Compilation of HPSG to TAG*

Robert Kasper
Dept. of Linguistics
Ohio State University
222 Oxley Hall
Columbus, OH 43210
U.S.A.
kasper@ling.ohio-state.edu

Bernd Kiefer **Klaus Netter**
Deutsches Forschungszentrum
für Künstliche Intelligenz, GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany
{kiefer|netter}@dfki.uni-sb.de

K. Vijay-Shanker
CIS Dept.
University of Delaware
Newark, DE 19716
U.S.A
vijay@cis.udel.edu

Abstract

We present an implemented compilation algorithm that translates HPSG into lexicalized feature-based TAG, relating concepts of the two theories. While HPSG has a more elaborated principle-based theory of possible phrase structures, TAG provides the means to represent lexicalized structures more explicitly. Our objectives are met by giving clear definitions that determine the projection of structures from the lexicon, and identify "maximal" projections, auxiliary trees and foot nodes.

1 Introduction

Head Driven Phrase Structure Grammar (HPSG) and Tree Adjoining Grammar (TAG) are two frameworks which so far have been largely pursued in parallel, taking little or no account of each other. In this paper we will describe an algorithm which will compile HPSG grammars, obeying certain constraints, into TAGs. However, we are not only interested in mapping one formalism into another, but also in exploring the relationship between concepts employed in the two frameworks.

HPSG is a feature-based grammatical framework which is characterized by a modular specification of linguistic generalizations through extensive use of principles and lexicalization of grammatical information. Traditional grammar rules are generalized to schemata providing an abstract definition of grammatical relations, such as head-of, complement-of, subject-of, adjunct-of, etc. Principles, such as the Head-Feature-, Valence-, Non-Local- or Semantics-Principle, determine the projection of information from the lexicon and recursively define the flow of information in a global structure. Through this modular design, grammatical descriptions are broken down into minimal structural units referring to local trees of depth one, jointly constraining the set of well-formed sentences.

In HPSG, based on the concept of "head-domains", *local* relations (such as complement-of, adjunct-of) are defined as those that are realized within the domain defined by the syntactic head. This domain is usually the maximal projection of the head, but it may be further extended in some cases, such as raising constructions. In contrast, filler-gap relations are considered *non-local*. This local vs. non-local distinction in HPSG cuts across the relations that are localized in TAG via the domains defined by elementary trees. Each elementary tree typically represents all of the arguments that are dependent on a lexical functor. For example, the complement-of and filler-gap relations are localized in TAG, whereas the adjunct-of relation is not.

Thus, there is a fundamental distinction between the different notions of localization that have been assumed in the two frameworks. If, at first sight, these frameworks seem to involve a radically different organization of grammatical relations, it is natural to question whether it is possible to compile one into the other in a manner faithful to both, and more importantly, why this compilation is being explored at all. We believe that by combining the two approaches both frameworks will profit.

From the HPSG perspective, this compilation offers the potential to improve processing efficiency. HPSG is a "lexicalist" framework, in the sense that the lexicon contains the information that determines which specific categories can be combined. However, most HPSG grammars are not lexicalized in the stronger sense defined by Schabes et.al. (SAJ88), where lexicalization means that each elementary structure in the grammar is anchored by some lexical item. For example, HPSG typically assumes a rule schema which combines a subject phrase (e.g.

*We would like to thank A. Abeillé, D. Flickinger, A. Joshi, T. Kroch, O. Rambow, I. Sag and H. Uszkoreit for valuable comments and discussions. The research underlying the paper was supported by research grants from the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) to the DFKI projects DISCO, FKZ ITW 9002 0, PARADICE, FKZ ITW 9403 and the VERBMOBIL project, FKZ 01 IV 101 K/1, and by the Center for Cognitive Science at Ohio State University.

NP) with a head phrase (e.g. VP), neither of which is a lexical item. Consider a sentence involving a transitive verb which is derived by applying two rule schemata, reducing first the object and then the subject. In a standard HPSG derivation, once the head verb has been retrieved, it must be computed that these two rules (and no other rules) are applicable, and then information about the complement and subject constituents is projected from the lexicon according to the constraints on each rule schema. On the other hand, in a lexicalized TAG derivation, a tree structure corresponding to the combined instantiation of these two rule schemata is directly retrieved along with the lexical item for the verb. Therefore, a procedure that compiles HPSG to TAG can be seen as performing significant portions of an HPSG derivation at compile-time, so that the structures projected from lexical items do not need to be derived at run-time. The compilation to TAG provides a way of producing a strongly lexicalized grammar which is equivalent to the original HPSG, and we expect this lexicalization to yield a computational benefit in parsing (cf. (SJ90)).

This compilation strategy also raises several issues of theoretical interest. While TAG belongs to a class of mildly context-sensitive grammar formalisms (JVW91), the generative capacity of the formalism underlying HPSG (viz., recursive constraints over typed feature structures) is unconstrained, allowing any recursively enumerable language to be described. In HPSG the constraints necessary to characterize the class of natural languages are stated within a very expressive formalism, rather than built into the definition of a more restrictive formalism, such as TAG. Given the greater expressive power of the HPSG formalism, it will not be possible to compile an aribitrary HPSG grammar into a TAG grammar. However, our compilation algorithm shows that particular HPSG grammars may contain constraints which have the effect of limiting the generative capacity to that of a mildly context-sensitive language.[1] Additionally, our work provides a new perspective on the different types of constituent combination in HPSG, enabling a classification of schemata and principles in terms of more abstract functor-argument relations.

From a TAG perspective, using concepts employed in the HPSG framework, we provide an explicit method of determining the content of the elementary trees (e.g., what to project from lexical items and when to stop the projection) from an HPSG source specification. This also provides a method for deriving the distinctions between initial and auxiliary trees, including the identification of foot nodes in auxiliary trees. Our answers, while consistent with basic tenets of traditional TAG analyses, are general enough to allow an alternate linguistic theory, such as HPSG, to be used as a basis for deriving a TAG. In this manner, our work also serves to investigate the utility of the TAG framework itself as a means of expressing different linguistic theories and intuitions.

In the following we will first briefly describe the basic constraints we assume for the HPSG input grammar and the resulting form of TAG. Next we describe the essential algorithm that determines the projection of trees from the lexicon, and give formal definitions of auxiliary tree and foot node. We then show how the computation of "sub-maximal" projections can be triggered and carried out in a two-phase compilation.

2 Background

As the target of our translation we assume a Lexicalized Tree-Adjoining Grammar (LTAG), in which every elementary tree is anchored by a lexical item (SAJ88).

We do not assume atomic labelling of nodes, unlike traditional TAG, where the root and foot nodes of an auxiliary tree are assumed to be labelled identically. Such trees are said to factor out recursion. However, this identity itself isn't sufficient to identify foot nodes, as more than one frontier node may be labelled the same as the root. Without such atomic labels in HPSG, we are forced to address this issue, and present a solution that is still consistent with the notion of factoring recursion.

Our translation process yields a lexicalized feature-based TAG (VSJ88) in which feature structures are associated with nodes in the frontier of trees and two feature structures (top and bottom) with nodes in the interior. Following (VS92), the relationships between such top and bottom feature structures represent underspecified domination links. Two nodes standing in this domination relation could become the same, but they are necessarily distinct if adjoining takes place. Adjoining separates them by introducing the path from the root to the foot node of an auxiliary tree as a further specification of the underspecified domination link.

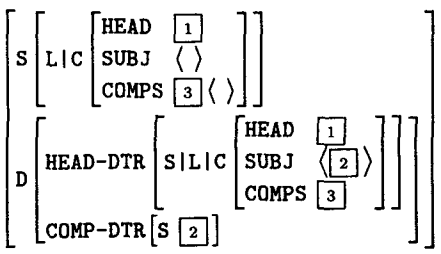
For illustration of our compilation, we consider an extended HPSG following the specifications in (PS94)[404ff]. The rule schemata include rules for complementation (including head-subject and head-complement relations), head-adjunct, and filler-head relations.

The following rule schemata cover the combination of heads with subjects and other complements respectively as well as the adjunct constructions.[2]

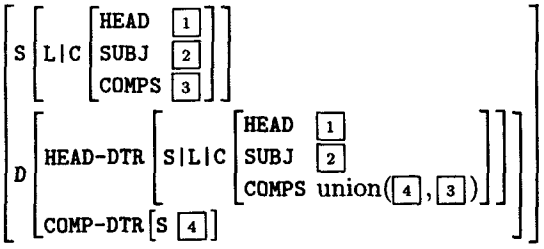
[1] We are only considering a syntactic fragment of HPSG here. It is not clear whether the semantic components of HPSG can also be compiled into a more constrained formalism.

[2] We abstract from quite a number of properites and use the following abbreviations for feature names: S=SYNSEM, L=LOCAL, C=CAT, N-L=NON-LOCAL, D=DTRS.

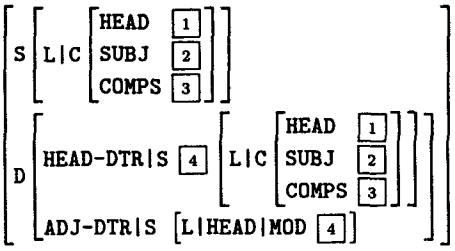
Head-Subj-Schema



Head-Comps-Schema

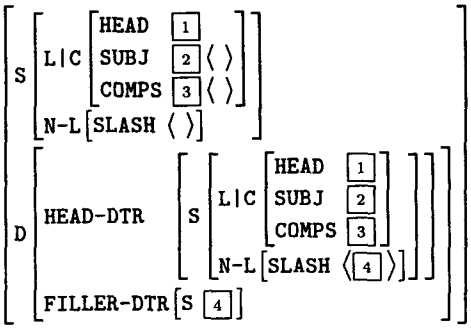


Head-Adjunct-Schema



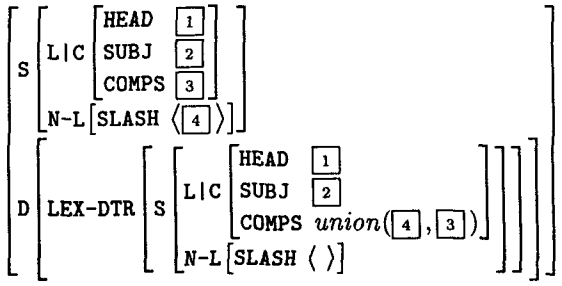
We assume a slightly modified and constrained treatment of non-local dependencies (SLASH), in which empty nodes are eliminated and a lexical rule is used instead. While SLASH introduction is based on the standard filler-head schema, SLASH percolation is essentially constrained to the HEAD spine.

Head-Filler-Schema



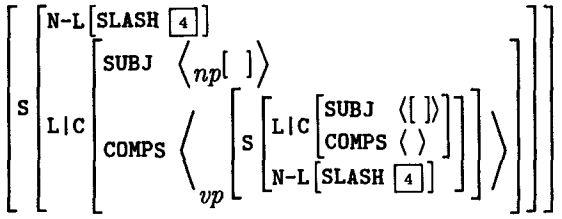
SLASH termination is accounted for by a lexical rule, which removes an element from one of the valence lists (COMPS or SUBJ) and adds it to the SLASH list.

Lexical Slash-Termination-Rule



The percolation of SLASH across head domains is lexically determined. Most lexical items will be specified as having an empty SLASH list. Bridge verbs (e.g., equi verbs such as *want*) or other heads allowing extraction out of a complement share their own SLASH value with the SLASH of the respective complement.[3]

Equi and Bridge Verb



Finally, we assume that rule schemata and principles have been compiled together (automatically or manually) to yield more specific subtypes of the schemata. This does not involve a loss of generalization but simply means a further refinement of the type hierarchy. LP constraints could be compiled out beforehand or during the compilation of TAG structures, since the algorithm is lexicon driven.

3 Algorithm

3.1 Basic Idea

While in TAG all arguments related to a particular functor are represented in one elementary tree structure, the 'functional application' in HPSG is distributed over the phrasal schemata, each of which can be viewed as a partial description of a local tree. Therefore we have to identify which constituents in

[3]We choose such a lexicalized approach, because it will allow us to maintain a restriction that every TAG tree resulting from the compilation must be rooted in a non-emtpy lexical item. The approach will account for extraction of complements out of complements, i.e., along paths corresponding to chains of government relations.

As far as we can see, the only limitation arising from the percolation of SLASH only along head-projections is on extraction out of adjuncts, which may be desirable for some languages like English. On the other hand, these constructions would have to be treated by multi-component TAGs, which are not covered by the intended interpretation of the compilation algorithm anyway.

a phrasal schema count as functors and arguments. In TAG different functor argument relations, such as head-complement, head-modifier etc., are represented in the same format as branches of a trunk projected from a lexical anchor. As mentioned, this anchor is not always equivalent to the HPSG notion of a head; in a tree projected from a modifier, for example, a non-head (ADJUNCT-DTR) counts as a functor. We therefore have to generalize over different types of daughters in HPSG and define a general notion of a functor. We compute the functor-argument structure on the basis of a general *selection* relation. Following (Kas92)[4], we adopt the notion of a *selector daughter* (SD), which contains a *selector feature* (SF) whose value constrains the argument (or non-selector) daughter (non-SD).[5] For example, in a head-complement structure, the SD is the HEAD-DTR, as it contains the list-valued feature COMPS (the SF) each of whose elements selects a COMP-DTR, i.e., an element of the COMPS list is identified with the SYNSEM value of a COMP-DTR.

We assume that a *reduction* takes place along with selection. Informally, this means that if F is the selector feature for some schema, then the value (or the element(s) in the list-value) of F that selects the non-SD(s) is not contained in the F value of the mother node. In case F is list-valued, we assume that the rest of the elements in the list (those that did not select any daughter) are also contained in the F at the mother node. Thus we say that F has been reduced by the schema in question.

The compilation algorithm assumes that all HPSG schemata will satisfy the condition of simultaneous selection and reduction, and that each schema reduces at least one SF. For the head-complement- and head-subject-schema, these conditions follow from the Valence Principle, and the SFs are COMPS and SUBJ, respectively. For the head-adjunct-schema, the ADJUNCT-DTR is the SD, because it selects the HEAD-DTR by its MOD feature. The MOD feature is reduced, because it is a head feature, whose value is inherited only from the HEAD-DTR and not from the ADJUNCT-DTR. Finally, for the filler-head-schema, the HEAD-DTR is the SD, as it selects the FILLER-DTR by its SLASH value, which is bound off, not inherited by the mother, and therefore reduced.

We now give a general description of the compilation process. Essentially, we begin with a lexical description and project phrases by using the schemata to reduce the selection information specified by the lexical type.

Basic Algorithm Take a lexical type L and initialize by creating a node with this type. Add a node n dominating this node.

For any schema S in which specified SFs of n are reduced, try to instantiate S with n corresponding to the SD of S. Add another node m dominating the root node of the instantiated schema. (The domination links are introduced to allow for the possibility of adjoining.) Repeat this step (each time with n as the root node of the tree) until no further reduction is possible.

We will fill in the details below in the following order: what information to raise across domination links (where adjoining may take place), how to determine auxiliary trees (and foot nodes), and when to terminate the projection.

We note that the trees produced have a *trunk* leading from the lexical anchor (node for the given lexical type) to the root. The nodes that are siblings of nodes on the trunk, the selected daughters, are not elaborated further and serve either as foot nodes or substitution nodes.

3.2 Raising Features Across Domination Links

Quite obviously, we *must* raise the SFs across domination links, since they determine the applicability of a schema and licence the instantiation of an SD. If no SF were raised, we would lose all information about the saturation status of a functor, and the algorithm would terminate after the first iteration.

There is a danger in raising more than the SFs. For example, the head-subject-schema in German would typically constrain a verbal head to be finite. Raising HEAD features would block its application to non-finite verbs and we would not produce the trees required for raising-verb adjunction. This is again because *heads* in HPSG are not equivalent to *lexical anchors* in TAG, and that other local properties of the top and bottom of a domination link could differ. Therefore HEAD features and other LOCAL features cannot, in general, be raised across domination links, and we assume for now that only the SFs are raised.

Raising all SFs produces only fully saturated elementary trees and would require the root and foot of any auxiliary tree to share all SFs, in order to be compatible with the SF values across any domination links where adjoining can take place. This is too strong a condition and will not allow the resulting TAG to generate all the trees derivable with the given HPSG (e.g., it would not allow unsaturated VP complements). In § 3.5 we address this concern by

[4]The algorithm presented here extends and refines the approach described by (Kas92) by stating more precise criteria for the projection of features, for the termination of the algorithm, and for the determination of those structures which should actually be used as elementary trees.

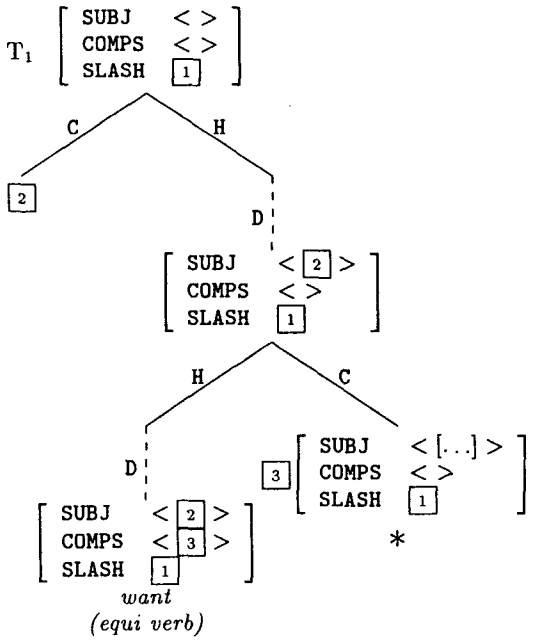
[5]Note that there might be mutual selection (as in the case of the specifier-head-relations proposed in (PS94)[44ff]). If there is mutual selection, we have to stipulate one of the daughters as the SD. The choice made would not effect the correctness of the compilation.

using a multi-phase compilation. In the first phase, we raise all the SFs.

3.3 Detecting Auxiliary Trees and Foot Nodes

Traditionally, in TAG, auxiliary trees are said to be minimal recursive structures that have a foot node (at the frontier) labelled identical to the root. As such category labels (S, NP etc.) determine where an auxiliary tree can be adjoined, we can informally think of these labels as providing selection information corresponding to the SFs of HPSG. Factoring of recursion can then be viewed as saying that auxiliary trees define a path (called the *spine*) from the root to the foot where the nodes at extremities have the same selection information. However, a closer look at TAG shows that this is an oversimplification. If we take into account the adjoining constraints (or the top and bottom feature structures), then it appears that the root and foot share only some selection information.

Although the encoding of selection information by SFs in HPSG is somewhat different than that traditionally employed in TAG, we also adopt the notion that the extremities of the spine in an auxiliary tree share some part (but not necessarily all) of the selection information. Thus, once we have produced a tree, we examine the root and the nodes in its frontier. A tree is an auxiliary tree if the root and some frontier node (which becomes the foot node) have some non-empty SF value in common. Initial trees are those that have no such frontier nodes.



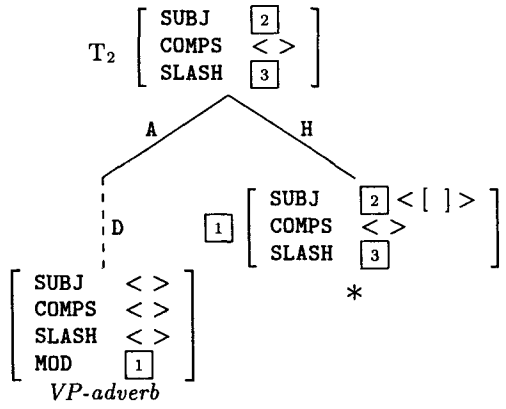
In the trees shown, nodes detected as foot nodes are marked with $*$. Because of the SUBJ and SLASH values, the HEAD-DTR is the foot of T_2 below (anchored by an adverb) and COMP-DTR is the foot of T_3 (anchored by a raising verb). Note that in the tree T_1 anchored by an equi-verb, the foot node is detected because the SLASH value is shared, although the SUBJ is not. As mentioned, we assume that bridge verbs, i.e., verbs which allow extraction out of their complements, share their SLASH value with their clausal complement.

3.4 Termination

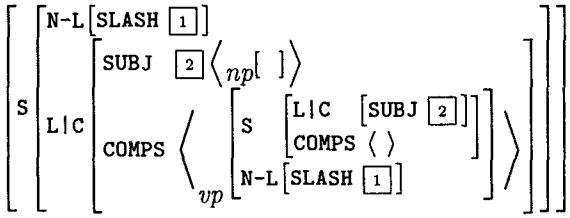
Returning to the basic algorithm, we will now consider the issue of termination, i.e., how much do we need to reduce as we project a tree from a lexical item.

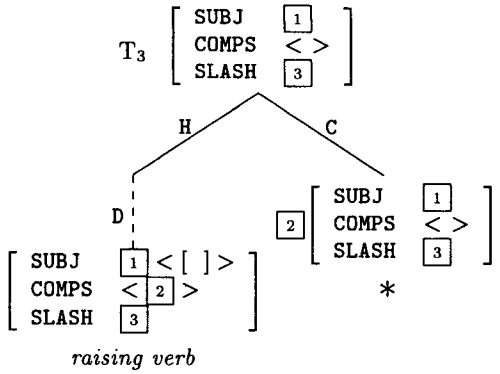
Normally, we expect a SF with a specified value to be reduced fully to an empty list by a series of applications of rule schemata. However, note that the SLASH value is unspecified at the root of the trees T_2 and T_3. Of course, such nodes would still unify with the SD of the filler-head-schema (which reduces SLASH), but applying this schema could lead to an infinite recursion. Applying a reduction to an unspecified SF is also linguistically unmotivated as it would imply that a functor could be applied to an argument that it never explicitly selected.

However, simply blocking the reduction of a SF whenever its value is unspecified isn't sufficient. For example, the root of T_2 specifies the SUBJ to be a non-empty list. Intuitively, it would not be appropriate to reduce it further, because the lexical anchor (adverb) doesn't semantically license the SUBJ argument itself. It merely constrains the modified head to have an unsaturated SUBJ.



Raising Verb (and Infinitive Marker to*)*





raising verb

To motivate our termination criterion, consider the adverb tree and the asterisked node (whose SLASH value is shared with SLASH at the root). Being a non-trunk node, it will either be a foot or a substitution node. In either case, it will eventually be unified with some node in another tree. If that other node has a reducible SLASH value, then we know that the reduction takes place in the other tree, because the SLASH value must have been raised across the domination link where adjoining takes place. As the same SLASH (and likewise SUBJ) value should not be reduced in both trees, we state our termination criteria as follows:

Termination Criterion The value of an SF F at the root node of a tree is not reduced further if it is an empty list, or if it is shared with the value of F at some non-trunk node in the frontier.

Note that because of this termination criterion, the adverb tree projection will stop at this point. As the root shares some selector feature values (SLASH and SUBJ) with a frontier node, this node becomes the foot node. As observed above, adjoining this tree will preserve these values across any domination links where it might be adjoined; and if the values stated there are reducible then they will be reduced in the other tree. While auxiliary trees allow arguments selected at the root to be realized elsewhere, it is never the case for initial trees that an argument selected at the root can be realized elsewhere, because by our definition of initial trees the selection of arguments is not passed on to a node in the frontier.

We also obtain from this criterion a notion of *local completeness*. A tree is locally complete as soon as all arguments which it licenses and which are not licensed elsewhere are realized. *Global completeness* is guaranteed because the notion of "elsewhere" is only and always defined for auxiliary trees, which have to adjoin into an initial tree.

3.5 Additional Phases

Above, we noted that the preservation of some SFs along a path (realized as a path from the root to the foot of an auxiliary tree) does not imply that all SFs need to be preserved along that path. Tree T_1 provides such an example, where a lexical item, an equi-verb, triggers the reduction of an SF by taking a complement that is unsaturated for SUBJ but never shares this value with one of its own SF values.

To allow for adjoining of auxiliary trees whose root and foot differ in their SFs, we could produce a number of different trees representing partial projections from each lexical anchor. Each partial projection could be produced by raising some subset of SFs across each domination link, instead of raising all SFs. However, instead of systematically raising all possible subsets of SFs across domination links, we can avoid producing a vast number of these partial projections by using auxiliary trees to provide guidance in determining when we need to raise only a particular subset of the SFs.

Consider T_1 whose root and foot differ in their SFs. From this we can infer that a SUBJ SF should not always be raised across domination links in the trees compiled from this grammar. However, it is only useful to produce a tree in which the SUBJ value is not raised when the bottom of a domination link has both a one element list as value for SUBJ and an empty COMPS list. Having an empty SUBJ list at the top of the domination link would then allow for adjunction by trees such as T_1.

This leads to the following multi-phase compilation algorithm. In the first phase, all SFs are raised. It is determined which trees are auxiliary trees, and then the relationships between the SFs associated with the root and foot in these auxiliary trees are recorded. The second phase begins with lexical types and considers the application of sequences of rule schemata as before. However, immediately after applying a rule schema, the features at the bottom of a domination link are compared with the foot nodes of auxiliary trees that have differing SFs at foot and root. Whenever the features are compatible with such a foot node, the SFs are raised according to the relationship between the root and foot of the auxiliary tree in question. This process may need to be iterated based on any new auxiliary trees produced in the last phase.

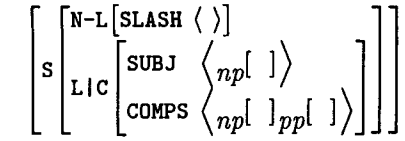
3.6 Example Derivation

In the following we provide a sample derivation for the sentence

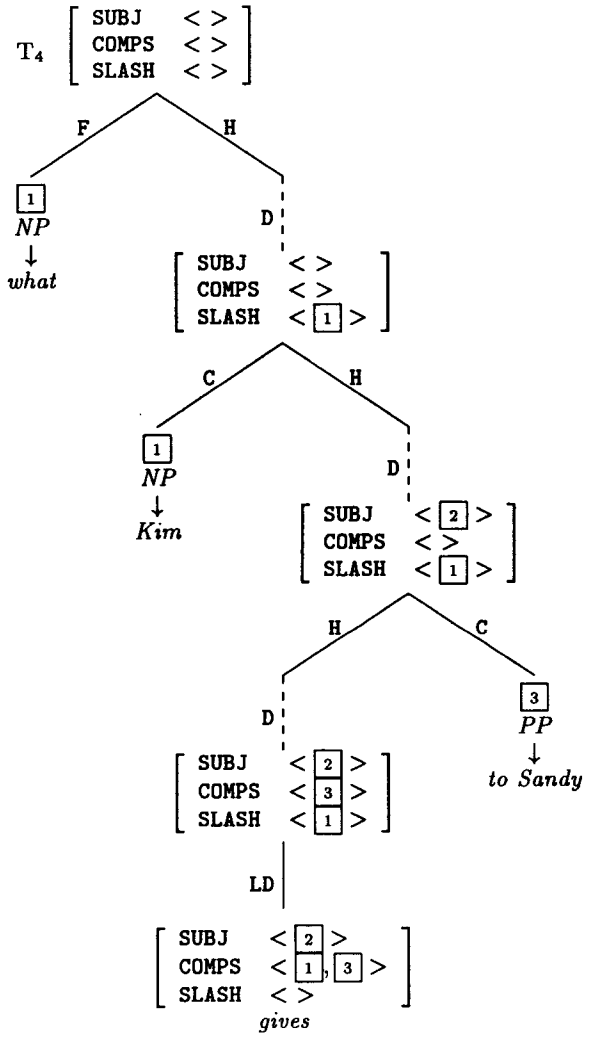
(I know) what Kim wants to give to Sandy.

Most of the relevant HPSG rule schemata and lexical entries necessary to derive this sentence were already given above. For the noun phrases *what*, *Kim* and *Sandy*, and the preposition *to* no special assumptions are made. We therefore only add the entry for the ditransitive verb *give*, which we take to subcategorize for a subject and two object complements.

Ditransitive Verb

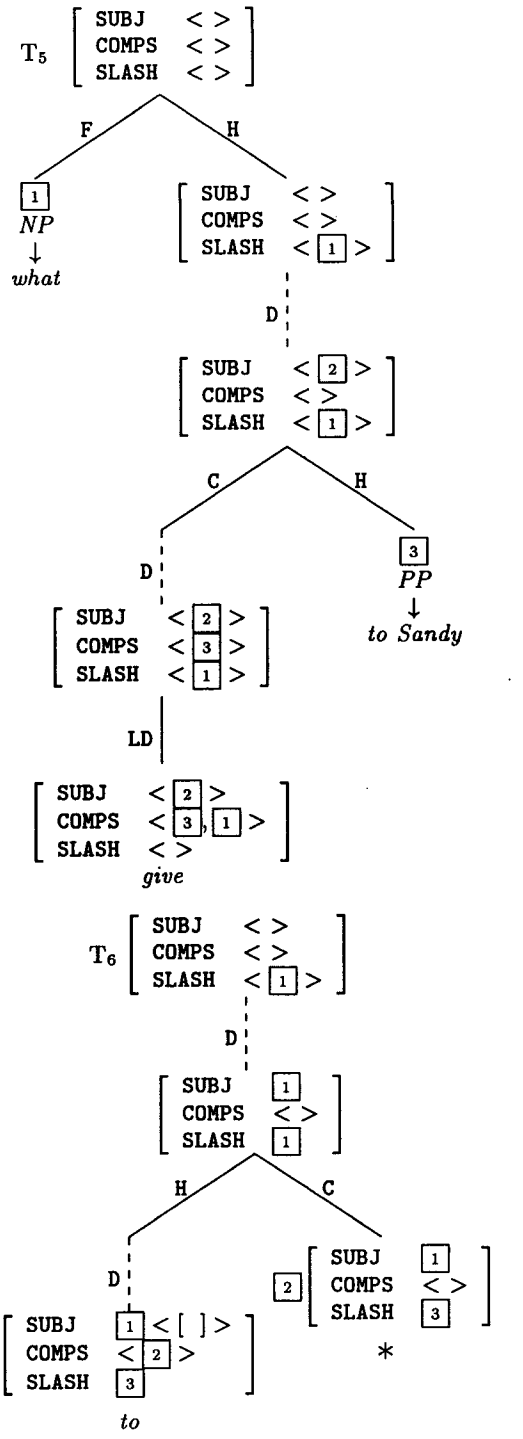


From this lexical entry, we can derive in the first phase a fully saturated initial tree by applying first the lexical slash-termination rule, and then the head-complement-, head-subject and filler-head-rule. Substitution at the nodes on the frontier would yield the string *what Kim gives to Sandy*.

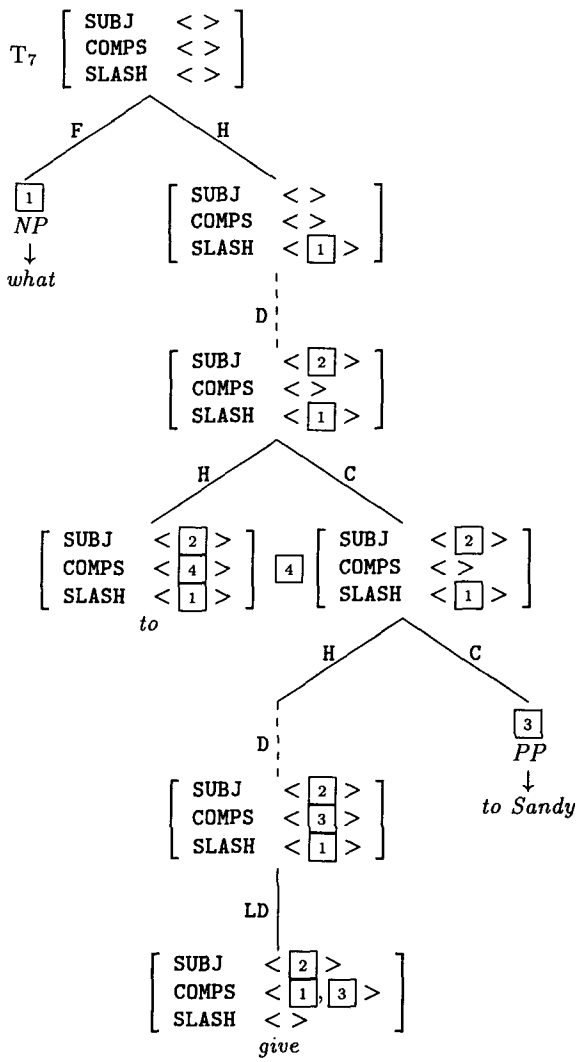


The derivations for the trees for the matrix verb *want* and for the infinitival marker *to* (equivalent to a raising verb) were given above in the examples T_1 and T_3. Note that the SUBJ feature is only reduced in the former, but not in the latter structure.

In the second phase we derive from the entry for *give* another initial tree (T_5) into which the auxiliary tree T_1 for *want* can be adjoined at the topmost domination link. We also produce a second tree with similar properties for the infinitive marker *to* (T_6).



By first adjoining the tree T_6 at the topmost domination link of T_5 we obtain a structure T_7 corresponding to the substring *what ... to give to Sandy*. Adjunction involves the identification of the foot node with the bottom of the domination link and identification of the root with top of the domination link. Since the domination link at the root of the adjoined tree mirrors the properties of the adjunction site in the initial tree, the properties of the domination link are preserved.



The final derivation step then involves the adjunction of the tree for the equi verb into this tree, again at the topmost domination link. This has the effect of inserting the substring *Kim wants* into *what ... to give to Sandy*.

4 Conclusion

We have described how HPSG specifications can be compiled into TAG, in a manner that is faithful to both frameworks. This algorithm has been implemented in Lisp and used to compile a significant fragment of a German HPSG. Work is in progress on compiling an English grammar developed at CSLI.

This compilation strategy illustrates how linguistic theories other than those previously explored within the TAG formalism can be instantiated in TAG, allowing the association of structures with an enlarged domain of locality with lexical items. We have generalized the notion of factoring recursion in TAG, by defining auxiliary trees in a way that is not only adequate for our purposes, but also provides a uniform treatment of extraction from both clausal and non-clausal complements (e.g., VPs) that is not possible in traditional TAG.

It should be noted that the results of our compilation will not always conform to conventional linguistic assumptions often adopted in TAGs, as exemplified by the auxiliary trees produced for equi verbs. Also, as the algorithm does not currently include any downward expansion from complement nodes on the frontier, the resulting trees will sometimes be more fractioned than if they had been specified directly in a TAG.

We are currently exploring the possiblity of compiling HPSG into an extension of the TAG formalism, such as D-tree grammars (RVW95) or the UVG-DL formalism (Ram94). These somewhat more powerful formalisms appear to be adequate for some phenomena, such as extraction out of adjuncts (recall §2) and certain kinds of scrambling, which our current method does not handle. More flexible methods of combining trees with dominance links may also lead to a reduction in the number of trees that must be produced in the second phase of our compilation.

There are also several techniques that we expect to lead to improved parsing efficiency of the resulting TAG. For instance, it is possible to declare specific non-SFs which can be raised, thereby reducing the number of useless trees produced during the multiphase compilation. We have also developed a scheme to effectively organize the trees associated with lexical items.

References

Robert Kasper. On Compiling Head Driven Phrase Structure Grammar into Lexicalized Tree Adjoining Grammar. In *Proceedings of the 2nd Workshop on TAGs*, Philadelphia, 1992.

A. K. Joshi, K. Vijay-Shanker and D. Weir. The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, eds., *Foundational Issues in Natural Language Processing.* MIT Press, 1991.

Carl Pollard and Ivan Sag. *Head Driven Phrase Structure Grammar.* CSLI, Stanford & University of Chicago Press, 1994.

O. Rambow. Formal and Computational Aspects of Natural Language Syntax. Ph.D. thesis. Univ. of Philadelphia. Philadelphia, 1994.

O. Rambow, K. Vijay-Shanker and D. Weir. D-Tree Grammars. In: ACL-95.

Y. Schabes, A. Abeille, and A. K. Joshi. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. COLING-88, pp. 578–583.

Y. Schabes, and A. K. Joshi. Parsing with lexicalized tree adjoining grammar. In M. Tomita, ed., *Current Issues in Parsing Technologies.* Kluwer Academic Publishers, 1990.

K. Vijay-Shanker. Using Descriptions of Trees in a TAG. *Computational Linguistics*, 18(4):481–517, 1992.

K. Vijay-Shanker and A. K. Joshi. Feature Structure Based Tree Adjoining Grammars. In: COLING-88.