Structure-Sharing in Lexical Representation

Daniel Flickinger, Carl Pollard, and Thomas Wasow
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA. 94303, USA

Abstract

The lexicon now plays a central role in our implementation of a Head-driven Phrase Structure Grammar (HPSG), given the massive relocation into the lexicon of linguistic information that was carried by the phrase structure rules in the old GPSG system. HPSG's grammar contains fewer than twenty (very general) rules; its predecessor required over 350 to achieve roughly the same coverage. This simplification of the grammar is made possible by an enrichment of the structure and content of lexical entries, using both inheritance mechanisms and lexical rules to represent the linguistic information in a general and efficient form. We will argue that our mechanisms for structure-sharing not only provide the ability to express important linguistic generalizations about the lexicon, but also make possible an efficient, readily modifiable implementation that we find quite adequate for continuing development of a large natural language system.

1. Introduction

The project we refer to as HPSG is the current phase of an ongoing effort at Hewlett-Packard Laboratories to develop an English language understanding system which implements current work in theoretical linguistics. Incorporating innovations in the areas of lexicon, grammar, parser, and semantics, HPSG is the successor to the GPSG system reported on at the 1982 ACL meeting[1] Like the GPSG system, the current implementation is based on the linguistic theory known as Generalized Phrase Structure Grammar,[2] though incorporating insights from Carl Pollard's recent work on Head Grammars[3] which lead us to employ a richer lexicon and a significantly smaller grammar. We report here on the structure of our lexicon, the mechanisms used in its representation, and the resulting sharp decrease in the number of phrase structure rules needed.[4]

[1] Gawron, et al. (1982).

[2] Gazdar, Klein, Pullum, and Sag (1985).

[3] Pollard (1984).

2. Mechanisms employed

We employ three types of mechanisms for structure-sharing in our representation of the lexicon for the HPSG system: inheritance, lexical rules, and an operation to create nouns from ordinary database entities. In order to present a detailed description of these mechanisms, we offer a brief sketch of the representation language in which the lexicon is constructed. This language is a descendant of FRL and is currently under development at HP Labs.[5] Those readers familiar with frame-based knowledge representation will not need the review provided in the next section.

2.1. The representation language

The basic data structures of the representation language are frames with slots, superficially analogous to Pascal records with fields. However, frames are linked together by means of class and inheritance links, such that when a particular frame F0 is an instance or subclass of a more general frame F1, information stored in F1 can be considered part of the description of F0. For example, our lexicon database contains frames specifying properties of classes of words, such as the VERB class, which numbers among its subclasses BASE and FINITE. Having specified on the VERB class frame that all verbs have the value V for the MAJOR feature, this value does not have to be stipulated again on each of the subclasses, since the information will be inherited by each subclass. This class linkage is transitive, so information can be inherited through any number of intermediate frames. Thus any instance of the FINITE class will inherit the value FINITE for the feature FORM directly from the FINITE class frame, and will also inherit the value V for the MAJOR feature indirectly from the VERB class frame.

[4] Significant contributions to the basic design of this lexicon were made by Jean Mark Gawron and Elizabeth Ann Paulson, members of the Natural Language Project when the work on HPSG was begun in 1983. We are also indebted to Geoffrey K. Pullum, a consultant on the project, for valuable assistance in the writing of this paper.

[5] For a description of this language, see Rosenberg (1983).

Of course, to make good use of this information, one must be able to exercise some degree of control over the methods of access to the information stored in a hierarchical structure of this sort, to allow for subregularities and exceptions, among other things. The language provides two distinct modes of inheritance; the one we will call the *normal* mode, the second the *complete* mode. When using the normal mode to collect inheritable information, one starts with the frame in question and runs up the inheritance links in the hierarchy, stopping when the first actual value for the relevant slot is found. The complete mode of inheritance simply involves collecting all available values for the relevant slot, beginning with the particular frame and going all the way up to the top of the hierarchy. We illustrated the complete mode above in describing the feature values that a finite verb like *works* would inherit. To illustrate a use of the normal mode, we note that the VERB class will specify that the CASE of the ordinary verb's subject is OBJECTIVE, as in *Mary wanted him to work.* (not *Mary wanted he to work.*). But the subjects of finite verbs have nominative case, so in the FINITE class frame we stipulate (CASE NOMINATIVE) for the subject. If we used the complete mode of inheritance in determining the case for a finite verb's subject, we would have a contradiction, but by using the normal mode, we find the more local (NOMINATIVE) value for CASE first, and stop. In short, when normal mode inheritance is employed, locally declared values override values inherited from higher up the hierarchy.

The third and final property of the representation language that is crucial to our characterization of the lexicon is the ability of a frame to inherit information along more than one inheritance path. For example, the lexical frame for the finite verb *works* is not only an instance of FINITE (a subclass of VERB), but also an instance of INTRANSITIVE, from which it inherits the information that it requires a subject and nothing else in order to make a complete sentence. This ability to establish multiple inheritance links for a frame proves to be a powerful tool, as we will illustrate further below.

2.2. Inheritance

Having presented some of the tools for inheritance, let us now see how and why this mechanism proves useful for representing the information about the lexicon that is needed to parse sentences of English.[6] We make use of the frame-based representation language to impose a rich hierarchical structure on our lexicon, distributing throughout this structure the information needed to describe the particular lexical items, so that each distinct property which holds for a given class of words need only be stated once. We do this by defining generic lexical frames for grammatical categories at several levels of abstraction, beginning at the top with a generic WORD frame, then dividing and subdividing into ever more specific categories until we hit bottom in frames for actual English words. An example will help clarify the way in which we use this first basic mechanism for sharing structure in representing lexical information.

We employ, among others, generic (class) frames for VERB, TRANSITIVE, and AUXILIARY, each containing just that information which is the default for its instances. The AUXILIARY frame stores the fact that in general auxiliary verbs have as their complement a verb phrase in base form (e.g. the base VP *be a manager* in *will be a manager*). One of the exceptions to this generalization is the auxiliary verb *have* as in *have been consultants*, where the complement VP must be a past participle rather than in base form. The exception is handled by specifying the past participle in the COMPLEMENT slot for the HAVE frame, then being sure to use the *normal* mode of inheritance when asking for the syntactic form of a verb's complement.

To illustrate the use we make of the *complete* mode of inheritance, we first note that we follow most current syntactic theories in assuming that a syntactic category is composed (in part) of a set of syntactic features each specified for one or more out of a range of permitted values. So the category to which the auxiliary verb *has* belongs can be specified (in part) by the following set of feature-value pairs:

```
[(MAJOR V) (TENSE PRES)
 (AGREEMENT 3RD-SING)
 (CONTROL SSR) (AUX PLUS)]
```

Now if we have included among our generic frames one for the category of present-tense verbs, and an instance of this class for third-person-singular present-tense verbs, then we can distribute the structure given in the list above in the following way. We specify that the generic VERB frame includes in its features (MAJOR V), that the PRESENT-TENSE frame includes (TENSE PRES), that the THIRD-SING frame includes (AGREEMENT 3RD-SING), that the SUBJECT-RAISE frame includes (CONTROL SSR), and the AUXILIARY frame includes (AUX PLUS). Then we can avoid saying anything explicitly about features in the frame for the auxiliary verb *have*; we need only make sure it is an instance of the three rather unrelated frames THIRD-SING, SUBJECT-RAISE, and AUXILIARY. As long as we use the *complete* mode

[6] The use of inheritance for efficiently representing information about the lexicon is by no means an innovation of ours; see Bobrow and Webber (1980a,b) for a description of an implementation making central use of inheritance. However, we believe that the powerful tools for inheritance (particularly that of multiple inheritance) provided by the representation language we use have allowed us to give an unusually precise, easily modifiable characterization of the generic lexicon, one which greatly facilitates our continuing efforts to reduce the number of necessary phrase structure rules).

of inheritance when asking for the value of the FEATURES slot for the HAVE frame, we will collect the five feature-value pairs listed above, by following the three inheritance path links up through the hierarchy, collecting all of the values that we find.

2.3. Lexical rules

The second principal mechanism we employ for structure-sharing is one familiar to linguists: the lexical redundancy rule[7], which we use to capture both inflectional and derivational regularities among lexical entries. In our current implementation, we have made the lexical rules directional, in each case defining one class as input to the rule, and a related but distinct class as output. By providing with each lexical rule a generic class frame which specifies the general form and predictable properties of the rule's output, we avoid unnecessary work when the lexical rule applies. The particular output frame will thus get its specifications from two sources: idiosyncratic information copied or computed from the particular input frame, and predictable information available via the class/inheritance links.

As usual, we depend on an example to make the notions clear; consider the lexical rule which takes active, transitive verb frames as input, and produces the corresponding passive verb frames. A prose description of this passive lexical rule follows:

Passive Lexical Rule

If F0 is a transitive verb frame with spelling XXX, then F1 is the corresponding passive frame, where
(1) F1 is an instance of the generic PASSIVE class frame
(2) F1 has as its spelling whatever the past participle's spelling is for F0 (XXXED if regular, stipulated if irregular)
(3) F1 has as its subject's role the role of F0's object, and assigns the role of F0's subject to F1's optional PP-BY.
(4) F1 has OBJECT deleted from its obligatory list.
(5) F1 has as its semantics the semantics of F0.

It is in the TRANSITIVE frame that we declare the applicability of the passive lexical rule, which potentially can apply to each instance (unless explicitly blocked in some frame lower in the lexicon hierarchy, for some particular verb like *resemble*). By triggering particular lexical rules from selected generic frames, we avoid unnecessary attempts to apply irrelevant rules each time a new lexical item is created. The TRANSITIVE frame, then, has roughly the following structure:

```
(TRANSITIVE
   (CLASSES  (subcyclic))
   (OBLIGATORY  (object) (subject))
   (FEATURES  (control trans))
   (LEX-RULES  (passive-rule))
   )
```

The generic frame of which every output from the passive rule is an instance looks as follows:

```
(PASSIVE
   (CLASSES  (verb))
   (FEATURES  (predicative plus) (form pas))
   (OPTIONAL  (pp-by))
   )
```

An example, then, of a verb frame which serves as input to the passive rule is the frame for the transitive verb *make*, whose entry in the lexicon is given below. Keep in mind that a great deal of inherited information is part of the description for *make*, but does not need to be mentioned in the entry for *make* below; put differently, the relative lack of grammatical information appearing in the *make* entry below is a consequence of our maintaining the strong position that only information which is idiosyncratic should be included in a lexical entry.

```
(MAKE
   (CLASSES  (main) (base) (transitive))
   (SPELLING  (make))
   (SUBJECT  (role (make.er)))
   (OBJECT  (role (make.ed)))
   (LEX-RULES (past-participle
                (irreg-spell: "made"))
              (past
                (irreg-spell: "made"))))
```

Upon application of the passive lexical rule to the make frame, the corresponding passive frame MADE-PASSIVE is produced, looking like this:

```
(MADE-PASSIVE
   (CLASSES  (main) (passive) (transitive))
   (SPELLING (made))
   (SUBJECT  (role (make.ed)))
   (PP-BY    (role (make.er)))
   )
```

Note that the *MADE-PASSIVE* frame is still a main verb and still transitive, but is not connected by any inheritance link to the active *make* frame; the passive frame is not an instance of the active frame. This absence of any actual inheritance link between input and output frames is generally true for all lexical rules,

[7] See, e.g., Stanley (1967), Jackendoff (1975), Bresnan (1982).

not surprisingly once the inheritance link is understood. As a result, all idiosyncratic information must (loosely speaking) be copied from the input to the output frame, or it will be lost. Implicit in this last remark is the assumption that properties of a lexical item which are idiosyncratic should only be stated once by the creator of the lexicon, and then propagated as appropriate by lexical rules operating on the basic frame which was entered by hand.

All of our lexical rules, including both inflectional rules, such as the lexical rule which makes plural nouns from singular nouns, and derivational rules, such as the nominalization rule which produces nouns from verbs, share the following properties: each rule specifies the class of frames which are permissible inputs; each rule specifies a generic frame of which every one of the rule's outputs is an instance; each rule copies idiosyncratic information from the input frame while avoiding copying information which can still be inherited; each rule takes as input a single-word lexical frame and produces a single-word lexical frame (no phrases in either case); each rule permits the input frame to stipulate an irregular spelling for the corresponding output frame, blocking the regular spelling; and each rule produces an output which cannot be input to the same rule. Most of these properties we believe to be well-motivated, though it may be that, for example, a proper treatment of idioms will cause us to weaken the single-word input and output restriction, or we may find a lexical rule which can apply to its own output. The wealth of work in theoretical linguistics on properties of lexical rules should greatly facilitate the fine-tuning of our implementation we extend our coverage.

One weakness of the current implementation of lexical rules is our failure to represent the lexical rules themselves as frames, thus preventing us from taking advantage of inheritance and other representational tools that we use to good purpose both for the lexical rules and for the phrase structure rules, about which we'll say more below.

A final remark about lexical rules involves the role of some of our lexical rules as replacements for metarules in the standard GPSG framework. Those familiar with recent developments in that framework are aware that metarules are now viewed as necessarily constrained to operate only on lexically-headed phrase structure rules,[8] but once that move has been made, it is then not such a drastic move to attempt the elimination of metarules altogether in favor of lexical rules.[9] This is the very road we are on. We maintain that the elimination of metarules is not only a nice move theoretically, but also advantageous for implementation.

[8] See Flickinger (1983) for an initial motivation for such a restriction on metarules.

[9] See Pollard (1985) for a more detailed discussion of this important point.

2.4. Nouns from database entities

The third mechanism we use for structure-sharing allows us to leave out of the lexicon altogether the vast majority of common and proper nouns that refer to entities in the target database, including in the lexicon only those nouns which have some idiosyncratic property, such as nouns with irregular plural forms, or mass nouns. This mechanism is simply a procedure much like a lexical rule, but which takes as input the name of some actual database frame, and produces a lexical frame whose spelling slot now contains the name of the database frame, and whose semantics corresponds to the database frame. Such a frame is ordinarily created when parsing a given sentence in which the word naming the database frame appears, and is then discarded once the query is processed. Of course, in order for this strategy to work, the database frame must somehow be linked to the word that refers to it, either by having the frame name be the same as the word, or by having constructed a list of pairings of each database frame with the English spelling for words that refer to that frame. Unlike the other two mechanisms (inheritance and lexical redundancy rules), this pairing of database frames with lexical entries tends to be application-specific, since the front end of the system must depend on a particular convention for naming or marking database frames. Yet the underlying intuition is a reasonable one, namely that when the parser meets up with a word it doesn't recognize, it attempts to treat it as the name of something, either a proper noun or a common noun, essentially leaving it up to the database to know whether the name actually refers to anything.

As an example, imagine that the frame for Pullum (the consultant, not the proper noun) is present in the target database, and that we wish to process a query which refers by name to Pullum (such as *Does Pullum have a modem?*). It would not be necessary to have constructed a proper-name frame for Pullum beforehand, given that the database frame is named Pullum. Instead, the mechanism just introduced would note, in analyzing the query, that Pullum was the name of a frame in the target database; it would consequently create the necessary proper-name frame usable by the parser, possibly discarding it later if space were at a premium. Where an application permits this elimination of most common and proper nouns from the lexicon, one gains not only considerable space savings, but a sharp reduction in the need for additions to the lexicon by naive users as the target database grows.

2.5. On-the-fly frames

All three of the mechanisms for structure-sharing that we have discussed here have in common the additional important property that they can be applied without modification either before ever analyzing a query, or on the fly when trying to handle a particular query. This property is important for us largely

because in developing the system we need to be able to make alterations in the structure of the lexicon, so the ability to apply these mechanisms on the fly means that changes to the lexicon have an immediate and powerful effect on the behavior of the system. As mentioned earlier, another significant factor has to do with time/space trade-offs, weighing the cost in memory of storing redundantly specified lexical entries against the cost in time of having to reconstruct these derived lexical entries afresh each time. Depending on the particular development task, one of the two options for deriving lexical items is preferable over the other, but both options need to be available.

3. The grammar

As we advertised above, the wholesale moving of grammatical information from the phrase structure rules to the lexicon has led to a dramatic reduction in the number of these rules. The rules that remain are usually quite general in nature, and make crucial use of the notion *head* of a constituent, where the head of a noun phrase is a noun, the head of a verb phrase (and of a sentence) is a verb, and so on. In each case, it is the head that carries most of the information about what syntactic and semantic properties its sister(s) in the constituent must have.[10] For example, the single rule which follows is sufficient to construct the phrase structure for the sentence *The consultant works.*

```
Grammar-Rule-1
   X  ->  C1  H[CONTROL INTRANS]
```

The rule is first used to construct the noun phrase *the consultant*, taking *consultant* as the head, and using the information on the lexical frame CONSULTANT-COMMON (which is inherited from the COMMON-NOUN class) that it requires a determiner as its only complement in order to form a complete noun phrase. Then the rule is used again with the lexical frame WORK-THIRD-SING taken as the head, and using the information that it requires a nominative singular noun phrase (which was just constructed) as its only obligatory complement in order to make a complete sentence.

Another example will also allow us to illustrate how information once thought to be clearly the responsibility of phrase structure rules is in fact more simply represented as lexical information, once one has the power of a highly structured lexicon with inheritance available. A second rule in the grammar is provided to admit an optional constituent after an intransitive head, such as *works on Tuesdays* or *modem for Pullum*:

```
Intransitive-Adjunct-Rule
   X  ->  H[CONTROL INTRANS]  ADJUNCT
```

[10] See Pollard (1984) for a thorough discussion of head grammars.

This works quite well for prepositional phrases, but is by no means restricted to them. Eventually we noticed that another standard rule of English grammar could be eliminated given the existence of this rule; namely, the rule which admits relative clauses as in *man who works* for the sentence *Smith hired the man who works*:

```
Relative-Clause-Rule
   X  ->  H[MAJOR N]  S[REL]
```

It should soon be clear that if we add a single piece of information to the generic COMMON-NOUN class frame, we can eliminate this rule. All that is necessary is to specify that a permissible adjunct for common nouns is a relative clause (leaving aside the semantics, which is quite tractable). By stating this fact on the COMMON-NOUN frame, every lexical common noun will be ready to accept a relative clause in just the right place using the Intransitive-Adjunct-Rule. In fact, it seems we can use the same strategy to eliminate any other specialized phrase structure rules for admitting post-nominal modifiers (such as so-called reduced relative clauses as in *The people working for Smith are consultants).*

This example suggests one direction of research we are pursuing: to reduce the number of rules in the grammar to an absolute minimum. At present it still seems to be the case that some small number of phrase structure rules will always be necessary; for example, we seem to be unable to escape a PS rule which admits plural nouns as full noun phrases without a determiner, as in *Consultants work* (but not **Consultant work).* Relevant issues we will leave unaddressed here involve the role of the PS rules in specifying linear order of constituents, whether the linking rules of GPSG (which we still employ) could ever be pushed into the lexicon, and whether in fact both order and linking rules ought to be pushed instead into the parser.

4. Conclusion

Having sketched the mechanisms employed in reducing redundant specification in the lexicon for the HPSG system, and having indicated the brevity of the grammar which results from our rich lexicon, we now summarize the advantages we see in representing the lexicon as we do, apart from the obvious advantage of a much smaller grammar. These advantages have to do in large part with the rigors of developing a large natural language system, but correspond at several points to concerns in theoretical linguistics as well.

First are a set of advantages that derive from being able to make a single substitution or addition which will effect a desired change throughout the system. This ability obviously eases the task of development based on experimentation, since one can quickly try several minor variations of, say, feature combinations and accu-

rately judge the result on the overall system. Of equal importance to development is the consistency provided, given that one can make a modification to, say, the features for plural nouns, and be sure that all regular nouns will reflect the change consistently. Third, we can handle many additions to the lexicon by users without requiring expertise of the user in getting all the particular details of a lexical entry right, for an important (though far from complete) range of cases. Note that this ability to handle innovations seems to have a close parallel in people's ability to predict regular inflected forms for a word never before encountered.

A second advantage that comes largely for free given the inheritance mechanisms we employ involves the phenomenon referred to as *blocking*[11], where the existence of an irregular form of a word precludes the application of a lexical rule which would otherwise produce the corresponding regular form. By allowing individual lexical entries to turn off the relevant lexical rules based on the presence in the frame of an irregular form, we avoid producing, say, the regular past tense form **maked*, since as we saw, the entry for *make* warns explicitly of an irregular spelling for the past tense form.

Already mentioned above was a third advantage of using the mechanisms we do, namely that we can use inheritance to help us specify quite precisely the domain of a particular lexical rule, rather than having to try every lexical rule on every new frame only to discover that in most cases the rule fails to apply.

Finally, we derive an intriguing benefit from having the ability to create on-the-fly noun frames for any database entry, and from our decision to store our lexical items using the same representation language that is used for the target database: we are able without additional effort to answer queries about the make-up of the natural language system itself. That is, we can get an accurate answer to a question like *How many verbs are there?* in exactly the way that we answer the question *How many managers are there?*. This ability of our system to reflect upon its own structure may prove to be much more than a curiosity as the system continues to grow; it may well become an essential tool for the continued development of the system itself. The potential for usefulness of this reflective property is enhanced by the fact that we now also represent our grammar and several other data structures for the system in this same frame representation language, and may progress to representing in frames other intermediate stages of the processing of a sentence. This enhanced ability to extend the lexical coverage of our system frees us to invest more effort in meeting the many other challenges of developing a practical, extensible implementation of a natural language system embedded in a serious linguistic theory.

[11] See Aronoff (1976) for discussion.

REFERENCES

Aronoff, M. (1976) *Word Formation in Generative Grammar*. Linguistic Inquiry Monograph 1. Cambridge, Mass.: The MIT Press.

Bobrow, R. and B. Webber (1980a) "Psi-Klone" in *Proceedings of the 1980 CSCSI/SCEIO AI Conference*, Victoria, B.C.

Bobrow, R. and B. Webber (1980b) "Knowledge Representation for Syntactic/ Semantic Processing," in *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford University.

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

Flickinger, Daniel P. (1983) "Lexical Heads and Phrasal Gaps," in Flickinger, et al., *Proceedings of the West Coast Conference on Formal Linguistics*, vol. 2. Stanford University Linguistics Dept.

Gawron, J. et al. (1982) "Processing English with a Generalized Phrase Structure Grammar", *ACL Proceedings* 20.

Gazdar, G., E. Klein, G. K. Pullum, and I. A. Sag (1985) *Generalized Phrase Structure Grammar*. Cambridge, Mass.: Harvard University Press.

Jackendoff, R. (1975) "Morphological and Semantic Regularities in the Lexicon", *Language* 51, no. 3.

Pollard, C. (1984) *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*. Doctoral dissertation, Stanford University.

Pollard, C. (1985) "Phrase Structure Grammar Without Metarules," presented at the 1985 meeting of the West Coast Conference on Formal Linguistics, Los Angeles.

Rosenberg, S. (1983) "HPRL: A Language for Building Expert Systems", *IJCAI* 83: 215-217.

Stanley, R. (1967) "Redundancy Rules in Phonology", *Language* 43, no. 1.