Fully Lexicalized Head–Driven Syntactic Generation

Tilman Becker

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
`becker@dfki.de`

Abstract

We describe a new approach to syntactic generation with Head-Driven Phrase Structure Grammars (HPSG) that uses an extensive off–line preprocessing step. Direct generation algorithms apply the phrase-structure rules (schemata) of the grammar on–line which is an computationally expensive step. Instead, we collect off-line for every lexical type of the HPSG grammar all minimally complete projections (called elementary trees) that can be derived with the schemata. This process is known as 'compiling HPSG to TAG' and derives a Lexicalized Tree-Adjoining Grammar (LTAG). The representation as an LTAG is 'fully lexicalized' in the sense that all grammatical information is directly encoded with the lexical item (as a set of elementary trees) and the combination operations are reduced from schema applications to the TAG primitives of adjunction and substitution. Given this LTAG, the generation task has a very different search space that can be traversed very efficiently, avoiding the costly on–line applications of HPSG unification. The entire generation task from a semantic representation to a surface string is split into two tasks, a microplanner and a syntactic realizer. This paper discusses the syntactic generator and the preprocessing steps as implemented in the Verbmobil system.

1 Generation in a Speech–to–Speech System

The syntactic generation algorithm and the preprocessing steps presented in this paper are integrated into the Verbmobil system (see [Wahlster 1993, Bub, Wahlster, and Waibel 1997]). It is a system for speech–to–speech dialog translation. The input for the generation module VM–GECO[1] is generated by a semantic–based transfer component (see [Dorna and Emele 1996]). The interface language chosen comprises the encoding of target language–specific semantic information in a combination of Underspecified Discourse Representation Theory and Minimal Recursion Semantics (see [Bos et al. 1996] and [Copestake, Flickinger, and Sag 1997]).
The internal architecture of the generation module is modularized: it is separated into two phases, a *microplanner* and a *syntactic generator*. Throughout the system, we emphasize *declarativity*, which is also a necessary precondition for a comprehensive off–line preprocessing of external knowledge bases–in particular the preprocessing of the underlying Head–Driven Phrase Structure Grammar (HPSG, see [Pollard and Sag 1994]) which has been developed at CSLI, reflecting the latest developments in the linguistic theory and with a fairly wide coverage and also covering phenomena of spoken language.

[1]**V**erb**M**obil **GE**neration **CO**mponents

2 Microplanning and Syntactic Generation

Starting from the semantic representation, the microplanning component generates an annotated *dependency structure* which is used by the syntactic generation component to realize a surface string. The microplanner also carries out word–choice.
One goal of this modularization is a stepwise constraining of the search–space of alternative linguistic realizations, using different views in the different modules. In each step, only an abstraction of the multitude of information contained in an alternative needs to be considered.
Another aspect of this architecture is the separation into a *kernel system*, i.e., the language independent core algorithms (a constraint-solver for microplanning and the search and combination algorithms for syntactic generation described in section 5) and declarative knowledge bases, e.g., the language specific word-choice constraints in microplanning and the TAG grammars used in syntactic realization. This separation allows for an easy adaptation of the system to other languages and domains (see [Becker et al. 1998]).

3 Declarativity in the Syntactic Generator

All modules of the generator utilize external, declarative knowledge bases. For the syntactic generator, extensive off-line preprocessing of the highly declarative HPSG grammar for English[2] is applied. The grammar has not even been written exclusively as a generation grammar[3]. It is specialized, however, in that it covers phenomena of spoken language. The high level of abstraction which is achieved in the hierarchically organized grammar description (see [Flickinger 1987]) allows for easy maintenance as well as off-line preprocessing.
The off-line preprocessing steps described in the next section keep the declarative nature of the grammar intact, i.e. they retain explicitly the phrase structures and syntactic features as defined by the HPSG grammar.
In general, declarative knowledge bases allow for an easier adaptation of the system to other domains and languages. This is a huge benefit in the current second phase of the Verbmobil project [Becker et al. 1996] where the generator is extended to cover German, English and Japanese as well as additional and extended domains with a considerably larger vocabulary.

4 Off–Line Preprocessing: HPSG to TAG Compilation

The subtasks in a direct syntactic generator based on an HPSG grammar will always include the application of schemata (the HPSG equivalent of phrase structure rules) such that all syntactic constraints introduced by a lexical item (especially its SUBCAT list) are fulfilled. This results in a constant repetition of, e.g., building up the projection of a verb in a declarative sentence. In preprocessing the HPSG grammar we aim at computing all possible partial phrase structures which can be derived from the information in a lexicon entry. Given such sets of possible syntactic realization together with a set of selected lexicon entries for an utterance and finally their dependencies, the task of a syntactic generator is simplified considerably. Instead of exploring all

[2]The HPSG grammar is being developed at CSLI, Stanford University. Development is carried out on a grammar development platform which is based on TDL [Krieger and Schäfer 1994].

[3]In fact, most of the testing during grammar development depends on the use of a parser.

possible, computationally expensive applications of HPSG schemata, it merely has to find suitable precomputed syntactic structures for each lexical item and combine them appropriately.
For this preprocessing of the HPSG grammar, we adapted the 'HPSG to TAG compilation' process described in [Kasper et al. 1995]. The basis for the compilation is an identification of syntactically relevant *selector features* which express subcategorization requirements of a lexical item, e.g. the VALENCE features. In general, a phrase structure is complete when these selector features are empty.
Starting from the feature structure for a lexical item, HPSG schemata are applied such that the current structure is unified with a daughter feature of the schema. The resulting structure is again subject to this process. This compilation process stops when certain termination criteria are met, e.g., when all selector features are empty. Thus, all projections from the lexical item are collected as a set of minimally complete phrase structures which can also be interpreted as *elementary trees* of a Tree–Adjoining Grammar (TAG).
Instead of actually applying this compilation process to all lexical items, certain abstractions over the lexical entries are specified in the HPSG grammar. In fact, the needs of the compilation process have led to a clear–cut separation of lexical types and lexical entries as shown in Figure 1. A typical lexical entry is shown in Figure 2 and demonstrates that only three kinds of information are stored: the lexical type MV_NP_TRANS_LE[4], the semantic contribution (the relation _SUIT_REL) and morphological information (the stem and potentially irregular forms). By expanding the lexical type, the full feature structure can be obtained.

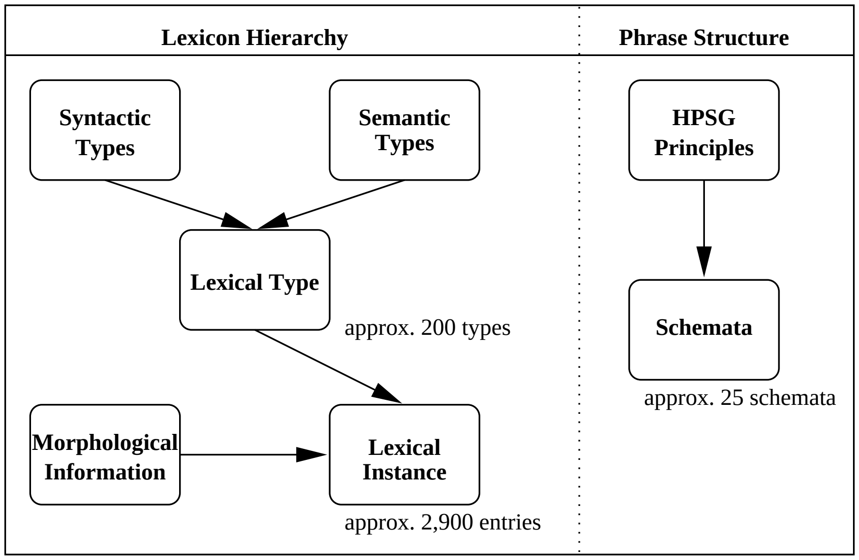


Figure 1: Organization of the HPSG grammar.

Some of the trees which result from the preprocessing of the lexical type MV_NP_TRANS_LE are shown in Figure 3. The figure shows only the phrase structure and an abstraction of the

[4]MV_NP_TRANS_LE is an abbreviation for "Main Verb, NP object, TRANSitive Lexical Entry" used in sentences like *"Monday suits me."*

```
suit_v1 := mv_np_trans_le &
  [ STEM < "suit" >,
    SYNSEM.LOCAL.CONT.STEMLISZT <! [ PRED _suit_rel ] !> ].
```

Figure 2: Specification of a lexical instance for the verb "suit."

node's categories. All nodes still represent the full HPSG feature structures. E.g., the tree MV_NP_TRANS_LE.2 of Figure 3 represents an imperative clause. As a consequence PERSON has the value SECOND and CL-MODE is set to IMPERATIVE. Note that the compilation process stopped at this node since the selector features are empty.

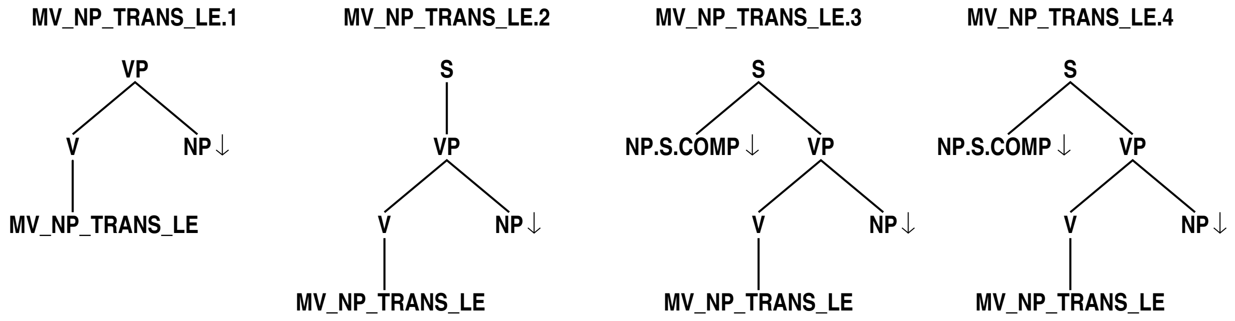
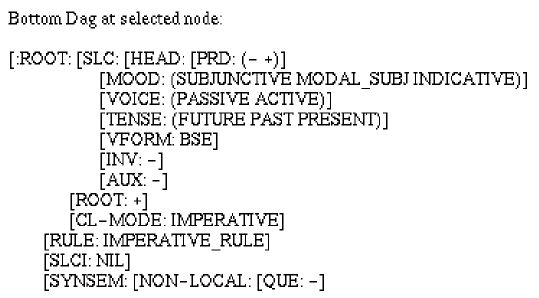


Figure 3: Some of the trees for transitive verbs. They are compiled from the corresponding lexical type MV_NP_TRANS_LE as defined in the HPSG grammar. Trees 3 and 4 differ only with respect to their feature structures which are not shown in this figure.

From these trees, two kinds of knowledge bases are built. For the microplanner, the relation between the lexical and syntactic realization and the semantic representation (encoded in the SYNSEM LOCAL CONT feature) is extracted as a constraint. For the syntactic generator, the relevant syntactic information is extracted in the form of a Feature–Based Lexicalized TAG (FB-LTAG) grammar, see [Joshi 1987, Vijay-Shanker and Joshi 1991, Schabes, Abeillé, and Joshi 1988]. This includes the phrase structure and a selected part of the feature structure (mainly the SYNSEM LOCAL CAT and SYNSEM NON-LOCAL features). Figure 4 shows the bottom feature structure extracted from the root node of MV_NP_TRANS_LE.2. Note that some of the feature paths are abbreviated, e.g. SLCI stands for SYNSEM LOCAL CONT INDEX. The elementary TAG trees which are built from the compilation result have so–called *restricted feature structures* which can be exploited for an efficient, specialized unification algorithm.
The node names shown in the figures represent a disjunction of possible categories, e.g. NP.S.COMP in tree MV_NP_TRANS_LE.3 implies that the subject of a transitive verb may be a nominal or sentential phrase.



```
Bottom Dag at selected node:

[:ROOT: [SLC: [HEAD: [PRD: (- +)]
              [MOOD: (SUBJUNCTIVE MODAL_SUBJ INDICATIVE)]
              [VOICE: (PASSIVE ACTIVE)]
              [TENSE: (FUTURE PAST PRESENT)]
              [VFORM: BSE]
              [INV: -]
              [AUX: -]
         [ROOT: +]
         [CL-MODE: IMPERATIVE]
     [RULE: IMPERATIVE_RULE]
     [SLCI: NIL]
     [SYNSEM: [NON-LOCAL: [QUE: -]
```

Figure 4: The bottom feature structure of the S node of tree MV_NP_TRANS_LE.2.

Finally, the leaf nodes of the trees (except for the lexical item itself) are marked either as substitution nodes or as a foot node, thus creating an auxiliary tree. In a TAG derivation, substitution nodes are replaced with trees bearing the correct category and a unifiable feature structure at their root node. Auxiliary trees can be inserted into other trees by the adjunction operation.

5 The Syntactic Generator VM-GIFT

The task of the syntactic generator is the construction of a sentence (or phrase, given the often incomplete utterances in spoken dialogs) from the microplanning result which is then sent to a speech-synthesis component. It proceeds in three major steps which are also depicted in Fig. 5.

- A *tree selection* phase determines the set of relevant TAG trees. A first tree retrieval step maps every object of the dependency tree into a set of applicable elementary TAG trees. The main tree selection phase uses information from the microplanner output to further refine the set of retrieved trees.
- A *combination* phase finds a successful combination of trees to build a (derived) phrase structure tree.
- An *inflection* phase uses the information in the feature structures of the leaves (i.e. the words) to apply appropriate morphological functions, including the use of *irregular forms* as provided by the HPSG lexicon and *regular inflection* function as supplied (as LISP code) by the HPSG grammar.

An initial preprocessing phase computes the necessary auxiliary verbs from the tense, aspect, and sentence mood information. It also rearranges the dependency tree accordingly (e.g. subject arguments are moved from the main verb to become dependents of the inflected auxiliary verb).
The two core phases are the tree selection and the tree combination phase. The tree selection phase consists of two steps. First, a set of possible trees is retrieved and then appropriate trees are selected from this set. The retrieval is driven by the HPSG instance or word class that is supplied by the microplanner. It is mapped to a lexical type by a lexicon that is automatically compiled from the HPSG grammar. The lexical types are then mapped to a tree family, i.e., a set of elementary TAG trees representing all possible minimally complete phrase structures that can be build from the instance. The additional information in the dependency tree is then used to add further feature

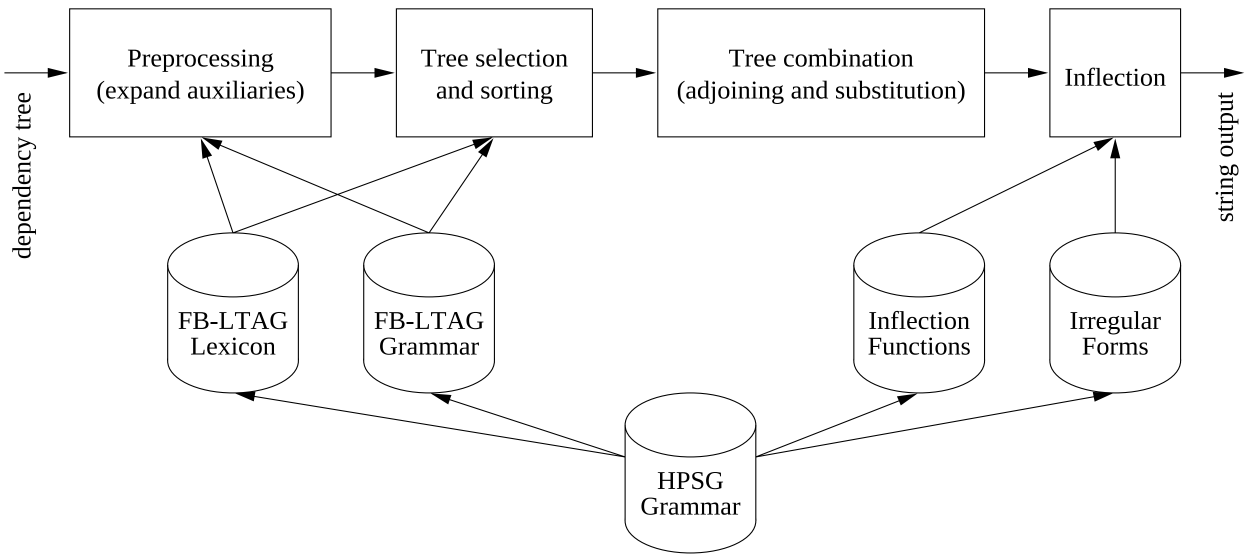


Figure 5: Steps of the syntactic generator.

values to the trees. This additional information acts as a filter for selecting appropriate trees in two stages:

- Some values are incompatible with values already present in the trees. These trees can therefore be filtered immediately from the set. E.g., a syntactic structure for an imperative clause is marked as such by a feature and can be discarded if a declarative sentence is to be generated.
- Additional features can prevent the combination with other trees during the combination phase. This is the case, for example with agreement features.

The combination phase explores the search space of all possible combinations of trees from the candidate sets for each lexical item (instance). An inefficient combination phase is a potential drawback of using the precomputed TAG trees. However, there is sufficient information available from the microplanner result and from the trees such that a well-guided best-first search strategy can be employed in the current system. The difference in run-time can be as dramatic as 24 seconds (comprehensive breadth-first) versus 1.5 seconds (best-first).
As part of the tree selection phase, based on the rich annotation of the input structure, the tree sets are sorted locally. Then a backtracking algorithm traverses the dependency tree in a bottom-up fashion[5]. At each node, and for each subtree in the dependency tree, a candidate for the phrase structures of the subtree is constructed. Then all possible adjunction or substitution sites are computed, possibly sorted (e.g. allowing for preferences in word order) and the best candidate for a combined phrase structure is returned. Since the combination of two partial phrase structures by adjunction or substitution might fail due to incompatible feature structures, a backtracking

[5]The algorithm stores intermediate results with a memoization technique.

algorithm must be used. A partial phrase structure for a subtree of the dependency is finally checked for completeness. These tests include the unifiability of all top and bottom feature structures and the satisfaction of all other constraints (e.g. obligatory adjunctions or open substitution nodes) since no further adjunctions or substitutions will occur in this subtree.
The necessity of a spoken dialog translation system to produce output robustly calls for some relaxations in these tests. E.g., 'obligatory' arguments may be missing in the utterance and the tests in the syntactic generator must accept a sentence with a missing obligatory object if no other complete phrase can be generated.
Figure 6 shows an example of the input of from the microplanner after the preprocessing phase has inserted the entity `LGV1` for the auxiliary *will*.

```
((ENTITY LGV1
  ((CAT V) (HEAD WILL_AUX_POS) (INTENTION WH-QUESTION) (FUNC AUX)
   (TENSE FUTURE) (MOOD INDICATIVE) (VOICE ACTIVE) (FORM ORDINARY)
   (VFORM FIN)))
 (ENTITY L5-WORK_ACCEPTABLE
  ((FORM ORDINARY) (VFORM BSE) (CAT V) (GOVERNED-BY WH-SENTENCE)
   (OPTIONAL-AGENT NO) (HEAD (OR SUIT_V1 SUIT_V2)) (REALIZED LOCAL)
   (REG LGV1)))
 (ENTITY L13-PRON
  ((REALIZED LOCAL) (CAT PPRON) (PERS 3) (NUM SG) (GENDER NTR)
   (TYPE NORMAL) (GOVERNED-BY V) (IS-COMPLEMENT T) (FORM CONTINUOUS)
   (REG LGV1) (FUNC AGENT)))
 (ENTITY L10-PRON
  ((REALIZED LOCAL) (CAT PPRON) (PERS 2A) (NUM SG) (GENDER FEM) (TYPE NORMAL)
   (GOVERNED-BY (OR V PREP SENTENCE)) (FORM CONTINUOUS) (REG L5-WORK_ACCEPTABLE)
   (FUNC PATIENT)))
 (ENTITY L6-TEMP_LOC
  ((CAT ADV) (REAL WH_QUEST) (SORT TIME) (POINTED-BY TEMP_LOC)
   (GOVERNED-BY (OR V N ADV SENTENCE)) (PRED TIME) (HEAD WHEN1)
   (REALIZED LOCAL) (WH-FOCUS T) (REG L5-WORK_ACCEPTABLE) (FUNC TEMP-SPEC)))
 (ENTITY L15-TEMP_LOC
  ((CAT ADV) (HEAD THEN_ADV) (REALIZED GROUP-TIME-DEMONSTRATIVE)
   (REAL (OR ADV WH_QUEST YOFC)) (SORT (SUBSORT TIME)) (POINTED-BY TEMP_LOC)
   (GOVERNED-BY (OR V N ADV SENTENCE)) (REG L5-WORK_ACCEPTABLE) (FUNC TEMP-SPEC))))
```

Figure 6: Example of the input from microplanning after preprocessing for auxiliaries

In the tree retrieval phase for `L5-WORK_ACCEPTABLE`, first the `HEAD` information is used to determine the lexical types of the possible realizations `SUIT_V1` and `SUIT_V2`, namely `MV_NP_TRANS_LE` and `MV_EXPL_PREP_TRANS_LE` respectively. These types are then mapped to their respective sets of elementary trees, a total of 25 trees. In the tree selection phase (as described above), this number is reduced to six. For example, the tree MV_NP_TRANS_LE.2 in Figure 3 has a feature CL-MODE with the value IMPERATIVE. Now, the microplanner output for the root entity `LGV1` contains the information `(INTENTION WH-QUESTION)`. The `INTENTION` information is unified with all appropriate CL-MODE features, which in this case fails. Therefore the tree MV_NP_TRANS_LE.2 can be discarded in the tree selection phase.

The combination phase uses the best-first bottom-up algorithm described above to determine one suitable tree for every entity and also a target node in the tree that is selected for the governing entity. For the above example, the selected trees and their combination nodes are shown in Figure 7[6].

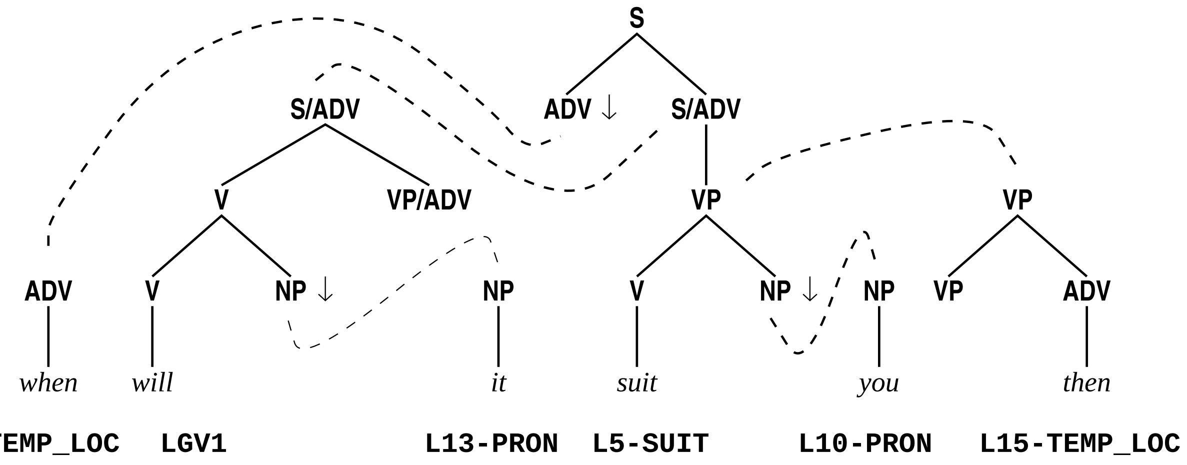


Figure 7: The trees finally selected for the entities of the example sentence. The dashed lines connect to suitable substitution or adjunction nodes. They correspond to the dependency tree.

The inflection function finally uses attribute values like verb-form, number and person from the final tree to derive the correct inflections. Information about the sentence mode WH-QUESTION can be used to annotate the resulting string for the speech-synthesis module.

6 Conclusion and Comparison

We have shown how preprocessing an HPSG grammar can be used to avoid the costly on–line application (unification) of HPSG schemata in a modularized generation system with a microplanner and a separate syntactic generator. The compilation of an HPSG grammar to TAG grammar allows the use of an efficient syntactic generator without sacrificing the declarative nature of the HPSG grammar.

It is important to compare the generation strategy presented here with semantic-head-driven generation [Shieber et al. 1990, van Noord 1990] which is a direct generation algorithm from logical form encodings. It improves previous algorithms in efficiency and in imposing less restrictions on the type of grammar. It is also applicable to HPSG and proceeds by applying the HPSG schemata in a bottom-up fashion, driven from the lexical heads of the schemata.

To a large extend, the TAG-based generation algorithm presented here goes through the same steps as semantic-head-driven generation. However, most of these steps will have been made during the off-line preprocessing and are encoded in the elementary trees of the TAG grammar thus resulting

[6]Note that the node labels shown in Figures 7 are only a concession to readability. The TAG requirement that in an auxiliary tree the foot node must have the same category label as the root node is formally fulfilled in our implementation.

in an important gain in efficiency. Note though, that the generation task in the algorithm presented here is shared between the microplanner and the syntactic generator, so a formal comparison must include both components.

Work on generation with TAG generally assumes that there is a one–to–one mapping between the information in the generator input and the choice of elementary tree [Mcdonald and Pustejovsky 1985, Yang, McCoy, and Vijay-Shanker 1991, Doran and Stone 1997]. In general, this will not be the case. In particular, in our system the input is not always sufficiently analyzed and the preprocessing from an HPSG grammar potentially creates more than one elementary tree that fits the input parameters.
One possible approach are choice nets–see [Yang, McCoy, and Vijay-Shanker 1991] who interpret systemic grammar in this way. Our approach has some similarity, though we have provided a more general algorithm that does not require the specification of grammar specific choice nets but rather executes tree selection and combination from more declarative knowledge bases. Tree selection is implemented mainly by unification (adding feature values from the input specification to the trees where unifiable) and the best-first search algorithm is a general framework for handling sets of possible elementary trees, including backtracking steps when non-local tests (e.g. unification in the resulting derived tree) fail. This approach is also a precondition in our system since we have no direct access to the TAG grammar as it is automatically preprocessed from an HPSG grammar.

VM–GECO is fully implemented (in Common Lisp) and integrated into the speech–to–speech translation system Verbmobil for English and German. For example, the underlying English HPSG grammar has almost 3000 lexical entries with over 200 lexical types. The resulting lexicalized TAG consists of about 2800 trees. The average overall generation time per sentence (up to length 24) is 0.7 cpu seconds on a SUN ULTRA-1 machine, 68% of the runtime are used for the microplanning while the remaining 32% of the runtime are used for syntactic generation.

7 Current Work

In general, the task of finding appropriate elementary trees for the chosen words and consequently a consistent phrase structure tree can exhibit constraints between any two elementary trees in the utterance (as expressed through feature equations). However, most of these constraints exist between elementary trees that are combined directly with each other (adjoined or substituted). To exploit this, we are currently experimenting with various well–established binary constraint–solving algorithms to preselect elementary trees that are pairwise consistent w.r.t. feature equations.

References

[Becker et al. 1996] Becker, T. W. Finkler, A. Kilger, and W. Wahlster. 1996. Vorhabensbeschreibung zur Sprachgenerierung innerhalb des Teilprojektes 5 (Sprachgenerierung und –synthese) in Verbmobil, Phase 2. Document, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, August.

[Becker et al. 1998] Becker, Tilman, Wolfgang Finkler, Anne Kilger, and Peter Poller. 1998. An efficient kernel for multilingual generation in speech–to–speech dialogue translation. In *Proceedings of COLING-ACL 98*, Montreal, Canada.

[Bos et al. 1996] Bos, J., B. Gambäck, C. Lieske, Y. Mori, M. Pinkal, and K. Worm. 1996. Compositional semantics in verbmobil. Technical report, University of the Saarland, Computational Linguistics, Saarbrücken, July. Verbmobil Report 135.

[Bub, Wahlster, and Waibel 1997] Bub, Th. W. Wahlster, and A. Waibel. 1997. Verbmobil: The combination of deep and shallow processing for spontaneous speech translation. In *Proceedings of ICASSP '97*. (forthcoming).

[Copestake, Flickinger, and Sag 1997] Copestake, Ann, Dan Flickinger, and Ivan A. Sag. 1997. Minimal recursion semantics: An introduction. available at `ftp://csli-ftp.stanford.edu/linguistics/sag/-mrs.ps.gz`.

[Doran and Stone 1997] Doran, Christy and Matthew Stone. 1997. Sentence planning as description using tree adjoining grammar. In *ACL-EACL*, Madrid, Spain, July.

[Dorna and Emele 1996] Dorna, M. and M. Emele. 1996. Semantic–based transfer. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*.

[Flickinger 1987] Flickinger, Daniel P. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University.

[Joshi 1987] Joshi, Aravind K. 1987. An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

[Kasper et al. 1995] Kasper, R., B. Kiefer, K. Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 92–99, Cambridge, Mass.

[Krieger and Schäfer 1994] Krieger, Hans-Ulrich and Ulrich Schäfer. 1994. $\mathcal{TDL}$—a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 893–899.

[Mcdonald and Pustejovsky 1985] Mcdonald, David D. and James D. Pustejovsky. 1985. Tags as a grammatical formalism for generation. In *Proc. of the 23 th ACL*, Chicago, IL.

[Pollard and Sag 1994] Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago.

[Schabes, Abeillé, and Joshi 1988] Schabes, Y., A. Abeillé, and A.K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. 12th International Conference on Computational Linguistics (COLING-88)*, pages 578–583, Budapest, August.

[Shieber et al. 1990] Shieber, Stuart, Gertjan Van Noord, Fernando Pereira, and Robert Moore. 1990. Semantic-head-driven generation. *Computational Linguistics Vol. 16 No. 1*, 16(1):30–43.

[van Noord 1990] van Noord, Gertjan. 1990. An overview of head-driven bottom-up generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*. Academic Press, New York, pages 141–165.

[Vijay-Shanker and Joshi 1991] Vijay-Shanker, K. and Aravind K. Joshi. 1991. Unification Based Tree Adjoining Grammars. In J. Wedekind, editor, *Unification-based Grammars*. MIT Press, Cambridge, Massachusetts.

[Wahlster 1993] Wahlster, W. 1993. Verbmobil: Translation of face–to–face dialoges. In *MT Summit IV*, Kobe, Japan.

[Yang, McCoy, and Vijay-Shanker 1991] Yang, Gijoo, Kathleen F. McCoy, and K. Vijay-Shanker. 1991. From functional specification to syntactic structures: Systemic grammar and tree adjoining grammar. *Computational Intelligence*, 7(4):207–219, November.