Syntactic-Head-Driven Generation

Esther König*

Institute for Computational Linguistics, Azenbergstr.12, 70174 Stuttgart,Germany, esther@ims.uni-stuttgart.de

Abstract

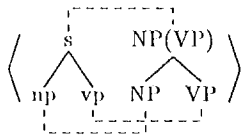
The previously proposed *semantic*-head-driven generation methods run into problems if none of the daughter constituents in the syntacto-semantic rule schemata of a grammar fits the definition of a semantic head given in [Shieber *et al.*, 1990]. This is the case for the semantic analysis rules of certain constraint-based semantic representations, e.g. Underspecified Discourse Representation Structures (UDRSs) [Frank and Reyle, 1992].

Since head-driven generation in general has its merits, we simply return to a syntactic definition of 'head' and demonstrate the feasibility of *syntactic*-head-driven generation. In addition to its generality, a syntactic-head-driven algorithm provides a basis for a logically well-defined treatment of the movement of (syntactic) heads, for which only ad-hoc solutions existed, so far.

1 Introduction

Head-driven generation methods combine both, top-down search and bottom-up combination, in an ideal way. [Shieber *et al.*, 1990] proposed to define the 'head' constituent h of phrase with category x on *semantic* grounds: the semantic representations of h and x are identical. This puts a strong restriction on the shape of semantic analysis rules: one of the leaves must share its semantic form with the root node. However, there are composition rules for semantic representations which violate this restriction, e.g. the schemata for the construction of Underspecified Discourse Representation Structures (UDRSs) [Frank and Reyle, 1992] where, in general, the root of a tree is associated with a strictly larger semantic structure than any of the leaves. In order to make a generation method available for grammars which do not follow the strict notion of a semantic head, a *syntactic*-head-driven generation algorithm is presented, which can be specialized to generate from UDRSs. In a second step, the method will be extended in order to handle the movement of (syntactic) heads in a logically well-defined manner.

The (tactical) generation problem is the task to generate a string from a semantic representation according to the syntax-semantics-relation defined in a given grammar. Let's assume that the latter relation is stated by pairs of trees. The left tree states a local syntactic dependency, i.e. the dominance relation between a root node and a set of leaf nodes and the linear precedence relation among the leaves. The right tree defines the relation among the semantic representation of the root and the semantic representations of the leaves. We assume that there is a one-to-one map from the nonterminal leaf nodes of the (local) syntax tree on the leaf nodes of the (local) semantic derivation tree. Example:



(1)

If one assumes a pairwise linking from left to right then the links between the two trees can be omitted. Although such pairs of trees are reminiscent of synchronous trees in TAG's [Shieber and Schabes, 1991], they are simpler in various ways, in particular because we will *not* make use of the adjunction operation later on. In essence, pairs of trees are just a graphical notation for what has been put forward as the 'rule-to-rule'-hypothesis, cf. [Gazdar *et al.*, 1985], the fact that in the grammar each syntax rule is related with a semantic analysis rule. However, on the long run, the tree notation suggests a more general relation, e.g. more internal structure or additional, terminal leaf nodes in the local syntax tree.

An obvious way to implement a generation procedure (see Fig.1) is to relate the input semantics with the start symbol of the grammar and then to try to expand this node in a top-down manner according to the rules specified in the grammar. This node expansion corresponds to an application of the *(predict)*-rule in the following abstract specification of a top-down generator. Generation terminates successfully if all the leaf nodes are labeled with terminals *(success)*. The question is which method is used to make two, possibly complex symbols equal. For the sake of simplicity, we assume that the open leaves x_0 resp. X_0 are matched by (feature) term unification with the corresponding mother nodes in the grammar rule. However, for the semantic form X_0, a decidable variant of higher order unification might be used instead, in order to include the reduction of λ-expressions. Of course, the necessary precautions have to be taken in order to avoid the confusion between object- and meta-level variables, cf. [Shieber *et al.*, 1990].

A depth-first realization of this abstract top-down algorithm would work fine as long as the semantic rep-

*The research reported here has been funded by the Sonderforschungsbereich 340 "Sprachtheoretische Grundlagen für die Computerlinguistik", a project of the German National Science Foundation DFG.

all leaves of the syntax tree are labeled with terminals *(success)*

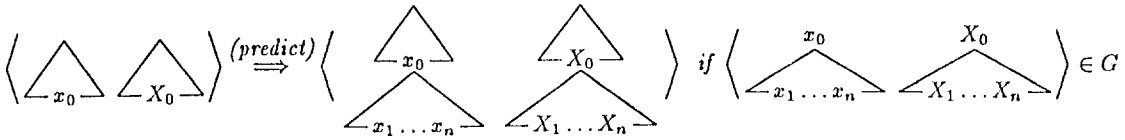
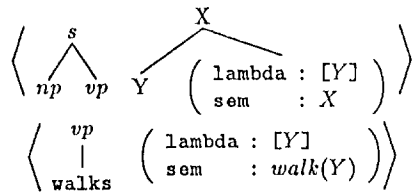


Figure 1: Top-Down Generation (G grammar description; x_i syntactic category; X_i semantic representation)

resentations of the leaves are always strictly smaller in size as the semantic form of the root node. But, if the actual semantic decomposition takes place in the lexicon, the semantic representations of some subgoals will be variables, which stand for semantic representations of any size:

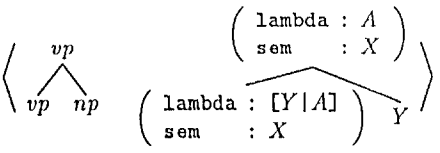


(2)

A strict left-to-right, depth-first expansion of subgoals might run into problems with the grammar fragment in (2) if a left-recursive *np*-rule exists, because the semantics of the *np* is only instantiated once the 'semantic head' of the *vp* has been looked up in the lexicon.

2 Previous work

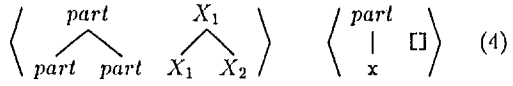
A top-down, semantic-structure-driven generation algorithm has been defined by [Wedekind, 1988] which gives a basis for dynamic subgoal-reordering guided by the semantic input. Some proposals have been made for subgoal reordering at compile-time, e.g. [Minnen *et al.*, 1993] elaborating on the work by [Strzalkowski, 1990]. But there will be no helpful subgoal reordering for rules with semantic head recursion:



(3)

Obviously, a bottom-up component is required. One solution is to keep to a top-down strategy but to do a breadth-first search, cf. [Kohl, 1992], which will be fair and not delay the access to the lexicon forever, as a pure depth-first strategy does. Alternatively, one could adopt a pure bottom-up strategy like the one which has been proposed in [Shieber, 1988] and which is presented in Fig.2 in a highly schematic manner. A lexical entry qualifies as a potential leaf node if its semantic form is a non-trivial substructure of the input semantics (rule *(lex)*). The derivation trees are built up by the *(complete)*-rule. Generation finally succeeds if the root node of the current syntax tree is labeled with the start symbol of the grammar and the root of the semantic analysis tree with the input semantics. Due to the exclusion of phrases with 'empty' semantics (which would be trivial substructures of the input semantics), the method always terminates. However, the lack of top-down guidance will lead, in general, to a lot of non-determinism. The strong substructure condition means that the algorithm will be incomplete for grammars which cover semantically void phrases like expletive expressions, particles, and subphrases of idioms.

The head-corner generator in [van Noord, 1993] is an illustrative instance of a sophisticated combination of top-down prediction and bottom-up structure building, see Fig.3. The rule *(lex)* restricts the selection of lexical entries to those which can be 'linked' to the local goal category (visualized by a dotted line). According to van Noord, two syntax-semantics pairs are linkable if their semantic forms are identical, i.e. $link(\langle x, X\rangle, \langle x_i, X\rangle)$. The rule *(hc-complete)* performs a 'head-corner' completion step for a (linked) phrase x_h, which leads to the prediction of the head's sisters. A link marking can be removed if the linked categories resp. the linked semantic forms are identical (rule *(local-success)*). Generation succeeds if all the leaves of the syntax tree are labeled with terminals and if no link markings exist (rule *(global-success)*). In order to obtain completeness in the general case, the inference schemata of the head-corner generator must be executed by a breadth-first interpreter, since a depth-first interpreter will loop if the semantic analysis rules admit that subtrees are associated with semantic forms which are not proper substructures of the input semantics, and if these subtrees can be composed recursively. Such an extreme case would be a recursive rule for semantically empty particles: ('empty' semantics is represented by the empty list symbol []):



(4)

However, if we assume that structures of that kind do not occur, a depth-first interpreter will be sufficient, e.g. the inference rules of the algorithm can be encoded and interpreted directly in Prolog. Note that van Noord's method is restricted to grammars where phrases have always a lexical semantic head. The algorithm in [Shieber *et al.*, 1990] relaxes this condition.

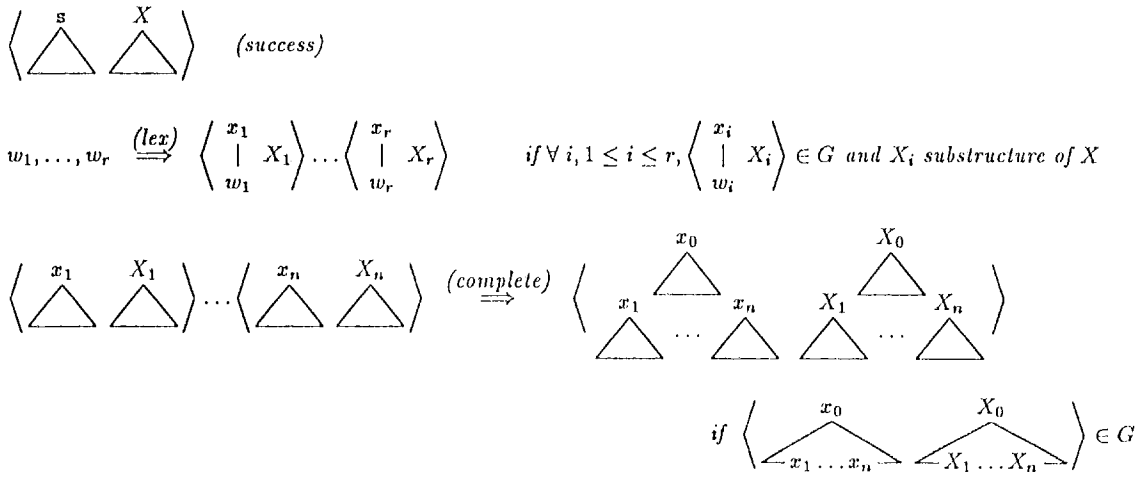


Figure 2: Bottom-Up Generation (G grammar description; s start symbol; X input semantics; x_i syntactic category; X_i semantic representation)

all leaves are labeled with terminals and the tree does not contain any dotted lines *(global-success)*

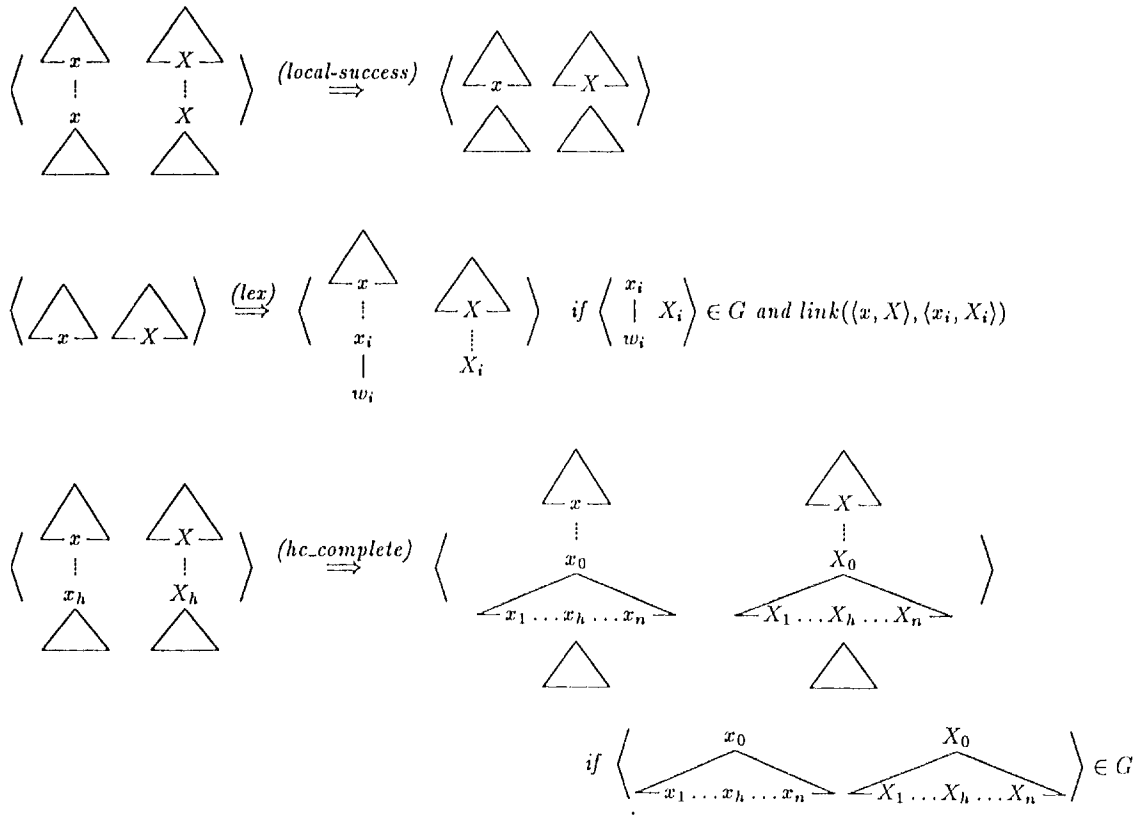


Figure 3: Head-Corner Generator (G grammar description; x_i syntactic category; X_i semantic representation)

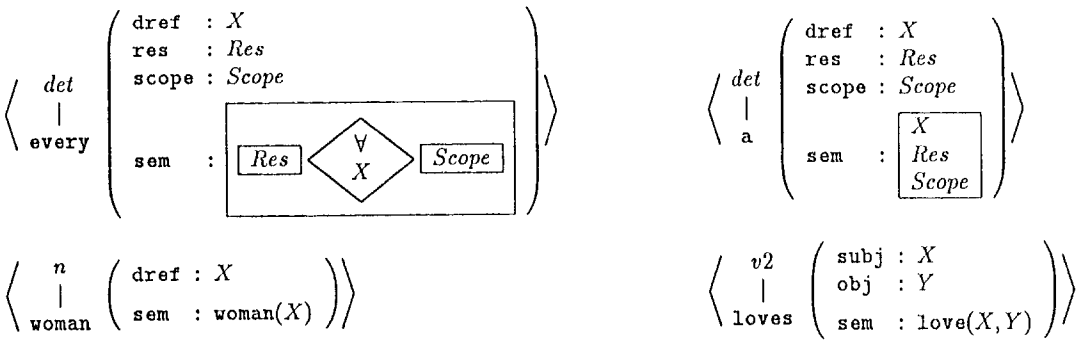
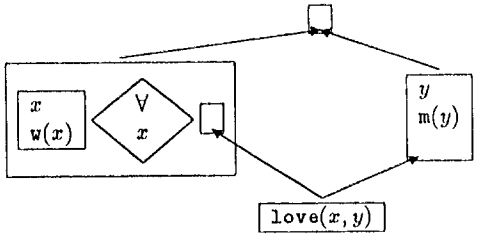


Figure 4: A grammar with UDRS-construction rules - lexicon

3 Underspecified Discourse Representation Structure

In the following, we will present shortly a semantic representation formalism and a corresponding set of analysis rules which resist to the definition of 'semantic head' as it is required in van Noord's head-corner algorithm. [Reyle, 1993] developed an inference system for *Underspecified Discourse Representation Structures* (UDRS's), i.e. Discourse Representation Structures [Kamp and Reyle, 1993] which are underspecified with respect to scope. The following UDRS represents simultaneously the two readings of the sentence `'every woman loves a man'` by leaving the exact structural embedding of the quantified phrases underspecified.



(5)

An arrow pointing from X_2 to X_1 is called a *subordination constraint* and means that the formula X_2 must not have wider scope than X_1. [Frank and Reyle, 1992] proposed rules for the construction of UDRS's in an HPSG-style syntax, cf. [Pollard and Sag, 1993], which are shown in Fig.4 and 5 in a somewhat adapted manner. Semantic composition is performed by the coindexing of the features `dref`, `res`, `subj`, etc. which serve as an interface to the value of the `sem` feature, the actual semantic representation. For the phrase-structure tree rooted with s, there is no leaf which would fulfill the definition of a semantic head given in [Shieber *et al.*, 1990] or [van Noord, 1993]. Hence, the head-corner generator of Fig.3 with a link relation based on semantic heads will not be applicable.

4 Syntactic-head-driven generation

4.1 A new link relation

One could define a *weak notion of a semantic head* which requires that the semantic form of the semantic head is a (possibly empty) substructure of the root semantics. But this is rather meaningless, since now every leaf will qualify as a semantic head. As a way out, there is still the notion of a syntactic head, which can serve as the pivot of the generation process.

Assume that the syntactic head leaf for each local syntax tree has been defined by the grammar writer. We get the following *preliminary version of a syntax-based link relation*:

$$link(\langle x, X\rangle, \langle x_i, X_i\rangle) \tag{6}$$

1. *if either* $x = x_i$
2. *or* x_j is a possible syntactic head of x
   and $link(\langle x_j, X_j\rangle, \langle x_i, X_i\rangle)$

This is the kind of link relation which is used for parsing. In general, it works fine there, because with each lexical lookup a part of the input structure, i.e. of the input string, is consumed. In order to reduce the number of non-terminating cases for generation, a similar precaution has to be added, i.e. the input structure has to be taken into account. The *final version of a syntax-based link relation* incorporates a test for the weak notion of a semantic head:

$$link(\langle x, X\rangle, \langle x_i, X_i\rangle) \;\; if \tag{7}$$

1. *either* $x = x_i$ and
   X_i is a (possibly empty) substructure of X
2. *or* x_j is a possible syntactic head of x
   and $link(\langle x_j, X\rangle, \langle x_i, X_i\rangle)$

The substructure check makes only sense if the semantics X of the current goal is instantiated. This might

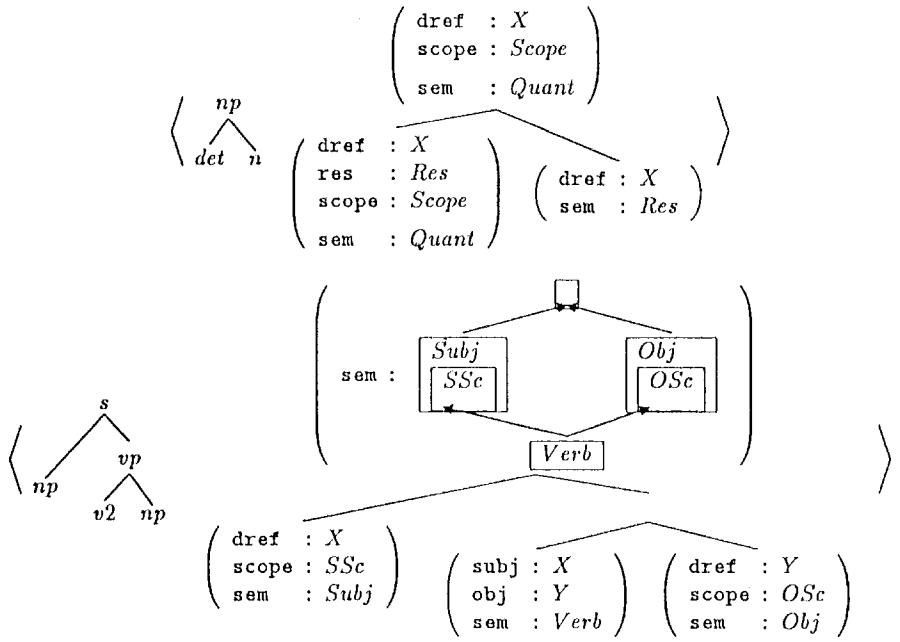
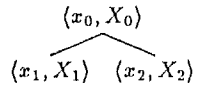


Figure 5: A grammar with UDRS-construction rules - syntax rules

not be the case, when the proper semantic head and the syntactic head differ, and a sister goal of the semantic head is to be expanded before the head itself. Hence, in general, the sister goals must be reordered according to the degree of instantiation of their semantic representations. In addition to the improved termination properties, the condition on the semantic representation helps to filter out useless candidates from the lexicon, i.e. lexical entries which will never become part of the final derivation because their semantic representations do not fit.

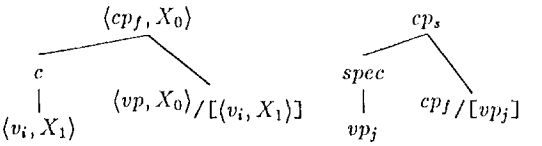
4.2 Grammars with head movement

In order to simplify the representation in the following, we assume that each syntax tree in a grammar is isomorphic to the corresponding semantic analysis tree. This means that both trees can merged into one tree by labeling the nodes with syntax-semantics-pairs:



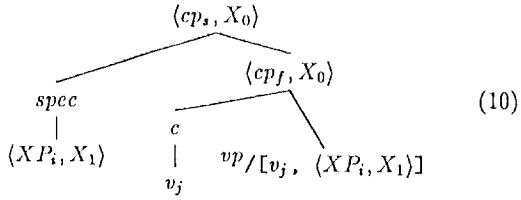
(8)

In [Shieber *et al.*, 1990] an ad-hoc solution was proposed to enforce termination when the semantic head has been moved. By adopting a syntactic-head-driven strategy, head-movement does not cause a problem if the landing site of the head is the 'syntactic head' (or rather: the main functor category of the clause, in categorial grammar terminology) of a superordinate clause. This is postulated by syntactic descriptions like



(9)

where $vp_{/[v_i]}$ means that the derivation of the vp-node has to include an empty v-leaf. In the example in Fig.6, the syntactic head (the c-position) of the cp_f will be visited before the vp is to be derived, hence the exact information of the verb trace will be available in time. Similarly for the movement to the 'vorfeld'. However, if verb second configurations are described by a single structure



(10)

the algorithm runs into a deadlock: the vp-node cannot be processed completely, because the semantics of the XP-trace is unknown, and the expansion of the XP-filler position will be delayed for the same reason. If this syntactic description had to be preferred over the one in (9), the link relation should be further modified. The substructure test wrt. the semantics of the current goal should be replaced by a substructure test

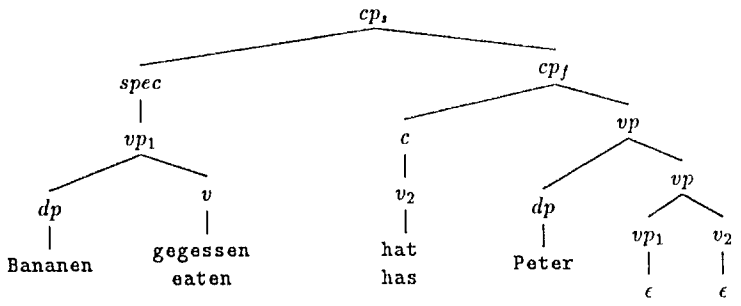


Figure 6: Movement of a complex non-head

wrt. the global input semantics, which leads to a loss of flexibility, as it has been discussed in connection with the pure bottom-up approach.

4.3 Implementation

Since the algorithm has been implemented in the CUF language[1], which includes a `wait`-mechanism, the reodering of subgoals can be delegated to CUF.

Instead of a full-blown substructure test which might be quite complicated on graphs like UDRS's, only the predicate names (and other essential 'semantic' keywords) of the lexical entry are mapped on the current goal semantics. If such a map is not feasible, this lexical entry is dropped.

We restrict the grammars to lexicalized ones. A grammar is lexicalized if for every local syntax tree there is at least one preterminal leaf, cf. [Schabes and Waters, 1993]. Note that lexicalization does not affect the expressibility of the grammar [Bar-Hillel *et al.*, 1960], [Schabes and Waters, 1993]. However, the generation algorithm turns much simpler and hence more efficient. There is no need for a transitive link relation, since a goal can match immediately the mother node of a preterminal. The lexicon access and the head-corner completion step can be merged into one rule schema[2].

A version of the Non-Local-Feature principle of HPSG has been integrated into the algorithm. Every non-head nonterminal leaf of a local tree must come with a (possibly empty) multiset of syntax-semantics pairs as the value of its `to_bind:slash`-feature (feature abbreviated as /), cf. example (9). From these static values, the dynamic `inherited:slash`-values (feature abbreviated as //) can be calculated during generation, see rule *(lex)* in Fig.7.

(1a) Choose a lexical entry as the head x_h of the current goal x_0. Then the substructure condition must hold for the corresponding semantic forms X_h and X_0. The //-value T_h must be empty.

(1b) Or choose an element of the //-value T_0 of the current head x_0. Then the //-value T_h becomes $[\langle x_h, X_h \rangle]$. The associated string w_h is empty.

(2) There must be a lexicalized tree which connects the goal x_0 and the chosen head x_h. The //-value T_0 is split into disjoint sets $T_1, \ldots, T_n$. The //-values of the new subgoals $x_1, \ldots, x_n$ are the disjoint set unions $T_i \uplus T_i'$ where T_i' is the /-value of x_i in the local tree given in the grammar.

Note that this version of the Non-Local-Feature principle corresponds to the hypothetical reasoning mechanism which is provided by the Lambek categorial grammars [Lambek, 1958], [König, 1994]. This is illustrated by the fact that e.g. the left tree in example (9) can be rendered in categorial grammar notation as $cp_f/(vp/v)$. Hence, the algorithm in Fig.7 has a clear logical basis.

5 Conclusion

This paper gives a syntactic-head-driven generation algorithm which includes a well-defined treatment of moved constituents. Since it relies on the notion of a syntactic head instead of a semantic head it works also for grammars where semantic heads are not available in general, like for a grammar which includes semantic decomposition rules of (scopally) Underspecified Discourse Representation Structures. By using the same notion of head both for parsing and for generation, both techniques become even closer. In effect, the abstract specifications of the generation algorithms which we gave above, could be read as parsing algorithms, modulo a few changes (of the success condition and the link relation).

Generation from Underspecified DRS's means that sentences can be generated from meaning representations which have not been disambiguated with regard to quantifier scope. This is of particular importance for applications in machine translation, where

[1] The CUF-system is an implementation of a theorem prover for a Horn clause logic with typed feature terms [Dörre and Dorna, 1993].

[2] An instance of our head-corner generator (without an integrated treatment of movement) is the UCG-generator by Calder et al. [Calder *et al.*, 1989] (modulo the use of unary category transformation rules) which relies, in addition, on the symmetry of syntactic and semantic head. A syntactic-head-driven generator for a kind of lexicalized grammars has been proposed independently by [Kay, 1993]. Another variant of a lexicalized grammar by [Dymetman *et al.*, 1990] does not make use of the head-corner idea but rather corresponds to the top-down generation schema presented in Fig.1.

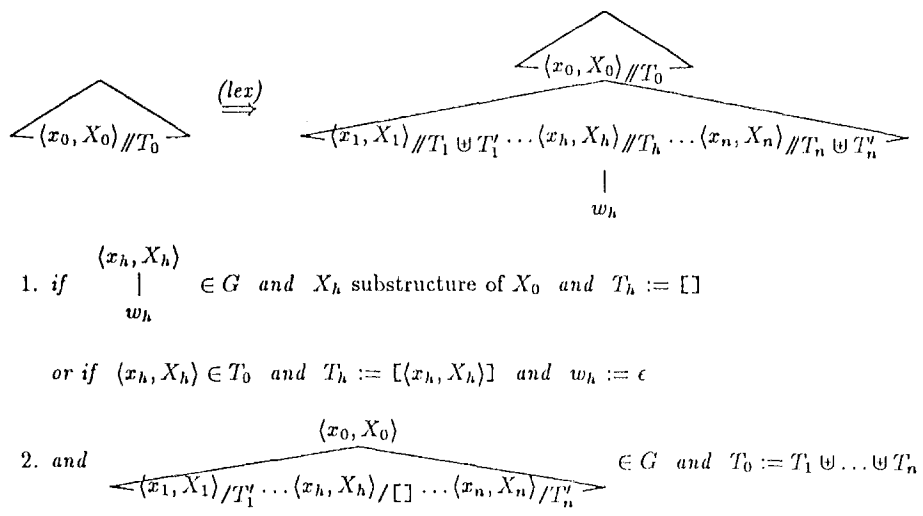


Figure 7: Head-Corner Generator for lexicalized grammars (G grammar description; x_i syntactic category symbol; X_i semantic representation; T_i `slash`-values)

one wants to avoid the resolution of scope relations as long as the underspecified meaning can be rendered in the source and in the target language. Future work should consider more the strategic part of the generation problem, e.g. try to find heuristics and strategies which handle situations of 'scope mismatch' where one language has to be more precise with regard to scope than the other.

References

[Bar-Hillel *et al.*, 1960] Yehoshua Bar-Hillel, Chaim Gaifman, and E. Shamir. On categorial and phrase structure grammars. *Bull. Res. Council Israel, Sec. F.*, 9:1–16, 1960.

[Calder *et al.*, 1989] Jonathan Calder, Mike Reape, and Henk Zeevat. An algorithm for generation in Unification Categorial Grammars. EACL, 1989.

[Dörre and Dorna, 1993] Jochen Dörre and Michael Dorna. CUF - a formalism for linguistic knowledge representation. Deliverable R.1.2A, DYANA 2, August 1993.

[Dymetman *et al.*, 1990] Dymetman, Isabelle, and Perrault. A symmetrical approach to parsing and generation. COLING, 1990.

[Frank and Reyle, 1992] Anette Frank and Uwe Reyle. How to cope with scrambling and scope. KONVENS, 1992.

[Gazdar *et al.*, 1985] Gerald Gazdar, Ewan Klein, G.K. Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Blackwell, Oxford, UK, 1985.

[Kamp and Reyle, 1993] Hans Kamp and Uwe Reyle. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Studies in Linguistics and Philosophy 42. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[Kay, 1993] Martin Kay. Machine translation. Lecture Notes, Institut für Maschinelle Sprachverarbeitung, June 1993.

[Kohl, 1992] Dieter Kohl. Generation from under- and over-specified structures. COLING, 1992.

[König, 1994] Esther König. A hypothetical reasoning algorithm for linguistic analysis. *Journal of Logic and Computation*, 4(1), 1994.

[Lambek, 1958] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[Minnen *et al.*, 1993] Guido Minnen, Dale Gerdemann, and Erhard Hinrichs. Direct automated inversion of logic grammars. In Yuji Matsumoto, editor, *Proceedings of the Fourth International Workshop on Natural Language Understanding and Logic Programming*, pages 17–36, Nara, Japan, 1993. Nara Institute of Science and Technology.

[Pollard and Sag, 1993] Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar.* University of Chicago Press, Chicago, 1993.

[Reyle, 1993] Uwe Reyle. Dealing with ambiguities by underspecification: Construction, representation, and deduction. *Journal of Semantics*, 10(2), 1993.

[Schabes and Waters, 1993] Yves Schabes and Richard Waters. Lexicalized context-free grammars. ACL, 1993.

[Shieber and Schabes, 1991] Stuart M. Shieber and Yves Schabes. Generation and synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 7(4):220–228, 1991.

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

[Shieber, 1988] Stuart M. Shieber. A uniform architecture for parsing and generation. COLING, 1988.

[Strzalkowski, 1990] Tomek Strzalkowski. How to invert a natural language parser into an efficient generator: An algorithm for logic grammars. COLING, 1990.

[van Noord, 1993] Gertjan van Noord. *Reversibility in Natural Language Processing.* PhD thesis, University of Utrecht, Utrecht, Netherlands, 1993.

[Wedekind, 1988] Jürgen Wedekind. Generation as structure driven derivation. COLING, 1988.