Head-Driven Generation with HPSG

Graham Wilcock*
Centre for Computational Linguistics
University of Manchester Institute
of Science and Technology
PO Box 88, Manchester M60 1QD
United Kingdom
graham@ccl.umist.ac.uk

Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-01
Japan
matsu@is.aist-nara.ac.jp

Abstract

As HPSG is head-driven, with clear semantic heads, semantic head-driven generation should be simple. We adapt van Noord's Prolog generator for use with an HPSG grammar in ProFIT. However, quantifiers and context factors are difficult to include in head-driven generation. We must adopt recent theoretical proposals for lexicalized scoping and context. With these revisions, head-driven generation with HPSG is not so simple, but it is possible.

1 Introduction

A natural approach to generation with Head-driven Phrase Structure Grammar (Pollard and Sag, 1994) is to use a head-driven algorithm. HPSG is head-driven not only syntactically, but also semantically. While the Head Feature Principle requires identity of major syntactic features between a phrase and its syntactic head daughter, the Semantics Principle (in various formulations) requires identity of major semantic features between a phrase and its semantic head daughter. Since the semantic head is very clearly defined in HPSG, semantic head-driven generation should be easy to implement.

Efficient head-driven generation algorithms, such as BUG, SHD and CSHD, have been presented as Prolog algorithms for use with DCG grammars. In Section 2 we briefly describe how an HPSG grammar can be implemented as a PSG with typed feature structures, which can be compiled into a DCG by the ProFIT system. In this way, HPSG grammars can be used with the existing Prolog algorithms.

Such a combination of head-driven grammar and head-driven generator works well if the semantics is strictly head-driven. However, in Section 3 we show that if we implement the HPSG textbook semantics, with quantifier storage and contextual background conditions, the notion of semantic head becomes unclear and this approach no longer works. In fact, head-driven generation of even simple phrases such as "Kim walks" (Chapter 1 of the HPSG textbook) raises fundamental difficulties.

To use a semantic head-driven algorithm, we must adopt recent HPSG proposals to put quantifier store and contextual background inside semantic heads. We summarize these proposals in Section 4, and show how they can be implemented in the ProFIT HPSG grammar. We conclude that head-driven generation with HPSG is possible, but there are some difficulties in implementing this approach.

2 Head-Driven Generation

We assume that generation starts from logical forms, which may be represented for HPSG as typed feature structures. Logical form is not a separate linguistic level in HPSG, but is equated with semantic content. In this section, we take the starting logical form for generation to be a semantic feature structure which will be identical to the CONTENT feature of the top-level HPSG sign to be generated.

2.1 Semantic heads

Head-driven generation algorithms are based on the idea that most grammar rules have a semantic head daughter whose logical form is identical to the logical form of the mother. The bottom-up generation (BUG) algorithm of van Noord (1990) requires every rule to have such a head (except lexical entries). The semantic head-driven (SHD) algorithm of Shieber et al. (1990) relaxes this, dividing rules into chain rules with such a head (processed bottom-up), and non-chain rules (processed top-down). The chart-based semantic head-driven (CSHD) algorithm[1] of Haruno et al. (1996) increases efficiency by using a chart to eliminate recomputation of partial results.

Head-driven bottom-up generation is efficient as it is geared both to the input logical form (head-driven) and to lexical information (bottom-up). It is good for HPSG, which is highly lexicalist and has

* Visiting researcher of Sharp Corporation, Japan.

[1] For simplicity we illustrate the approach with BUG. A ProFIT/HPSG framework using the CSHD algorithm is described by Wilcock and Matsumoto (1996).

```
'HFP'  := synsem!loc!cat!head!HF &
   hd_dtr!synsem!loc!cat!head!HF.
'SemP' := synsem!loc!cont!Cont &
   hd_dtr!synsem!loc!cont!Cont.
'SemP'(adjunct) := synsem!loc!cont!Cont &
           adj_dtr!synsem!loc!cont!Cont.

hd_ph := <hd_ph & @'HFP' &
    synsem!loc!cat!val!comps![].
hd_nexus_ph := <hd_nexus_ph & @hd_ph & @'SemP'.
hd_subj_ph := <hd_subj_ph & @hd_nexus_ph &
    @'VALP'(spr) & @'VALP'(comps) &
    synsem!loc!cat!val!subj![].
hd_comp_ph := <hd_comp_ph & @hd_nexus_ph &
    @'VALP'(subj) & @'VALP'(spr).

@hd_subj_ph & phon!P0-PN &
    hd_dtr!(Head &
            synsem!loc!cat!val!subj![S]) &
    subj_dtr!(Subj & synsem!S)
---> [Head & <phrase & phon!P1-PN,
      Subj & <phrase & phon!P0-P1].

@hd_comp_ph & phon!P0-PN &
    hd_dtr!(Head &
            synsem!loc!cat!val!comps![C]) &
    comp_dtrs![Comp & synsem!C]
---> [Head & <word & phon!P0-P1,
      Comp & <phrase & phon!P1-PN].
```

Figure 1: Principles, Phrase Types, Schemata

a clear definition of semantic head: in head-adjunct phrases, the adjunct daughter is the semantic head; in other headed phrases, the syntactic head daughter is the semantic head. In both cases, the Semantics Principle basically requires the content of the semantic head to be identical to the content of the mother. If we ignore coordinate structures, and if we equate logical form with semantic content for now, then all HPSG grammar rules are SHD chain rules, meeting the requirement of the BUG algorithm.

2.2 HPSG in ProFIT

ProFIT: Prolog with Features, Inheritance and Templates (Erbach, 1995) is an extension of Prolog which supports inheritance-based typed feature structures. The type hierarchy is declared in a signature, which defines subtypes and appropriate features of every type. Terms with typed feature structures can then be used alongside normal terms. Using the signature declarations, the ProFIT system compiles the typed feature structures into normal Prolog terms, which can be compiled by the Prolog system.

Figure 1 shows some implementation details. We use ProFIT templates (defined by ':=') for principles such as the Head Feature Principle ('HFP') and Semantics Principle ('SemP'). Templates are expanded where they are invoked (by @'HFP' or @'SemP'). The type hierarchy includes the phrase type hierarchy of Sag (1997). As ProFIT does not support dynamic constraints, we use templates to specify phrasal constraints. For example, for head-nexus phrases, the **hd_nexus_ph** template specifies the **<hd_nexus_ph** type, invokes general constraints on headed phrases (such as HFP) by @hd_ph, and invokes the Semantics Principle by @'SemP'.

Immediate dominance schemata are implemented as PSG rules, using schematic categories *word* and *phrase*, not traditional categories (NP, VP etc). To simplify the generator, the semantic head is first in the list of daughters. Linear precedence is specified by the PHON strings, implemented as Prolog difference lists. Example rules for Head-Subject and Head-Complements Schemata are shown in Figure 1.

2.3 HPSG Interface for BUG1

van Noord (1990) implements the BUG algorithm as BUG1 in Prolog. For HPSG, we add the ProFIT interface in Figure 2. Templates identify the head features (HF) and logical form (LF), and keep the algorithm independent from HPSG internal details. Note that **link**, used by van Noord (1990) to improve the efficiency of the algorithm, is replaced by the HPSG Head Feature Principle.

```
hf(HF) := synsem!loc!cat!head!HF.
lf(LF) := synsem!loc!cont!LF.

predict_word( @lf(LF) & @hf(HF), Word ) :-
  lex( Word & @lf(LF) & @hf(HF) ).
predict_rule(Head,Mother,Others,@hf(HF)) :-
  ( Mother & @hf(HF) ---> [Head|Others] ).

generate(LF, Sign, String) :-
  bug1( Sign & phon!String-[] & @lf(LF) ).

/* BUG1: van Noord 1990 */
bug1(Node) :-
  predict_word(Node, Small),
  connect(Small, Node).
connect(Node, Node).
connect(Small, Big) :-
  predict_rule(Small,Middle,Others,Big),
  gen_ds(Others),
  connect(Middle, Big).
gen_ds([]).
gen_ds([Node|Nodes]) :-
  bug1(Node),
  gen_ds(Nodes).
```

Figure 2: ProFIT/HPSG Interface for BUG1

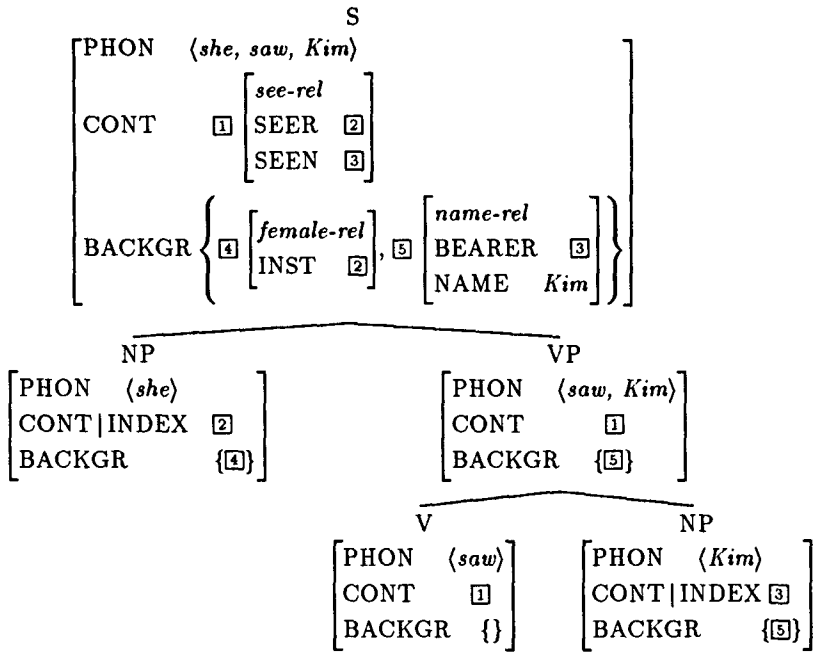

Figure 3: Contextual Background (Phrasal Amalgamation)

3 Quantifiers and Context

Head-driven generation as in Section 2 works fine if the semantics is strictly head-driven. All semantic information must be inside the CONTENT feature, and cannot be distributed in other features such as QSTORE or BACKGR. When an NP is assigned to the semantic role of a verb, the whole of the NP's CONTENT must be assigned, not only its INDEX. This differs significantly from HPSG theory.

3.1 Quantifier Storage and Retrieval

There is a complication in Pollard and Sag (1994) caused by the use of Cooper storage to handle scope ambiguities. While scoped quantifiers are included in the QUANTS list within CONTENT, unscoped quantifiers are stored in the QSTORE set outside CONTENT. So logical form for generation needs to include QSTORE as well as CONTENT.

In this approach, a quantifier may be retrieved at any suitable syntactic node. A quantifier retrieved at a particular node is a member of the QSTORE set (but not the QUANTS list) of some daughter of that node. Due to the retrieval it is a member of the QUANTS list (but not the QSTORE set) of the mother node. Pollard and Sag (1994) define a modified Semantics Principle to cater for this, but the effect of retrieval on QSTORE and QUANTS means that the mother and the semantic head daughter must have different logical forms. The daughter is the semantic head by the HPSG definition, but not as required by the generation algorithm.

3.2 Contextual Background

In addition to semantic content, natural language generation requires presuppositions and other pragmatic and discourse factors. In HPSG, such factors are part of CONTEXT. To specify these factors for generation, the usual approach is to include them in the logical form. So logical form needs to include CONTEXT as well as CONTENT and QSTORE. This extended logical form is defined for BUG1 by replacing the ProFIT template for 'lf(LF)' shown in Figure 2 with the new template in Figure 4.

```
lf(ct!CT & qs!QS & cx!CX) :=
    synsem!loc!(cont!CT & qstore!QS & conx!CX).
```

Figure 4: Extending the Logical Form

However, head-driven generation does not work with this inclusive logical form, given the theory of Pollard and Sag (1994). Even if we ignore quantifier retrieval and look at a very simple sentence, there is a fundamental difficulty with CONTEXT.

Figure 3, from Wilcock (1997), shows the HPSG analysis of *she saw Kim*. Note that *she* has a non-empty BACKGR set (shown by tag 4), stating a pragmatic requirement that the referent is female.

This background condition is part of CONTEXT, and is passed up from NP to S by the Principle of Contextual Consistency. Similarly, *Kim* has a background condition (shown by tag [5]) that the referent bears this name. This is also passed from NP to VP, and from VP to S.

S, VP and V share the same CONTENT (shown by tag [1]). If logical form is restricted to semantic content as in Figure 2, then V is the semantic head of VP and VP is the semantic head of S, not only in terms of the HPSG definition but also in terms of the BUG algorithm. In this case, *saw* can be found immediately by `predict_word` in BUG1. But if we extend logical form as in Figure 4, to include the context factors required for adequate realization, it is clear from Figure 3 that S does not have the same logical form as VP, and VP does not have the same logical form as V, as their BACKGR sets differ. Therefore, although V is still the semantic head of VP according to the HPSG definition, it is not the semantic head according to the BUG algorithm. Similarly, VP is still the semantic head of S for HPSG, but it is not the semantic head for BUG. In this case, `predict_word` cannot find any semantic head word in the lexicon, and BUG1 cannot generate the sentence.

4 Revising the Grammar

If we include unscoped quantifiers and contextual background in logical form, we see that there are two different definitions of "semantic head": the HPSG definition based on adjunct daughter or syntactic head daughter, and the BUG algorithm definition based on identity of logical forms. However, recent proposals for changes in HPSG theory suggest that the two notions of semantic head can be brought back together.

4.1 Lexical amalgamation in HPSG

In Pollard and Sag (1994), QSTORE and BACKGR sets are *phrasally amalgamated.* The Quantifier Inheritance Principle requires a phrase's QSTORE to be the set union of the QSTOREs of all daughters, minus any quantifiers in the phrase's RETRIEVED list. The Principle of Contextual Consistency requires a phrase's BACKGR to be the set union of the BACKGR sets of all the daughters.

It has recently been proposed that these sets should be *lexically amalgamated.* A syntactic head word's arguments are now lexically specified in its ARGUMENT-STRUCTURE list. The word's set-valued features can therefore be defined in terms of the amalgamation of the set-valued features of its arguments.

Lexical amalgamation of quantifier storage was proposed by Pollard and Yoo (1995). They change QSTORE into a local feature which can be included in the features subcategorized for by a lexical head, and can therefore be lexically amalgamated in the head. A phrase no longer inherits unscoped quantifiers directly from all daughters, instead they are inherited indirectly via the semantic head daughter.

Lexical amalgamation of CONTEXT, proposed by Wilcock (1997), follows the same approach. As CONTEXT is a local feature, it can be subcategorized for by a head word and lexically amalgamated in the head by means of a BACKGR amalgamation constraint. Instead of a phrase inheriting BACKGR conditions directly from all daughters by the Principle of Contextual Consistency, they are inherited indirectly via the "contextual head" daughter which is the same as the semantic head daughter.

4.2 Lexical amalgamation in ProFIT

In the ProFIT implementation, QSTORE sets and BACKGR sets are Prolog difference lists. Lexical amalgamation of both sets is shown in Figure 5, the lexical entry for the verb "saw". The subject's BACKGR set B0-B1 and the object's BACKGR set B1-BN are amalgamated in the verb's BACKGR set B0-BN. The subject and object QSTORE sets, Q0-Q1 and Q1-QN, are similarly amalgamated in the verb's QSTORE Q0-QN.

```
lex( phon![saw|X]-X & @verb &
  synsem!loc!(
    cat!(head!<verb &
         val!(subj![@np &
              loc!(cat!head!case!<nom &
                   cont!index!Subj &
                   conx!backgr!B0-B1 &
                   qstore!Q0-Q1)] &
              comps![@np &
              loc!(cat!head!case!<acc &
                   cont!index!Obj &
                   conx!backgr!B1-BN &
                   qstore!Q1-QN)])) &
    cont!nuc!(seer!Subj & seen!Obj) &
    conx!backgr!B0-BN &
    qstore!Q0-QN) ).
```

Figure 5: Lexical amalgamation

The basic Semantics Principle, for semantic content only, was implemented by the ProFIT templates 'SemP' and 'SemP'(adjunct) as shown in Figure 1. In order to include unscoped quantifiers and background conditions in logical form, as in Figure 4, and still make it possible for the logical form of a phrase to be identical to the logical form of its

semantic head, the Semantics Principle is replaced and extended. As proposed by Wilcock (1997), we need three principles: Semantic Head Inheritance Principle (SHIP), Quantifier Inheritance Principle (QUIP), and Contextual Head Inheritance Principle (CHIP). These are implemented by templates as shown in Figure 6 (only the non-adjunct forms are shown). To include the three principles in the grammar, the template for **hd_nexus_ph** in Figure 1 is extended as shown in Figure 6.

```
'SHIP'    := synsem!loc!cont!Cont &
     hd_dtr!synsem!loc!cont!Cont.
'QUIP'    := synsem!loc!qstore!QS &
     hd_dtr!synsem!loc!qstore!QS.
'CHIP'    := synsem!loc!conx!Conx &
     hd_dtr!synsem!loc!conx!Conx.

hd_nexus_ph := <hd_nexus_ph & @hd_ph &
    @'SHIP' & @'QUIP' & @'CHIP'.
```

Figure 6: Inheritance of Logical Form

With these revisions, it is possible to include unscoped quantifiers and background conditions in the starting logical form, and perform head-driven generation successfully using the BUG1 generator. However, there remain various technical difficulties in this implementation. The ProFIT system does not support either dynamic constraint checking or set-valued features. The methods shown (template expansion and difference lists) are only partial substitutes for the required facilities.

5 Conclusion

The combination of a head-driven HPSG grammar with a head-driven generation algorithm is a natural approach to surface realization. We showed how van Noord's BUG1 generator can easily be adapted for use with an HPSG grammar implemented in ProFIT, and that this works well if the semantics is strictly head-driven. However, while the apparently clear definition of semantic head in HPSG should make semantic head-driven generation easy to implement, we found that if we implement the full HPSG textbook semantics, with quantifier storage and contextual background conditions, the notion of semantic head becomes unclear. Surprisingly, this natural approach does not work, even for simple examples.

In order to use semantic head-driven generation algorithms with HPSG, we must adopt recent proposals to include quantifier storage and contextual background inside semantic heads by means of lexical amalgamation. We showed how the grammar in ProFIT can be extended with these proposals. We therefore conclude that head-driven generation with HPSG is indeed a feasible approach to surface realization, although there are some technical difficulties.

Acknowledgements

We are grateful to Mr Yoshikazu Nakagawa of Sharp Corporation for making our collaboration possible.

References

Gregor Erbach. 1995. ProFIT: Prolog with Features, Inheritance, and Templates. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 180–187, Dublin.

Masahiko Haruno, Yasuharu Den, and Yuji Matsumoto. 1996. A chart-based semantic head driven generation algorithm. In G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, pages 300–313. Springer.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar.* CSLI Publications and University of Chicago Press.

Carl Pollard and Eun Jung Yoo. 1995. Quantifiers, *wh*-phrases and a theory of argument selection. Tübingen HPSG workshop.

Ivan A. Sag. 1997. English relative clause constructions. *Journal of Linguistics*, 33(2):431–484.

Stuart M. Shieber, Gertjan van Noord, Fernando C.N. Pereira, and Robert C. Moore. 1990. Semantic head-driven generation. *Computational Linguistics*, 16(1):30–42.

Gertjan van Noord. 1990. An overview of head-driven bottom-up generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 141–165. Academic Press.

Graham Wilcock and Yuji Matsumoto. 1996. Reversible delayed lexical choice in a bidirectional framework. In *16th International Conference on Computational Linguistics (COLING-96)*, pages 758–763, Copenhagen.

Graham Wilcock. 1997. Lexicalization of Context. 4th International Conference on HPSG, Ithaca. To appear in G. Webelhuth, J.-P. Koenig and A. Kathol, editors, *Lexical and Constructional Aspects of Linguistic Explanation.* CSLI Publications.