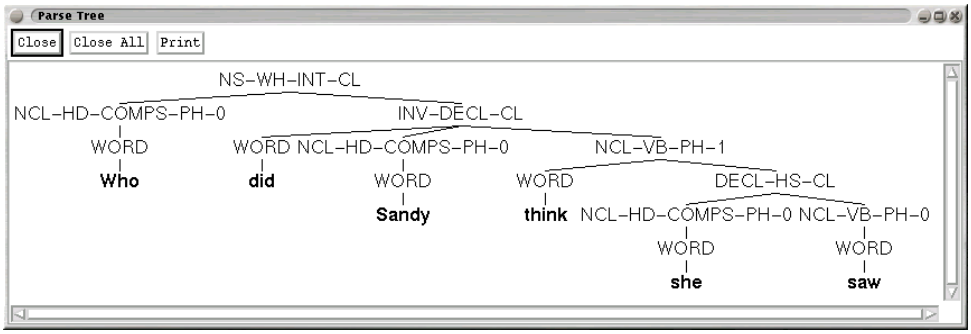


Figure 4.8: The parse for "Who did Sandy think she saw?"

its LOCAL value identified with the LOCAL value of the complement of "see". Thus the semantic index of "who" becomes associated with the SEEN argument of the *see* relation.

The process of extraction and amalgamation is treated through two sets of non-branching pumping rules in my implementation. The first are the STORE Amalgamation rules which explicitly stitch together the STORE values of the arguments on lexical-signs with varying lengths of ARG-ST, and assign an empty store to items with no arguments:

```
store_amalg-0 := inflected-lexeme+store_amalg &
[ SS [ LOC [ STORE <! !>,
             CAT.ARG-ST < > ] ] ].

store_amalg-1 := inflected-lexeme+store_amalg &
[ SS [ LOC [ STORE #1,
             CAT.ARG-ST < [ LOC.STORE #1 ] > ] ] ].


store_amalg-2 := inflected-lexeme+store_amalg &
[ SS [ LOC [ STORE [ LIST #first,
                     LAST #last ],
             CAT.ARG-ST < [ LOC.STORE [ LIST #first,
                                        LAST #second ] ],

                          [ LOC.STORE [ LIST #second,
                                        LAST #last ] ] > ] ] ].


store_amalg-3 := inflected-lexeme+store_amalg &
[ SS [ LOC [ STORE [ LIST #first,
                     LAST #last ],
```

```
            CAT.ARG-ST < [ LOC.STORE [ LIST #first,
                                       LAST #second ] ],
 [ LOC.STORE [ LIST #second,
               LAST #third ] ], [ LOC.STORE [ LIST #third,
                                              LAST #last ] ] > ] ] ].
```

The GAP rules work similarly, but instead of doing difference list stitching the GAP rules mark arguments on the daughter's COMPS list as *gap-ss* (thus making the corresponding ARG-ST item a *gap-ss*) and then manually remove the non-canonical arguments from the mother's COMPS list:

```
canonical-comps := word &
[ SS [ LOC.CAT.COMPS #comps ],
  DTRS < [ SS.LOC.CAT.COMPS #comps ] > ].

gap-1 := word &
[ SS [ LOC.CAT.COMPS < > ],
  DTRS < [ SS.LOC.CAT.COMPS < gap-ss > ] > ].

gap-1-of-2 := word &
[ SS [ LOC.CAT.COMPS < #2 > ],
  DTRS < [ SS.LOC.CAT.COMPS < gap-ss, #2 > ] > ].

gap-2 := word &
[ SS [ LOC.CAT.COMPS < #1 > ],
  DTRS < [ SS.LOC.CAT.COMPS < #1, gap-ss > ] > ].

gap-both := word &
[ SS [ LOC.CAT.COMPS < > ],
  DTRS < [ SS.LOC.CAT.COMPS < gap-ss, gap-ss > ] > ].
```

4.3.1 Distinguished From Topicalization

(55) a. Who did/will Sandy see?

b. *Kim did Sandy see?

The sentence "Kim did Sandy see?" fails as a *ns-wh-int-cl* construction, because the *Wh*-interrogative Clause type stipulates that the filler daughter's WH element must be identified with an item on the PARAMS list. Because "Kim" is [WH { }] it is incompatible with a non-empty WH value.

4.3.2 As Embedded Questions

(56) a. *Who Sandy saw?

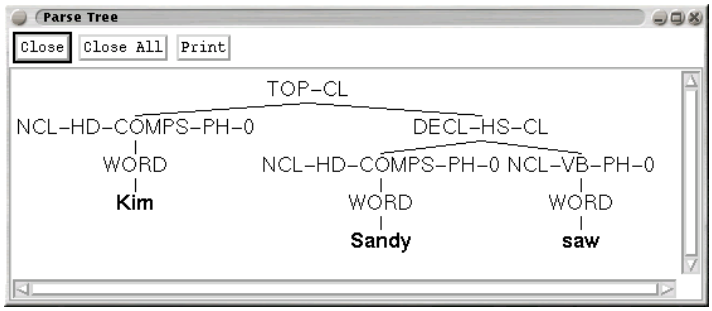


Figure 4.9: The parse for "Kim, Sandy saw."

b. Kim Sandy saw.

c. I wonder who Sandy saw.

d. *I wonder who did/will Sandy see.

The question "Who Sandy saw?" is ruled out because of the constraint on *ns-wh-int-cl* that the head daughter (and thereby the construction itself, by the Generalized Head Feature Principle - shown in (2)) be inverted only if it is also an independent clause:

(57) $$\textit{ns-wh-int-cl} \rightarrow \left[\text{HD-DTR}|\text{SS}|\text{LOC}|\text{CAT}|\text{HEAD}\begin{bmatrix}\text{IC} & \boxed{1}\\ \text{INV} & \boxed{1}\end{bmatrix}\right]$$

Since the constituent "Sandy saw" would only be formed through the Declarative Head-Subject Clause (*decl-hs-cl*) construction as in the topicalized example "Kim Sandy saw" (see Figure 4.9), it is marked [IC +] but [INV –]. Therefore "Sandy saw" fails to unify as the head daughter of a *ns-wh-int-cl*, because it can be built only by the *decl-hs-cl* construction which requires that the clause be [INV –].

The constraint on *ns-wh-int-cl* also distinguished between (56c) and (56d), because the lexical entry for "wonder" constrains its complement to be [IC –], and thus must be uninverted.

4.4 Subject *Wh*-Interrogatives

An example of a *subject wh-interrogative* construction is shown in Figure 4.10. The verb phrase "saw Kim" is formed through the standard application of the *non-clausal verb phrase* construction.[2] After that it is pumped through the *declarative non-subject clause* construction which makes the subject a *gap-ss*, so that it is

[2]Note that my implementation covers an earlier draft of the Ginzburg and Sag text which made a distinction between *verb phrase*, *complementizer phrase*, and *non-verbal* head-complement constructions, rather than the *finite* vs. *non-finite* head complement constructions in the current version.

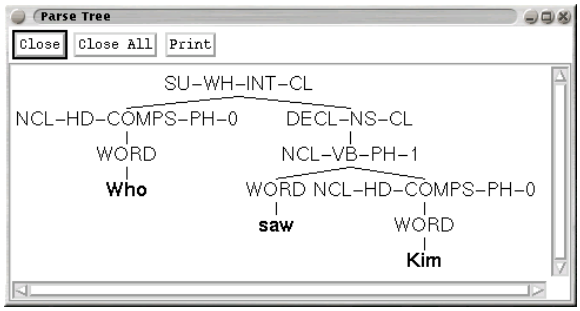


Figure 4.10: The parse for "Who saw Kim?"

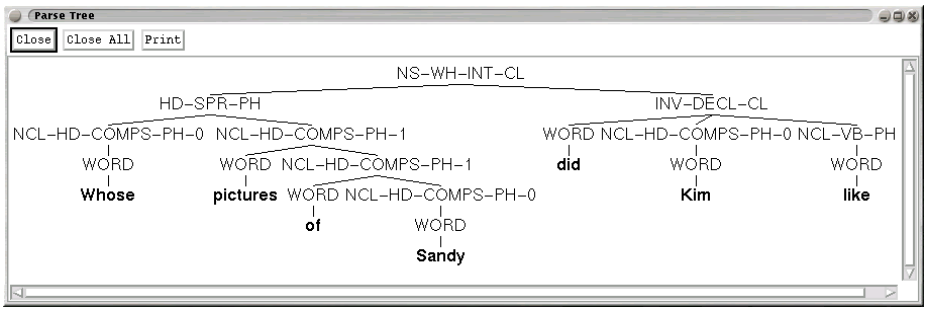


Figure 4.11: The parse for "Whose pictures of Sandy did Kim like?"

treated as extracted for reasons explained in section 2.5.3. The *decl-ns-cl* also produces a proposition which feeds the question formed by the *subject wh-interrogative* construction.

The constraints on *su-wh-int-cl* and its parent *wh-int-cl* identify the LOC value of the *wh*-word "who" with the subject of the declarative clause, thus matching its index to the correct semantic role.

4.5 Sensitivity to the Presence of *Wh*-Words

4.5.1 Pied Piping

The constraints contributed by the WH-Amalgamation Constraint (29) correctly predict that a *wh*-word can be properly contained within the filler daughter of a *wh-interrogative-clause*. This sensitivity to the presence of a *wh*-word is illustrated by so called "pied piping" examples like the following:

(58) a. Whose books did Kim read?

b. Whose pictures of Sandy did Kim like?

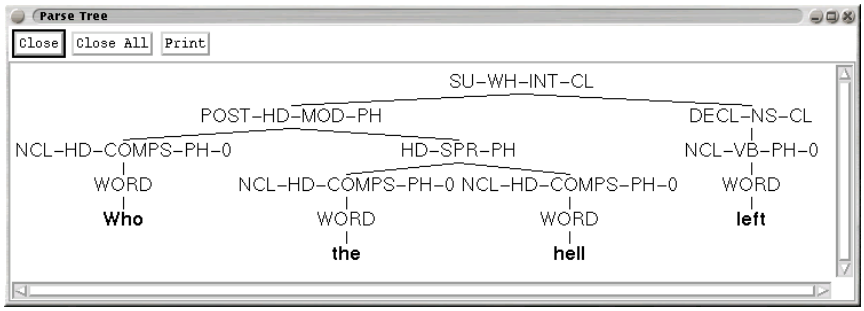


Figure 4.12: The parse for "Who the hell left?"

c. *Sandy's pictures of whom did Kim like?

d. I wonder whose pictures of Sandy Kim likes.

e. *I wonder Sandy's pictures of whom Kim likes.

The distribution of WH specified elements is further constrained by the WH-Subject Prohibition (59) and the WH-Constraint (60):

(59) WH-Subject Prohibition (WHSP):
$word \Rightarrow \left[\text{SS|LOC|CAT|SUBJ list}\left(\left[\text{WH } \{\}\right]\right)\right]$

(60) WH-Constraint (WHC):
Any non-initial element of a lexeme's ARG-ST list must be [WH {}].

These constraints are formulated in such a way that the specifier of a common noun like "picture" or "books" contributes its WH value to the noun, but a common noun's complements do not. Hence predicting the distinction between (58b - shown in Figure 4.11) and (58c).

4.5.2 "The Hell" Examples

The modifier "the hell" is sensitive to the presence of *wh*-words. As well. It's perfectly mellifluous (if a bit base) to utter (61a - Figure 4.12), but is decidedly ungrammatical to say (61b).

(61) a. Who the hell left?

b. *Did Kim the hell leave?

This fact is straightforwardly captured by creating a lexical entry for the modifier "hell" indicating that the thing it modifies must have a *parameter* as an element in its WH value:

(62)
$$\begin{bmatrix} \text{ORTH} \langle \text{ “hell” } \rangle \\ \text{SS|LOC|CAT|HEAD} \begin{bmatrix} \textit{noun} \\ \text{MOD} \langle \left[\text{WH} \langle \text{ param } \rangle \right] \rangle \end{bmatrix} \end{bmatrix}$$

Furthermore examples like (63a) and (63b) are ruled out by the constraint on ARG-ST given by (60) that all non-initial elements be [WH $<>$].[3]

(63) a. *Kim read WHAT the hell?

b. *Who saw WHAT the hell?

Further, Ginzburg and Sag indicate that the WH value on *wh*-words is optional (though the STORE still contains a parameter in interrogative *wh*-words regardless of whether it is coindexed with the WH value). This leads to multiple lexical entries in my implementation. Therefore, the entry for “who” which acts as the initial argument of a verb is:

(64)
$$\begin{bmatrix} \text{ORTH} \langle \text{ “who” } \rangle \\ \text{SS} \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT|HEAD} \begin{bmatrix} \textit{noun} \\ \text{COMPS} \langle \, \rangle \end{bmatrix} \\ \text{CONT|INDEX } \boxed{i} \\ \text{STORE} \langle \boxed{1} \rangle \end{bmatrix} \\ \text{WH} \langle \boxed{1} \begin{bmatrix} \textit{param} \\ \text{INDEX } \boxed{i} \\ \text{RESTR} \langle \left[\text{R-PERSON } \boxed{i} \right] \rangle \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix}$$

And the entry which is a non-initial argument is:

(65)
$$\begin{bmatrix} \text{ORTH} \langle \text{ “who” } \rangle \\ \text{SS} \begin{bmatrix} \text{LOC} \begin{bmatrix} \text{CAT|HEAD} \begin{bmatrix} \textit{noun} \\ \text{COMPS} \langle \, \rangle \end{bmatrix} \\ \text{CONT|INDEX } \boxed{i} \\ \text{STORE} \langle \begin{bmatrix} \textit{param} \\ \text{INDEX } \boxed{i} \\ \text{RESTR} \langle \left[\text{R-PERSON } \boxed{i} \right] \rangle \end{bmatrix} \rangle \end{bmatrix} \\ \text{WH} \langle \, \rangle \end{bmatrix} \end{bmatrix}$$

[3]This constraint is realized though the non-branching rule which applied the ARP in my implementation, because it's applied through a list type constraint (*list-of-wh-empty-synsems*) on the REST of ARG-ST and empty ARG-ST lists don't have a feature REST. Therefore, since my ARP rules pay attention to whether the ARG-ST is empty, the constraint can be applied when it is not.

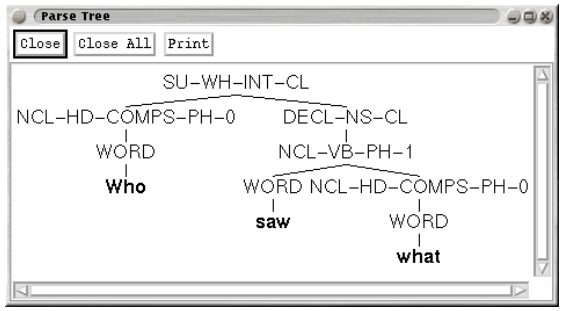


Figure 4.13: The parse for "Who saw what?"

Ginzburg and Sag indicate that the WH Amalgamation Constraint is a default so that lexical entries for *wh*-words which specify a WH value will not inherit their WH value from their arguments. My implementation does not use the default machinery to do this (and it's not clear to me that it would work quite the way outlined in the theory). Instead my implementation has the lexical entries for *wh*-words marked as the intermediate type *inflected-lexeme+amalag* so that the WH Amalgamation Constraint (which is implemented as a non-branching rule, producing things of that type) does not apply to them.

Thus the distribution patterns of the modifying phrase "the hell" shown in (63a) and (63b) is correctly predicted, since it only modifies things with a WH value of $< param >$, which only occurs with initial arguments.

4.6 Multiple *Wh*-Questions

Recall from Section 2.6 that the Store Amalgamation Constraint (36) combines the STORE value for each of a word's arguments into its own STORE, and passes that information up the tree in a mechanism similar to SLASH. Recall also that the *Wh*-Question Retrieval constraint (38) allows any number of items from the STORE to be retrieved at any *headed-interrogative-clause*. Therefore the parameters contributed by multiple *wh*-words in a question are passed up through the tree and retrieved by any question construction.

(66) a. Who saw WHAT?

b. Who wondered who saw WHAT?

The parse for "Who saw what?" is given in Figure 4.13. My implementation builds the following semantics for the question:

```
[QUESTION
 PARAMS: <! [PARAM
             INDEX:  <0>
```

```
            RESTR: <! [ R_PERSON
                        PERSON:  <0> ] !> ],
           [PARAM
            INDEX:  <1>
            RESTR:  <! [ R_THING
                         THING:  <1> ] !> ] !>

PROP:  [PROPOSITION
         SOA:   [R-SOA
                 NUCL:  [R_SEE
                         OBSERVER:  <0>
                         OBSERVED:  <1>]]
         SIT:   (SIT)]]
```

Notice that the the *parameter* contributed by "what" is correctly realized in the PARAMS list of the question, along with the *parameter* of "who".

Since the parameter for "what" in the question "Who wondered who saw what?" can either be retrieved at the level of either of the *subject-wh-interrogative-clauses*, it gets two interpretations. The semantics for the two parses is constructed in an similar fashion as shown in (39) and (40):

```
[QUESTION
 PARAMS:  <!  [PARAM
               INDEX:  <0>
               RESTR: <! [ R_PERSON
                           PERSON:  <0> ] !> ],
              [PARAM
               INDEX:  <1>
               RESTR: <! [ R_THING
                           THING:  <1> ] !> ] !>

 PROP: [PROPOSITION
        SOA: [R-SOA
              NUCL:  [R_WONDER
                      WONDERER: <0>
                      WONDERED: [QUESTION
                                 PARAMS: <! [PARAM
                                             INDEX: <3>
                                             RESTR: <! [R_PERSON
                                                        PERSON: <3>] !> ] !>,
                                 PROP: [PROPOSITION
                                        SOA: [R-SOA
                                              NUCL:  [R_SEE
                                                      OBSERVER: <3>
                                                      OBSERVED: <1>]]
                                        SIT:   (SIT)]]]]
```

```
          SIT:   (SIT)]]
```

The second reading is the following:

```
[QUESTION
 PARAMS: <! [PARAM
             INDEX:  <0>
             RESTR:  <! [ R_PERSON
                          PERSON:  <0> ] !> ] !>
 PROP: [PROPOSITION
        SOA: [R-SOA
              NUCL: [R_WONDER
                     WONDERER: <0>
                     WONDERED: [QUESTION
                                PARAMS: <! [PARAM
                                            INDEX:  <2>
                                            RESTR: <! [R_PERSON
                                                       PERSON:  <2>] !> ],
                                           [PARAM
                                            INDEX:  <3>
                                            RESTR:  <! [R_THING
                                                       THING:  <3>] !> ] !>
                                PROP:  [PROPOSITION
                                        SOA:   [R-SOA
                                                NUCL:  [R_SEE
                                                       OBSERVER:  <2>
                                                       OBSERVED:  <3>]]
                                        SIT:   S ]]]]
       SIT:   S]]
```

The only difference between way that the above semantics are constructed from the way that the semantics in the diagrams in Section 2.6 are constructed is that the non-deterministic retrieval of elements from a set is simulated using lists. This is accomplished by placing an upper bound on the length of the STORE list and ballooning the rules for the subject and non-subject *wh*-interrogative constructions, such that there is one rule for each possible retrieval from a position on the list.

4.7 In-situ *Wh*-Questions

Ginzburg and Sag analyze IN SITU questions – questions of the form

(67) a. Kim saw WHO?

which are often used as 'echo' or 'reference' questions, clarifying or reprising a previous statement. That is, the term 'echo' describes the particular use resulting from

mishearing a previous speech act; this use is marked by a characteristic intonation pattern (focus-associated rise with spreading high tone). Reference questions, by contrast, ask for clarification of the reference of some element in the immediately prior utterance and have a distinct intonation pattern (focus-associated fall with spreading low tone). Reprise uses of *wh*-sentences have generally been neglected by other theories of questions – such uses have been dismissed as 'extra-grammatical' and supposed not to be genuine questions. However, Ginzburg and Sag analyze reprise uses using similar mechanisms to the other question types that they treat. The content of reprise of the following utterance is shown in (69):

(68) a. Prior Utterance: Chris is annoyed with Jan.

b. Interrogative Reprise: Chris is annoyed with WHO(M)?

c. Paraphrase of Reprise: Who did you assert/say that Chris is annoyed with?

(69) Content of Reprise:

$$\left[\begin{array}{ll} \textit{question} & \\ \text{PARAMS} & \left\{\left[\begin{array}{ll}\text{INDEX} & k\\ \text{RESTR} & \{\textit{person}(k)\}\end{array}\right]\right\} \\ \text{PROP} & \left[\begin{array}{ll}\text{SIT} & s_1\\ \text{SOA} & \left[\begin{array}{ll}\text{QUANTS} & \langle\ \rangle\\ \text{NUCL} & \left[\begin{array}{ll}\textit{assert-rel} & \\ \text{UTTERER} & i\\ \text{MSG-ARG} & \left[\begin{array}{ll}\textit{proposition} & \\ \text{SIT} & s_2\\ \text{SOA} & \left[\begin{array}{ll}\text{QUANTS} & \langle\ \rangle\\ \text{NUCL} & \left[\begin{array}{ll}\textit{annoy-rel} & \\ \text{ANNOYED} & j\\ \text{ANNOYANCE} & k\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right] \\ \text{BCKGRND} & \{\textit{prev-spkr}\,(i),\ \textit{named}(j,\textit{Chris})\ \}\end{array}\right]$$

This type of construction, where the a previous statement is reprised and includes a *wh*-constituent to clarify part of the statement, can be formulated as:

(70) *repr-int-cl*:

$$\left[\begin{array}{l}\textsc{store}\ \{\} \\ \textsc{cont}\ \left[\textsc{prop|soa}\ \left[\begin{array}{ll}\textsc{quants} & \langle\ \rangle \\ \textsc{nucl} & \left[\begin{array}{ll} \textit{illoc-rel} & \\ \textsc{utterer} & i \\ \textsc{msg-arg} & \boxed{2}\end{array}\right]\end{array}\right]\right] \\ \textsc{bckgrnd}\ \left\{\begin{array}{l}\textit{prev-spkr}(i),\ \textit{prev-utt}(\boxed{3}), \\ \textit{subst-inst}(\boxed{2},\boxed{3})\end{array}\right\} \cup \boxed{\Sigma}\end{array}\right] \rightarrow \mathbf{H}\left[\begin{array}{ll}\textsc{cont} & \boxed{2} \\ \textsc{bckgrnd} & \boxed{\Sigma}\end{array}\right]$$

subst-inst(X,Y) abbreviates fact that the message X is a substitution-instance of the message Y. *prev-utt*(X) indicates the previous speaker, and *prev-spkr*(X) indicates the previous utterance. Thus, the background contains the information that the speaker is asking for clarification on the previous utterance, rather than introducing a new question.

Reprise questions involve focussed *wh*-words with the intonational patterns described above. These focussed *wh*-words are distinguished in the theory and the implementation using a feature FOC on the *parameter* in lexical entries for those words.

Ginzburg and Sag further give an analysis of non-reprising, or 'direct' in situ questions. In contexts which allow non-reprising in situ *wh*-interrogatives, there always exists the option of using a form that is ALL-FOCUS, namely a sluice:

(i) A: I met someone in the office today.
B: Yeah, who?
B′: Yeah, you met who in the office today?

(ii) A: Dana told me that Chris claimed they found something in the office.
B: Hmm, what?
B′: Hmm, Dana told you that Chris claimed they found what in the office?

These sentences are analyzed as instances of the type *direct-in-situ-interrogatives*, which is defined as:

(71) *dir-is-int-cl*:

$$\left[\textsc{cont|prop}\ \boxed{1}\right] \rightarrow \mathbf{H}\left[\textsc{cont}\ \boxed{1}\right]$$

I have implemented both types of in situ questions, and get the same parses as the theory predicts for the following sentences, shown in Figures 4.14 and 4.15:

(72) a. Kim saw WHO? (2 readings)

b. Who wondered what WHO saw? (ambiguous multiple wh or reprise).

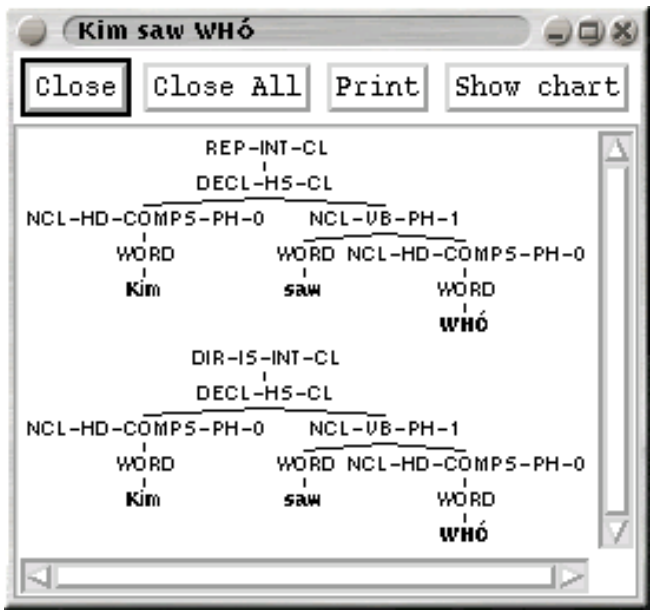


Figure 4.14: The parse for “Kim saw WHO?”

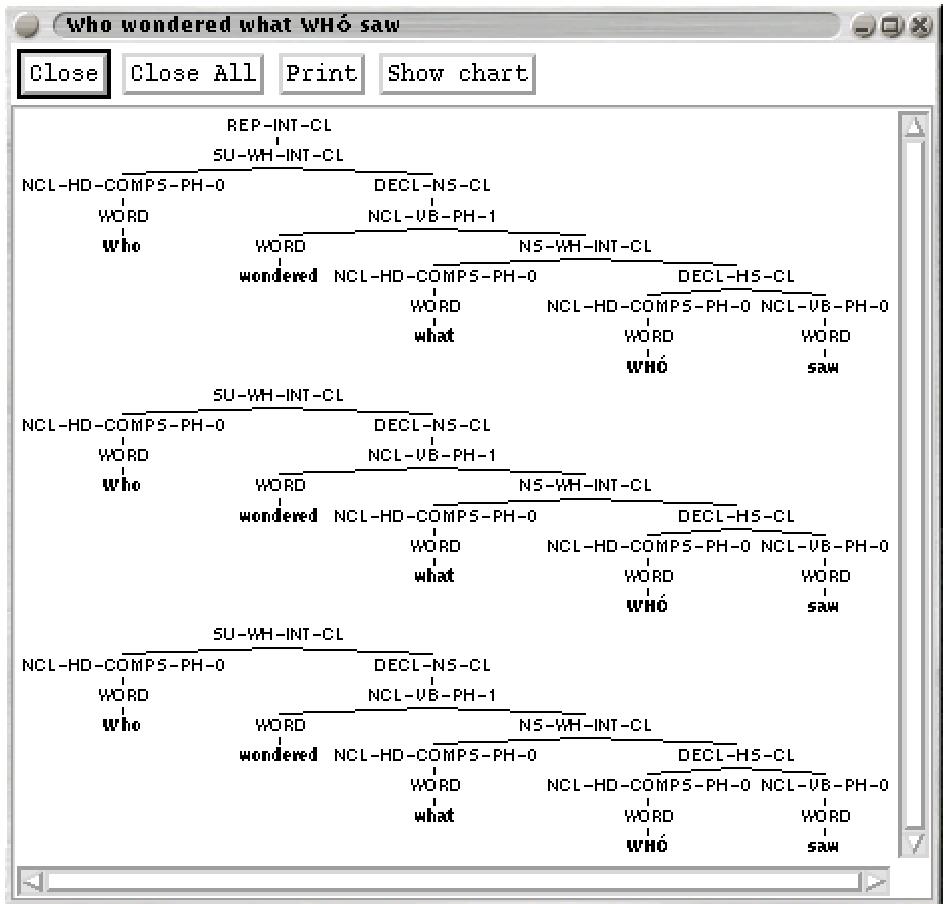


Figure 4.15: The parse for "Who wondered what WHO saw?"

Chapter 5

Appendices

5.1 Types

TYPES		
TYPE	FEATURES/CONSTRAINTS	IST
feat-struc		**top**
sign	[ORTH *list-of-strings* ; SS synsem ; DTRS *list*]	*feat-struc*
synsem	[LOC local ; SLASH *list-of-locals* ; WH *list-of-scope-objs* ; REL *diff-list*]	*feat-struc*
local	[CAT category ; CONT sem-obj ; STORE *list-of-params*]	*feat-struc*
category	[HEAD pos ; SUBJ list-of-synsems ; SPR list-of-synsems ; COMPS list-of-synsems]	*feat-struc*
lex-cat	[ARG-ST list-of-synsems]	*category*
phrase-cat		*category*

SUBTYPES OF SYNSEM		
TYPE	FEATURES/CONSTRAINTS	IST
canon-ss		*synsem*
noncanon-ss		*synsem*
gap-ss	$\begin{bmatrix} \text{LOC} & \boxed{1} \\ \text{SLASH} & \langle ! \ \boxed{1} \ ! \rangle \end{bmatrix}$	*noncanon-ss*
pro-ss	$\begin{bmatrix} \text{SLASH} \ \langle ! \ ! \rangle \\ \text{LOC} \begin{bmatrix} \text{STORE} & \langle ! \ ! \rangle \\ \text{CAT\|HEAD\|CASE} & \text{acc} \end{bmatrix} \end{bmatrix}$	*noncanon-ss*
ellip-ss	$\begin{bmatrix} \text{SLASH} \ \langle ! \ ! \rangle \\ \text{LOC} \begin{bmatrix} \text{CAT} & \text{phrase-cat} \\ \text{STORE} & \langle ! \ ! \rangle \end{bmatrix} \end{bmatrix}$	*noncanon-ss*
neg-ss		*noncanon-ss*

The type *synsem* is divided into two subtypes – *canonical-synsem* and *non-canonical-synsem*. I have included the standard non-canonical-synsem types *gap-ss* and *pro-ss*. I have also added the types *ellip-ss*, which distinguishes elided elements and keeps them from contributing anything to SLASH or STORE, and the type *neg-ss* which prevents "not" from being elided.

LEXICAL AND PHRASAL SIGNS		
TYPE	FEATURES/CONSTRAINTS	IST
phrase	$\begin{bmatrix} \text{SS}\vert\text{LOC}\vert\text{CAT} & \text{phrase-cat} \begin{bmatrix} \text{HEAD} & \text{pos} \\ \text{COMPS} & \langle\,\rangle \end{bmatrix} \end{bmatrix}$	*sign*
lex-sign	$\begin{bmatrix} \text{NEEDS-AFFIX} & \{\text{true, false}\} \\ \text{SS}\vert\text{LOC}\vert\text{CAT} & \text{lex-cat} \end{bmatrix}$	*sign*
lexeme		*lex-sign*
word	$\begin{bmatrix} \text{NEEDS-AFFIX} & \text{false} \\ \text{ORTH} & \boxed{orth} \\ \text{SS} & / \boxed{ss} \begin{bmatrix} \text{LOC}\vert\text{CAT}\vert\text{SUBJ} & \text{list-of-synsems-wh-empty} \end{bmatrix} \\ \text{DTRS} & \langle \begin{bmatrix} \text{ORTH} & \boxed{orth} \\ \text{SS} & / \boxed{ss} \end{bmatrix} \rangle \end{bmatrix}$	*lex-sign*

Ginzburg and Sag make a distinction between *lexeme*, *word*, and *phrase*, where lexical/inflectional rules apply to signs of type *lexeme* and produce signs of type *word*, and the head-complement constructions apply to *words* and produce *phrases* which the remainder of the grammar rules apply to. I've increased the distinction between types of sign by adding three intermediate types between *lexeme* and *word*.

INTERMEDIATE TYPES BETWEEN LEXEME AND WORD		
TYPE	FEATURES/CONSTRAINTS	IST
inflected-lexeme	$\begin{bmatrix} \text{NEEDS-AFFIX} & / \text{ false} \\ \text{SS} & / \boxed{ss} \\ \text{DTRS} & \langle \begin{bmatrix} \text{SS} / \boxed{ss} \end{bmatrix} \rangle \end{bmatrix}$	*lex-sign*
inflected-lexeme+ARP	$\begin{bmatrix} \text{NEEDS-AFFIX} & \text{false} \\ \text{ORTH} & \boxed{orth} \\ \text{SS} & / \boxed{ss} \\ \text{DTRS} & \langle \begin{bmatrix} \text{ORTH} & \boxed{orth} \\ \text{SS} & / \boxed{ss} \end{bmatrix} \rangle \end{bmatrix}$	*lex-sign*
inflected-lexeme+amalg	$\begin{bmatrix} \text{NEEDS-AFFIX} & \text{false} \\ \text{ORTH} & \boxed{orth} \\ \text{SS} & \boxed{ss} \\ \text{DTRS} & \langle \begin{bmatrix} \text{ORTH} & \boxed{orth} \\ \text{SS} & \boxed{ss} \end{bmatrix} \rangle \end{bmatrix}$	*lex-sign*

In my implementation, inflectional rules apply to signs of type *lexeme* and produce signs of type *inflected-lexeme*. Inflected lexemes become signs of type *inflected-lexeme+ARP* with the application of one of the Argument Realization Rules (shown below in Section 5.2). This type is fed into the Amalgamation Rules which produce signs of type *inflected-lexeme+amalg*. Inflected lexemes which have thus been constrained by the Argument Realization Principle and the SLASH, STORE and WH Amalgamation Constraints are then feed into the Gap Rules, which mark arguments as type *gap-ss* and remove them from the COMPS list, thus producing *words*, which the head-complement constructions apply to, as in the Ginzburg and Sag theory.

PARTS OF SPEECH		
TYPE	FEATURES/CONSTRAINTS	IST
pos	$\begin{bmatrix} \text{FORM} & \text{form-cat} \\ \text{PRED} & \{\text{true, false}\} \\ \text{ANA} & \{\text{true, false}\} \end{bmatrix}$	*feat-struc*
mod-type	$\begin{bmatrix} \text{MOD} & \text{*list*} \end{bmatrix}$	*feat-struc*
deg-type	$\begin{bmatrix} \text{DEG} & \{\text{true, false}\} \end{bmatrix}$	*feat-struc*
verbal	$\begin{bmatrix} \text{FORM} & \{\text{fin, inf, inf, base, prp, pfp, pas}\} \\ \text{IC} & \{\text{true, false}\} \\ \text{INF} & \{\text{true, false}\} \end{bmatrix}$	*pos*
verb	$\begin{bmatrix} \text{AUX} & \{\text{true, false}\} \\ \text{INV} & \{\text{true, false}\} \\ \text{NEG} & \{\text{true, false}\} \\ \text{MOD} & \langle\ \rangle \end{bmatrix}$	*verbal, mod-type*
nonverbal		*pos*
comp		*verbal*
topic		*nonverbal*
nominal-cat	$\begin{bmatrix} \text{AGR} & \text{index} \end{bmatrix}$	*topic*
noun	$\begin{bmatrix} \text{CASE} & \{\text{nom, acc}\} \end{bmatrix}$	*nominal-cat, mod-type*
det	$\begin{bmatrix} \text{COUNT} & \{\text{true, false}\} \end{bmatrix}$	*nominal-cat, deg-type*
conj		*nonverbal*
adv		*topic, mod-type*
adj		*nonverbal, mod-type, deg-type*
prep		*nominal-cat*

CLAUSALITY		
TYPE	FEATURES/CONSTRAINTS	IST
clause	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT\|SUBJ} & \text{list-of-noncanon-ss} \\ \text{CONT} & \text{message} \end{bmatrix} \\ \text{WH} & \langle !\ ! \rangle \end{bmatrix} \end{bmatrix}$	*phrase*
non-clause		*phrase*
core-cl	$\begin{bmatrix} \text{SS\|LOC\|CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{FORM} & \{\text{fin, inf}\} \\ \text{MOD} & \langle\ \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$	*clause*
rel-cl	$\begin{bmatrix} \text{SS\|LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{IC false} \\ \text{INV false} \\ \text{MOD} \langle \begin{bmatrix} \text{HEAD} & \text{noun} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix} \\ \text{CONT fact} \end{bmatrix} \end{bmatrix}$	*clause*
decl-cl	$\begin{bmatrix} \text{SS\|LOC\|CONT propositional} \begin{bmatrix} \text{SOA } \boxed{1} \end{bmatrix} \\ \text{HD-DTR\|SS\|LOC\|CONT } \boxed{1} \end{bmatrix}$	*core-cl, hd-ph*
inter-cl	$\begin{bmatrix} \text{SS\|LOC\|CONT question} \end{bmatrix}$	*core-cl*
hd-int-cl		*inter-cl, hd-ph*
canon-int-cl	$\begin{bmatrix} \text{SS\|LOC\|CAT\|HEAD} & \begin{bmatrix} \text{IC} & \boxed{1} \\ \text{INV} & \boxed{1} \end{bmatrix} \end{bmatrix}$	*hd-int-cl*
is-int-cl	$\begin{bmatrix} \text{SS\|LOC\|CAT\|HEAD} & \begin{bmatrix} \text{FORM} & \text{fin} \\ \text{IC} & \text{true} \end{bmatrix} \end{bmatrix}$	*hd-int-cl, hd-only-ph*
wh-int-cl	$\begin{bmatrix} \text{SS\|LOC\|CONT} & \begin{bmatrix} \text{PARAMS\|LIST\|FIRST } \boxed{1} \\ \text{PROP } \boxed{2} \end{bmatrix} \\ \text{HD-DTR\|SS\|LOC\|CONT proposition } \boxed{2} \\ \text{DTRS} \langle \begin{bmatrix} \text{SS\|WH} \langle !\ \boxed{1}\ \text{param}\ ! \rangle \end{bmatrix}, \text{sign} \rangle \end{bmatrix}$	*hd-int-cl, hd-fill-ph*
imp-cl	$\begin{bmatrix} \text{SS\|LOC\|CONT outcome} \end{bmatrix}$	*core-cl*
excl-cl	$\begin{bmatrix} \text{SS\|LOC\|CONT fact} \end{bmatrix}$	*core-cl*

HEADEDNESS		
TYPE	FEATURES/CONSTRAINTS	IST
hd-ph	$\begin{bmatrix} \text{SS} & / \boxed{1} \\ \text{HD-DTR} & \text{sign} \left[\text{SS} / \boxed{1}\right] \end{bmatrix}$	*phrase*
hd-comp-ph	$\begin{bmatrix} \text{HD-DTR} \boxed{1} \\ \text{DTRS} \langle \text{ word } \boxed{1}, \text{ ... } \rangle \end{bmatrix}$	*hd-ph*
nv-hd-comp-ph	$\left[\text{HD-DTR} \left[\text{SS\|LOC\|CAT\|HEAD} \quad \text{nonverbal}\right]\right]$	*hd-comp-ph*
hd-comp-ph-0	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \rangle \\ \text{DTRS} \langle \text{ sign } \rangle \end{bmatrix}$	*nv-hd-comp-ph*
hd-comp-ph-1	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \boxed{1} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right] \rangle \end{bmatrix}$	*nv-hd-comp-ph*
hd-comp-ph-2	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \boxed{1}, \boxed{2} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right], \text{phrase} \left[\text{SS } \boxed{2}\right] \rangle \end{bmatrix}$	*nv-hd-comp-ph*
vb-ph	$\left[\text{HD-DTR\|SS\|LOC\|CAT\|HEAD} \quad \text{verb} \begin{bmatrix} \text{AUX } \boxed{1} \\ \text{NEG } \boxed{1} \end{bmatrix}\right]$	*hd-comp-ph*
vb-ph-0	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \rangle \\ \text{DTRS} \langle \text{ sign } \rangle \end{bmatrix}$	*vb-ph*
vb-ph-1	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \boxed{1} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right] \rangle \end{bmatrix}$	*vb-ph*
vb-ph-2	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \boxed{1}, \boxed{2} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right], \text{phrase} \left[\text{SS } \boxed{2}\right] \rangle \end{bmatrix}$	*vb-ph*
vb-ph-3	$\begin{bmatrix} \text{HD-DTR\|SS\|LOC\|CAT\|COMPS} \langle \boxed{1}, \boxed{2}, \boxed{3} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right], \text{phrase} \left[\text{SS } \boxed{2}\right], \text{phrase} \left[\text{SS } \boxed{3}\right] \rangle \end{bmatrix}$	*vb-ph*

HEADEDNESS		
TYPE	FEATURES/CONSTRAINTS	IST
ellip-vp	$\left[\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{HEAD}\vert\text{AUX} \quad \text{true}\right]$	*hd-comp-ph*
ellip-vp-0	$\left[\begin{array}{l}\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS} \langle \text{ ellip-ss } \rangle \\ \text{DTRS} \langle \text{ sign } \rangle\end{array}\right]$	*ellip-vp*
ellip-vp-1	$\left[\begin{array}{l}\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS} \langle \boxed{1}, \text{ ellip-ss } \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right] \rangle\end{array}\right]$	*ellip-vp*
cp-ph	$\left[\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{HEAD} \quad \text{comp}\right]$	*hd-comp-ph*
cp-ph-1	$\left[\begin{array}{l}\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS} \langle \boxed{1} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right] \rangle\end{array}\right]$	*cp-ph*
cp-ph-2	$\left[\begin{array}{l}\text{HD-DTR}\vert\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS} \langle \boxed{1}, \boxed{2} \rangle \\ \text{DTRS} \langle \text{ sign, phrase} \left[\text{SS } \boxed{1}\right], \text{ phrase} \left[\text{SS } \boxed{2}\right] \rangle\end{array}\right]$	*cp-ph*
hd-subj-ph	$\left[\begin{array}{l}\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{SUBJ} \langle \rangle \\ \text{HD-DTR } \boxed{1} \left[\text{SS}\vert\text{LOC}\vert\text{CAT} \begin{bmatrix}\text{SUBJ} & \langle \boxed{2} \rangle \\ \text{SPR} & \langle \rangle\end{bmatrix}\right] \\ \text{DTRS} \langle \text{ phrase} \left[\text{SS } \boxed{2}\right], \text{ phrase } \boxed{1} \rangle\end{array}\right]$	*hd-ph*
hd-spr-ph	$\left[\begin{array}{l}\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{SPR} \langle \rangle \\ \text{HD-DTR } \boxed{1} \left[\text{SS}\vert\text{LOC}\vert\text{CAT} \left[\text{SPR} \langle \boxed{2} \rangle\right]\right] \\ \text{DTRS} \langle \text{ phrase} \left[\text{SS } \boxed{2}\right], \text{ phrase } \boxed{1} \rangle\end{array}\right]$	*hd-ph, binary-construction*

HEADEDNESS		
TYPE	FEATURES/CONSTRAINTS	IST
sai-ph	[SS|LOC|CAT|SUBJ ⟨ ⟩ HD-DTR [1] [SS|LOC|CAT [HEAD [INV true, AUX true] SUBJ ⟨ [2] ⟩]] DTRS ⟨ word [1], phrase [SS [2]], ... ⟩]	*hd-ph*
sai-ph-0	[HD-DTR|SS|LOC|CAT|COMPS ⟨ ellip-ss ⟩ DTRS ⟨ sign, sign ⟩]	*sai-ph*
sai-ph-1	[HD-DTR|SS|LOC|CAT|COMPS ⟨ [1] ⟩ DTRS ⟨ sign, sign, phrase [SS [1]]⟩]	*sai-ph*
sai-ph-2	[HD-DTR|SS|LOC|CAT|COMPS ⟨ [1], [2] ⟩ DTRS ⟨ sign, sign, phrase [SS [1]], phrase [SS [2]]⟩]	*sai-ph*
hd-adj-ph	[HD-DTR [1] [SS [2]] DTRS ⟨ [1], [SS|LOC|CAT|HEAD [MOD ⟨ [2] ⟩]]⟩]	*hd-ph, non-clause*
hd-fill-ph	[SS [LOC|CAT|HEAD verb SLASH [LIST *rest* LAST *last*]] HD-DTR [1] [SS|SLASH [LIST *cons* ⟨ [2] | *rest*⟩ LAST *null* *last*]] DTRS ⟨ phrase [SS [LOC [2] SLASH ⟨! !⟩]], phrase [1] ⟩]	*hd-ph*
hd-only-ph	[HD-DTR [1] DTRS ⟨ [1] ⟩]	*hd-ph*
non-hd-ph		*phrase*

5.2 Pumping Rules

Argument Realization Principle Rules		
TYPE	FEATURES/CONSTRAINTS	IST
ARP-no-args	$\begin{bmatrix} \text{SS|LOC|CAT} \begin{bmatrix} \text{SPR} \langle\,\rangle \\ \text{SUBJ} \langle\,\rangle \\ \text{COMPS} \langle\,\rangle \\ \text{ARG-ST} \langle\,\rangle \end{bmatrix} \\ \text{DTRS} \langle \text{ inflected-lexeme} \rangle \end{bmatrix}$	*inflected-lexeme+ARP*
ARP-comps-only	$\begin{bmatrix} \text{SS|LOC|CAT} \begin{bmatrix} \text{SPR} \langle\,\rangle \\ \text{SUBJ} \langle\,\rangle \\ \text{COMPS } \boxed{comps} \text{ ne-list} \\ \text{ARG-ST } \boxed{comps} \begin{bmatrix} \text{REST list-of-synsems-wh-none} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} \langle \text{ inflected-lexeme} \rangle \end{bmatrix}$	*inflected-lexeme+ARP*
ARP-spr+comps	$\begin{bmatrix} \text{SS|LOC|CAT} \begin{bmatrix} \text{SPR} \langle \boxed{spr} \rangle \\ \text{SUBJ} \langle\,\rangle \\ \text{COMPS } \boxed{comps} \\ \text{ARG-ST} \begin{bmatrix} \text{FIRST } \boxed{spr} \\ \text{REST list-of-synsems-wh-none } \boxed{comps} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} \langle \text{ inflected-lexeme} \rangle \end{bmatrix}$	*inflected-lexeme+ARP*
ARP-subj+comps	$\begin{bmatrix} \text{SS|LOC|CAT} \begin{bmatrix} \text{SPR} \langle\,\rangle \\ \text{SUBJ} \langle \boxed{subj} \rangle \\ \text{COMPS } \boxed{comps} \\ \text{ARG-ST} \begin{bmatrix} \text{FIRST } \boxed{subj} \\ \text{REST list-of-synsems-wh-none } \boxed{comps} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} \langle \text{ inflected-lexeme} \rangle \end{bmatrix}$	*inflected-lexeme+ARP*

Amalgamation Rules

TYPE	FEATURES/CONSTRAINTS	IST
amalgamation-0	$\left[\begin{array}{l}\text{SS}\left[\begin{array}{l}\text{SLASH}\ \langle !\ !\rangle\\ \text{WH}\ \langle !\ !\rangle\\ \text{LOC}\left[\begin{array}{l}\text{STORE}\ \langle !\ !\rangle\\ \text{CAT|ARG-ST}\ \langle\ \rangle\end{array}\right]\end{array}\right]\\ \text{DTRS}\ \langle\ \text{inflected-lexeme+ARP}\ \rangle\end{array}\right]$	*inflected-lexeme+amalg*
amalgamation-1	$\left[\begin{array}{l}\text{SS}\left[\begin{array}{l}\text{SLASH}\ \boxed{slash}\\ \text{WH}\ \boxed{wh}\\ \text{LOC}\left[\begin{array}{l}\text{STORE}\ \boxed{store}\\ \text{CAT|ARG-ST}\ \langle\left[\begin{array}{l}\text{SLASH}\ \boxed{slash}\\ \text{WH}\ \boxed{wh}\\ \text{LOC}\left[\text{STORE}\ \boxed{store}\right]\end{array}\right]\rangle\end{array}\right]\end{array}\right]\\ \text{DTRS}\ \langle\ \text{inflected-lexeme+ARP}\ \rangle\end{array}\right]$	*inflected-lexeme+amalg*
amalgamation-2	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\left[\begin{array}{l}\text{STORE}\left[\begin{array}{l}\text{LIST}\ \boxed{store{-}1}\\ \text{LAST}\ \boxed{store{-}last}\end{array}\right]\\ \text{CAT|ARG-ST}\ \langle\left[\text{LOC|STORE}\left[\begin{array}{l}\text{LIST}\ \boxed{store{-}1}\\ \text{LAST}\ \boxed{store{-}2}\end{array}\right]\right]\\ \left[\text{LOC|STORE}\left[\begin{array}{l}\text{LIST}\ \boxed{store{-}2}\\ \text{LAST}\ \boxed{store{-}last}\end{array}\right]\right]\rangle\end{array}\right]\right]\\ \text{SS}\left[\begin{array}{l}\text{SLASH}\left[\begin{array}{l}\text{LIST}\ \boxed{slash{-}1}\\ \text{LAST}\ \boxed{slash{-}last}\end{array}\right]\\ \text{LOC|CAT|ARG-ST}\ \langle\left[\text{SLASH}\left[\begin{array}{l}\text{LIST}\ \boxed{slash{-}1}\\ \text{LAST}\ \boxed{slash{-}2}\end{array}\right]\right]\\ \left[\text{SLASH}\left[\begin{array}{l}\text{LIST}\ \boxed{slash{-}2}\\ \text{LAST}\ \boxed{slash{-}last}\end{array}\right]\right]\rangle\end{array}\right]\\ \text{SS}\left[\begin{array}{l}\text{WH}\left[\begin{array}{l}\text{LIST}\ \boxed{wh{-}1}\\ \text{LAST}\ \boxed{wh{-}last}\end{array}\right]\\ \text{LOC|CAT|ARG-ST}\ \langle\left[\text{WH}\left[\begin{array}{l}\text{LIST}\ \boxed{wh{-}1}\\ \text{LAST}\ \boxed{wh{-}2}\end{array}\right]\right]\\ \left[\text{WH}\left[\begin{array}{l}\text{LIST}\ \boxed{wh{-}2}\\ \text{LAST}\ \boxed{wh{-}last}\end{array}\right]\right]\rangle\end{array}\right]\\ \text{DTRS}\ \langle\ \text{inflected-lexeme+ARP}\ \rangle\end{array}\right]$	*inflected-lexeme+amalg*

There are additional Amalgamation Rules for ARGUMENT STRUCTURES of length 3 and 4, but the difference list stitching that they use make them unwieldy to illustrate here.

Gap Rules		
TYPE	FEATURES/CONSTRAINTS	IST
canonical-comps	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \boxed{comps}\right] \\ \text{DTRS}\ \langle\ \text{inflected-lexeme+amalg}\left[\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \boxed{comps}\right]\rangle\end{array}\right]$	*word*
gap-1	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \rangle\right] \\ \text{DTRS}\ \langle\ \text{inflected-lexeme+amalg}\left[\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \text{gap-ss}\ \rangle\right]\rangle\end{array}\right]$	*word*
gap-1-of-2	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \boxed{2}\ \rangle\right] \\ \text{DTRS}\ \langle\ \text{inflected-lexeme+amalg}\left[\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \text{gap-ss},\ \boxed{2}\ \rangle\right]\rangle\end{array}\right]$	*word*
gap-2	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \boxed{1}\ \rangle\right] \\ \text{DTRS}\ \langle\ \text{inflected-lexeme+amalg}\left[\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \boxed{1},\ \text{gap-ss}\ \rangle\right]\rangle\end{array}\right]$	*word*
gap-both	$\left[\begin{array}{l}\text{SS}\left[\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \rangle\right] \\ \text{DTRS}\ \langle\ \text{inflected-lexeme+amalg}\left[\text{SS}\vert\text{LOC}\vert\text{CAT}\vert\text{COMPS}\ \langle\ \text{gap-ss},\ \text{gap-ss}\ \rangle\right]\rangle\end{array}\right]$	*word*

5.3 Semantics

SOME BASIC SEMANTIC TYPES		
TYPE	FEATURES/CONSTRAINTS	IST
sem-obj		*feat-struc*
message		*sem-obj*
propositional	[SOA soa SIT sit]	*message*
proposition	[SOA r-soa]	*propositional*
fact	[SOA r-soa]	*propositional*
outcome	[SOA i-soa]	*propositional*
question	[PARAMS *list-of-params* PROP proposition]	*message*
illoc-rel	[MSG-ARG message]	*reln*
assert-reln	[MSG-ARG proposition]	*illoc-rel*
ask-reln	[MSG-ARG question]	*illoc-rel*
order-reln	[MSG-ARG outcome]	*illoc-rel*
exclaim-reln	[MSG-ARG fact]	*illoc-rel*
soa	[NUCL reln]	*sem-obj*
r-soa		*soa*
i-soa		*soa*

MISC. SEMANTIC TYPES		
TYPE	FEATURES/CONSTRAINTS	IST
null-sem		*sem-obj*
scope-obj		*sem-obj*
g-quant		*scope-obj*
param	$\begin{bmatrix} \text{INDEX} & \text{index} \\ \text{FOC} & \text{boolean} \\ \text{RESTR} & \text{list-of-relns} \end{bmatrix}$	*scope-obj*
sit		*feat-struc*
reln		*feat-struc*

5.4 Lexeme Types

PART OF SPEECH		
TYPE	FEATURES/CONSTRAINTS	IST
v-lxm	$\left[\text{SS}\left[\text{LOC}\left[\text{CAT}\left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{ll}\text{NEG} & \text{false}\\ \text{ANA} & \text{false}\end{array}\right]\\ \text{SPR} & \langle\,\rangle\\ \text{SUBJ} & \langle\,\left[\text{LOC|CAT}\left[\begin{array}{ll}\textit{phrase-cat}\\ \text{HEAD} & \text{noun}\\ \text{SPR} & \langle\,\rangle\\ \text{COMPS} & \langle\,\rangle\end{array}\right]\right\rangle\end{array}\right]\right]\right]\right.$ $\left.\text{WH}\ \langle !\ !\rangle\right]$	*lexeme*
non-auxv-lxm	$\left[\text{SS|LOC|CAT|HEAD}\ \left[\begin{array}{ll}\text{INV} & \text{false}\\ \text{AUX} & \text{false}\end{array}\right]\right]$	*v-lxm*
auxv-lxm	$\left[\text{SS|LOC|CAT}\left[\begin{array}{ll}\text{SUBJ} & \langle\,\boxed{1}\,\rangle\\ \text{ARG-ST} & \langle\,\boxed{1},\left[\text{LOC|CAT}\left[\begin{array}{l}\text{SUBJ}\ \langle\,\boxed{1}\left[\text{LOC|STORE}\ \langle !\ !\rangle\right]\rangle\\ \text{COMPS}\ \langle\,\rangle\end{array}\right]\right]\rangle\end{array}\right]\right]$	*v-lxm*
n-lxm	$\left[\text{SS|LOC}\left[\text{CAT}\left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{l}\text{FORM / normal}\\ \text{ANA / false}\\ \text{AGR / ref-index}\\ \text{MOD / }\langle\,\rangle\end{array}\right]\\ \text{SUBJ} & \langle\,\rangle\\ \text{SPR} & \text{/}\ \langle\,\rangle\\ \text{COMPS} & \text{/}\ \langle\,\rangle\end{array}\right]\right]\right]$	*lexeme*

PART OF SPEECH		
TYPE	FEATURES/CONSTRAINTS	IST
pron-lxm	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC\|CAT\|HEAD} & \begin{bmatrix} \text{FORM} & / \text{ normal} \\ \text{ANA} & / \text{ false} \end{bmatrix} \\ \text{WH} & \langle ! \ ! \rangle \end{bmatrix} \end{bmatrix}$	*n-lxm, no-args*
cn-lxm	$\begin{bmatrix} \text{SS\|LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR\|PERNUM} & \boxed{agr} \end{bmatrix} \\ \text{SPR} & \langle \begin{bmatrix} \text{LOC\|CAT} & \begin{bmatrix} \textit{phrase-cat} & \\ \text{HEAD} & \begin{bmatrix} \textit{det} & \\ \text{AGR\|PERNUM} & \boxed{agr} \\ \text{COUNT} & / \text{ true} \end{bmatrix} \\ \text{SPR} & \langle \ \rangle \end{bmatrix} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$	*n-lxm*
pn-lxm	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR\|PERNUM} & \text{3sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{WH} & \langle ! \ ! \rangle \end{bmatrix} \end{bmatrix}$	*n-lxm, no-args*
const-lxm		*lexeme*
det-lxm	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \text{det} \\ \text{SUBJ} & \langle \ \rangle \\ \text{SPR} & / \langle \ \rangle \\ \text{COMPS} & \langle \ \rangle \end{bmatrix} \\ \text{CONT} & \text{null-sem} \end{bmatrix} \end{bmatrix} \end{bmatrix}$	*const-lxm*
p-lxm	$\begin{bmatrix} \text{SS\|LOC\|CAT} & \begin{bmatrix} \text{HEAD} & \text{prep} \\ \text{SPR} & \langle \ \rangle \\ \text{COMPS} & \langle \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{phrase-cat} & \\ \text{HEAD} & \text{noun} \\ \text{SPR} & \langle \ \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix}$	*const-lxm*

PART OF SPEECH		
TYPE	FEATURES/CONSTRAINTS	IST
mkp-lxm	$\begin{bmatrix} \text{SS|LOC|CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{PRED} & \text{true} \end{bmatrix} \\ \text{SUBJ} & \langle\ \text{synsem}\ \rangle \end{bmatrix} \end{bmatrix}$	*p-lxm*
pdp-lxm	$\begin{bmatrix} \text{SS|LOC|CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{PRED} & \text{false} \end{bmatrix} \\ \text{SUBJ} & \langle\ \rangle \end{bmatrix} \end{bmatrix}$	*p-lxm*
comp-lxm	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC|CAT} & \begin{bmatrix} \text{HEAD} & \text{comp} \\ \text{SUBJ} & \langle\ \rangle \\ \text{SPR} & \langle\ \rangle \end{bmatrix} \\ \text{WH} & \langle !\ ! \rangle \end{bmatrix} \end{bmatrix}$	*const-lxm*
adj-lxm	$\begin{bmatrix} \text{SS|LOC|CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{FORM} & \text{normal} \\ \text{PRED} & \text{true} \\ \text{MOD} & \langle \begin{bmatrix} \text{LOC|CAT|HEAD} & \text{noun} \end{bmatrix} \rangle \end{bmatrix} \\ \text{SUBJ} & / \langle\ \rangle \\ \text{SPR} & / \langle\ \rangle \\ \text{COMPS} & / \langle\ \rangle \end{bmatrix} \end{bmatrix}$	*const-lxm*
adv-lxm	$\begin{bmatrix} \text{SS|LOC|CAT} & \begin{bmatrix} \text{HEAD} & \text{adv} \\ \text{SUBJ} & \langle\ \rangle \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \end{bmatrix}$	*const-lxm*

ARGUMENT SELECTION		
TYPE	FEATURES/CONSTRAINTS	IST
no-args	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \rangle\right]$	*lexeme*
intr	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem, ...}\ \rangle\right]$	*lexeme*
str-intr	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem}\ \rangle\right]$	*intr*
intr-xcomp	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem, synsem}\ \rangle\right]$	*intr*
s-con	$\left[\text{SS|LOC|CAT|ARG-ST } \langle \left[\text{LOC } \left[\text{CONT } \left[\text{INDEX } \boxed{1}\right]\right]\right] \left[\text{LOC|CAT } \begin{bmatrix}\text{SPR} & \langle \left[\text{LOC } \left[\text{CONT } \left[\text{INDEX } \boxed{1}\right]\right]\right] \rangle \\ \text{COMPS} & \langle\ \rangle\end{bmatrix}\right]\rangle\right]$	*intr-xcomp*
tran	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem, synsem, ...}\ \rangle\right]$	*lexeme*
str-tr	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem, synsem}\ \rangle\right]$	*tran*
tran-xcomp	$\left[\text{SS|LOC|CAT|ARG-ST } \langle\ \text{synsem, synsem, synsem}\ \rangle\right]$	*tran*

SUBTYPES OF PART-OF-SPEECH AND ARG-SELECTION		
TYPE	FEATURES/CONSTRAINTS	IST
siv		*non-auxv-lxm, str-intr*
scv		*non-auxv-lxm, s-con*
srv		*non-auxv-lxm, s-rais*
prep-arg	$\left[\text{SS|LOC|CAT|ARG-ST} \quad \langle\ \text{synsem}, \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{prep} \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}\right] \rangle\right]$	*lexeme*
NP-trans-arg	$\left[\text{SS|LOC|CAT|ARG-ST} \quad \langle\ \text{synsem}, \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{noun} \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}\right] \rangle\right]$	*tran*
ditrans-arg	$\left[\text{SS|LOC|CAT|ARG-ST} \quad \langle\ \text{synsem, synsem, synsem}\ \rangle\right]$	*lexeme*
dt-arg	$\left[\text{SS|LOC|CAT|ARG-ST} \ \langle\ \text{synsem}, \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{noun} \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}\right] \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{noun} \\ \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}\right] \rangle\right]$	*ditrans-arg*
ptv-arg	$\left[\text{SS|LOC|CAT|ARG-ST} \ \langle\ \text{synsem}, \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{noun} \\ \text{SPR} & \langle\ \rangle \end{bmatrix}\right] \left[\text{LOC|CAT} \quad \begin{bmatrix} \textit{phrase-cat} \\ \text{HEAD} & \text{prep} \\ \text{SPR} & \langle\ \rangle \end{bmatrix}\right] \rangle\right]$	*ditrans-arg*
s-rais	$\left[\text{SS|LOC|CAT|ARG-ST} \quad \langle\ \boxed{1}, \left[\text{LOC|CAT} \begin{bmatrix} \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}\right] \rangle\right]$	*lexeme*

SUBTYPES OF PART-OF-SPEECH AND ARG-SELECTION		
TYPE	FEATURES/CONSTRAINTS	IST
piv		*non-auxv-lxm, prep-arg*
stv		*non-auxv-lxm, trans-arg*
ctv	$\left[\text{SS\|LOC\|CAT\|ARG-ST} \quad \langle \text{synsem}, \left[\text{LOC} \left[\begin{array}{ll}\text{CAT} & \left[\begin{array}{ll}\text{HEAD} & \left[\text{FORM} \quad \text{fin}\right] \\ \text{SPR} & \langle\,\rangle \\ \text{COMPS} & \langle\,\rangle\end{array}\right] \\ \text{CONT} & \text{proposition}\end{array}\right]\right]\rangle\right]$	*non-auxv-lxm, str-tr*
dtv		*non-auxv-lxm, dt-arg*
ptv		*non-auxv-lxm, ptv-arg*
sia		*adj-lxm, str-intr*
pia		*adj-lxm, prep-arg*
sta		*adj-lxm, str-tr*
sra		*adj-lxm, s-rais*
sca		*adj-lxm, s-con*

5.5 Root Symbol

TYPES		
TYPE	FEATURES/CONSTRAINTS	IST
root	$\begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{IC} & \text{true} \end{bmatrix} \\ \text{SPR} & \langle\ \rangle \\ \text{SUBJ} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \\ \text{CONT} & \text{message} \\ \text{STORE} & \langle !\ ! \rangle \end{bmatrix} \\ \text{SLASH} & \langle !\ ! \rangle \\ \text{WH} & \langle !\ ! \rangle \end{bmatrix} \end{bmatrix}$	*phrase*

Bibliography

Bouma, Gosse, Rob Malouf, and Ivan Sag. in press. Satisfying constraints on extraction and adjunction. Natural Language and Linguitic Theory .

Copestake, Ann. in preparation. Implementing typed feature structure grammars. Stanford University: CSLI.

Ginzburg, Jonathan, and Ivan Sag. 2000. English interrogative constructions. Stanford University: CSLI. forthcoming.

Hukari, Tomas, and Robert Levine. 1995. Adjunct extraction. Journal of Linguitics 31.195–226.

Lascarides, Alex, and Ann Copestake. 1999. Default representation in constraint-based frameworks. Computational Linguitics 25.55–106.

Pollard, Carl, and Ivan Sag. 1987. Information-based syntax and semantics. Stanford University: CSLI.

Sag, Ivan. 1997. English relative clause constructions. Journal of Linguistics .

——. 2000. Rules and exceptions in the English auxiliary system. Stanford University, ms.

——, and Tom Wasow. 1999. Syntactic theory: A formal introduction. Stanford University: CSLI.