Gefördert vom

bmb+f

Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil

Verbundvorhaben

Structural Translation with Synchronous Tree Adjoining Grammars in VERBMOBIL

Karin Harbusch

Universität Koblenz-Landau

Peter Poller

DFKI GmbH

Vm

Report 184
Dezember 1996

Dezember 1996

Karin Harbusch
Universität Koblenz-Landau, Abt. Koblenz
Fachbereich Informatik, Institut für Computerlinguistik
Rheinau 1
56075 Koblenz

Tel.: (0261) 9119 - 463
Fax: (0261) 9119 - 465
e-mail: `harbusch@informatik.uni-koblenz.de`

Peter Poller
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken

Tel.: (0681) 302-5255
Fax: (0681) 302-5341
e-mail: `poller@dfki.uni-sb.de`

Gehört zum Antragsabschnitt: 9 Spontansprachliche und Inkrementelle Generierung

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie BMBF unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Structural Translation with Synchronous Tree Adjoining Grammars in VERBMOBIL

Karin Harbusch[‡] & Peter Poller[†]

[†] DFKI — German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D — 66123 Saarbrücken, Germany
poller@dfki.uni-sb.de

[‡] University of Koblenz — Computer Science Department,
Institute of Computional Linguistics
Rheinau 1, D — 56075 Koblenz, Germany
harbusch@informatik.uni-koblenz.de

Table of Contents

Summary

The *VERBMOBIL project* is developing a translation system that can assist a face–to–face dialogue between two non–native english speakers. Instead of having continiously speak english, the dialogue partners have the option to switch to their respective mother tongues (currently german or japanese) in cases where they can't find the required word, phrase or sentence. In such situations, the users activate *VERBMOBIL* to translate their utterances into english.

A very important requirement for such a system is *realtime processing*. Realtime processing is essentially necessary, if such a system is to be smoothly integrated into an ongoing communication. This can be achieved by the use of *anytime* processing, which always provides a result. The quality of the result however, depends on the computation time given to the system. Early interruptions can only produce shallow results. Aiming at such a processing mode, methods for fast but preliminary translation must be integrated into the system assisted by others that refine these results. In this case we suggest *structural translation* with *Synchronous Tree Adjoining Grammars (S–TAGs)*, which can serve as a fast and shallow realisation of all steps necessary during translation, i.e. analysis, transfer and generation, in a system capable of running anytime methods. This mode is especially adequate for standardized speech acts and simple sentences. Furthermore, it provides a result for early interruptions of the translation process. By building an explicit linguistic structure, methods for refining the result can rearrange the structure in order to increase the quality of the translation given extended execution time.

This paper describes the formalism of S–TAGs and the parsing algorithm implemented in VERBMOBIL. Furthermore the language covered by the german grammar is described. Finally we list examples together with the execution time required for their processing.

1 Motivation

As a prime target of the *VERBMOBIL project* (see, e.g., [Wahlster 93]), a machine translation system is being developed to assist translation in the following situation. Two non–native english speakers one japanese the other german, are engaged in an englisch dialogue assisted by the *VERBMOBIL* system on demand. Both partners can also use their mother tongues. In this case *VERBMOBIL* translates their utterances into english.

The natural integration of such a system into an ongoing face–to–face communication requires realtime processing. In *VERBMOBIL*, realtime processing is realized by *anytime constructions* (see, e.g., [Dean & Boddy 88]). Under the presupposition of always providing a translation result in realtime, processes can be interrupted at any time cutting off time–intensive tasks. In such a situation more or less preliminary results (e.g., by estimating the most probable result or analyzing in a rudimentary mode) are preferred instead of a complete syntactic and/or semantic/pragmatic transfer[1] result whose computation could impose an unnatural delay on the ongoing communication.

Beside the fact that all components of an anytime system have to be able to produce shallow output on demand, they must also have the ability to deal with rudimentary information as input. In order to manage an efficient communication for the far range of differences in the input, the individual components can be combined with each other depending on the level of how rudimentary the processing was. In this way different processing levels become possible according to the expected processing time. In the field of language translation, we propose the addition of a component for immediate structural translation in addition to a complete syntactic and semantic/pragmatic transfer component based on the above mentioned idea of providing different processing levels for the same task. In addition to the above mentioned *anytime processing* with which time intensive tasks can be cut off in favour of more or less preliminary translation results, directly relating syntactic structures for conventionalized speech acts or idioms is often more efficient than the algorithmic transfer of their complete syntactic and semantic/pragmatic representations into each other.

The component presented here becomes active in the production of shallow translations with respect to "flat" transfer and early anytime interruptions. In both cases, the input is rudimentarily analyzed in the following two situations:

- the syntactic component has not yet been activated. The terminal words,

[1] *Transfer* is a mapping function from an analyzed syntactic or semantic/pragmatic source structure onto an according target structure (for more information on the differences between transfer and interlingua see, e.g., [Nirenburg 93]).

i.e., the best hypothesis of the speech analyzer, are associated with their syntactic category but they remain unrelated to each other,

- a basic hypothesis of the content of the dialogue contribution is given by the *VERBMOBIL*-dialogue component [Alexandersson et.al. 1995] (e.g., salutation or reject–date) applying statistical methods[2].

Since the quality of the result of an anytime component monotonously increases with time, basic linguistic structures must be created in such a way that the components can be rearranged if the systems gets more execution time. Therefore the synchronization of two adequate linguistic formalisms seems to be an appropriate representation for that. We propose *Synchronous Tree Adjoining Grammars (S–TAGs)* that have been introduced for the immediate structural translation in [Schabes & Shieber 90]. The synchronization of two TAGs is realized in the following way. The elementary trees in the so called source and target grammar are specified as pairs in order to express structural correspondencies. In addition, such tree pairs can contain links between nonterminal nodes. These links are used to restrict the recursion operation in such a way that it has to take place in linked nodes in parallel only. Finally, the two inserted structures must belong to the same rule pair.

According to these constraints S–TAGs can be used for structural translation as follows. First, the input is analyzed with respect to the source–grammar which produces a source–derivation D_L. In a second step the synchronized derivation D_R can be constructed by identifying the synchronized derivation step of each derivation step in D_L with respect to the condition that synchronous derivation steps must be sanctioned by an appropriate link. Finally the translation result is the terminal string of D_R.

Unfortunately, considering the original definition of S–TAGs one can show that it allows the construction of a non–Tree–Adjoining language in one component although this component itself is specified by a TAG. This makes the parsing problem for S–TAGs harder than TAG parsing meaning that the above mentioned translation algorithm cannot work. As a result, a restricted definition for S–TAGs that requires a valid synchronous derivation to consist of isomorphic derivation trees in both components (in the following, we refer to this formalism by *IS-TAG* for *Isomorphic S–TAG*) was proposed in [Shieber 94]. In this case, only tree–adjoining languages can be formed in each component.

The next section gives the formal definition of IS–TAGs. After briefly describing Peter Poller's TAG–parser [Poller 94], we extend it to produce the target

[2]Actually, the second kind of information can be used to skip the indepth analysis of the dialogue contribution, if a conventionalized speech act is expected to be uttered in this situation.

derivation trees in parallel in an efficient manner. Section 4 outlines the application of structural transfer in the VERBMOBIL system using IS–TAGs. Experience gained especially from a prototype implementation is discussed. Finally future work is addressed. The appendices describe the current grammar G for the domain of appointment scheduling, the output language of G and a collection of translation examples along with their execution time on a MAC IVORY LISP–Machine.

2 Isomorphic Synchronous TAGs

As outlined in [Shieber 94], the original definition of S–TAGs does not restrict the structures that can be produced in the source and target languages. As a result, this allows for non tree–adjoining languages in one component, despite the fact that this component is specified as a TAG. Shieber therefore proposes a restricted definition of synchronous derivation, namely the condition that a valid synchronous derivation must consist of two isomorphic TAG–derivations. Since this definition restricts the power in both component grammars to TAG languages, the so called *Isomorphic Synchronous TAGs (IS–TAGs)* allow for a TAG–parsing strategy for the two individual grammars. Therefore IS–TAGs can be applied in our translation module.

A formal definition of IS–TAGs follows below:

Definition 1
A **synchronous tree–adjoinng grammar** *G (*S–TAG*) is a set of triples* $\{\langle L_i, R_i, \frown_i\rangle\}$. L_i *and* R_i *are elementary trees, both either initial or auxiliary.* $G_L := \{L_i\}$ *is called the* **source grammar** *and* $G_R := \{R_i\}$ *the* **target grammar**[3]. $\frown_i$ *is the linking relation between tree addresses (i.e. nodes) in* L_i *and* R_i.

In [Schabes & Shieber 90], a synchronous derivation is defined as a synchronous rewriting process. This rewriting process is responsible for the mentioned additional generative power in one component. The following definition of synchronous derivation as isomorphic TAG–derivations uses the standard notations of TAG–derivation trees by [Vijay–Shanker 87]. Here is a short recapitulation of the notation: a derivation tree consists of a set of nodes η. Each arc from η to $\mathsf{parent}(\eta)$ is labeled with $\mathsf{addr}(\mathsf{parent}(\eta))$ which means that $\mathsf{tree}(\eta)$ is an auxiliary

[3]Both TAGs allow for the definition of initial and auxiliary trees with an empty leaf string as long as such a tree is paired with a tree that contains at least one terminal in its leaf string. A tree which possesses an empty leaf string is called *non productive* oder *empty tree*. So, for reasons of implementability, a prerequisite of our parser described in the next section is that the source grammar does not contain empty trees. This is explained there in more detail

(or substitution–) tree that has been adjoined (or substituted) in tree(parent(η)) at node address addr(parent(η)).

Definition 2
A **isomorphic synchronous derivation** *of a S–TAG $G = \{\langle G_L, G_R\rangle, \frown_i\}$ is a pair $\langle D_L, D_R\rangle$ for which the following conditions hold:*

1. *D_L and D_R are well-formed derivation trees with respect to G_L and G_R, respectively.*
2. *D_L and D_R are isomophic, i.e. there is a one–to–one mapping* f *from the nodes of D_L to the nodes of D_R that preserves dominance, i.e. if* $\mathsf{f}(\eta_l) = \eta_r$ *then* $\mathsf{f}(\mathsf{parent}(\eta_l)) = \mathsf{parent}(\eta_r)$
3. *The isomorphic operations are sanctioned by links in tree pairs, i.e. if* $\mathsf{f}(\eta_l) = \eta_r$*, then there is a pair* $\langle \mathsf{tree}(\eta_l), \mathsf{tree}(\eta_r), \frown'\rangle$ *in G. Furthermore, if η_l has a parent, then there is a tree pair* $\langle \mathsf{tree}(\mathsf{parent}(\eta_l)), \mathsf{tree}(\mathsf{parent}(\eta_r)), \frown\rangle$ *in G and* $\mathsf{addr}(\mathsf{parent}(\eta_l)) \frown \mathsf{addr}(\mathsf{parent}(\eta_r))$.

As compared to original S–TAGs, constraint propagation is impossible in Isomorphic S–TAGs because a constraint now always applies to the node it is attached to. Therefore, one individual component in IS–TAGs can only form tree-adjoining languages. The proof for that is a reduction to tree–set–local MCTAGs and it is given in [Shieber 94].

3 Efficient Parsing of IS–TAGs

The restriction of synchronous derivations to isomorphic TAG–derivations ensures the implementability of a translation module that bases on a TAG–parser. Additional effort is only necessary for the detection of valid synchronous adjoinings. This section shows how the TAG–parser designed by Peter Poller ([Poller 94]) has been extended to detect isomorphic synchronous derivation steps.

Peter Poller's TAG–parser has a time complexity of $O(n^6)$ in the worst case and works in a two–level mode in a Earley–style manner. This means that in a first step, a context–free analysis for the source grammar is carried out using an extended version the Earley algorithm [Earley 70]. In the second step all context–free derivations which do not correspond to a valid TAG-solution are ruled out on the resulting item lists of the first step. This is done by an iterative identification and elimination process of innermost (so called complete) elementary trees[4]. Before we show how this second parsing step is extended to the identification of valid synchronous adjoinings, let us look at a short introduction of the parser itself.

[4]Complete elementary trees are elementary trees in the context–free parsing result in which

3.1 The Parsing Algorithm for TAGs

As mentioned above the parser works in a two–level mode. The first parsing step is a context–free analysis of the input sentence using an extended version of the Earley algorithm with respect to a contextfree interpretation of a TAG, the so–called *context–free kernel* of a TAG. The second parsing step is the iterative elimination of complete trees on the context–free parsing result by which invalid TAG solutions are ruled out.

The context–free kernel of a TAG is defined as the context–free grammar that results from the interpretation of each mother–daughter relation in all elementary trees as a context–free rule. The computation of the context–free kernel of the underlying TAG grammar can be done in a preliminary step before parsing. In order to be able to identify the elementary tree from which a rule of the context–free kernel had been extracted all elementary trees and their nodes get unique numbers which are attached to them and thereby into the rules of the context–free kernel.

3.1.1 Context–free Analysis

The context–free analysis uses an extended version of the Earley algorithm. However, the second parsing step which involves the identification of valid adjoinings in the context free parsiong result, makes extensions necessary. In order to be able to identify elementary trees inside the context–free parsing result, additional effort is necessary for the administration of the mentioned unique node numbers during the Earley–analysis. Furthermore, additional pointers are used between the individual items produced by the Earley algorithm. This ensures an explicit encoding of all derived trees of the context–free kernel inside the context–free parsing result. When using the original Earley algorithm the derived trees are only implicitely given.

3.1.2 Iterative Elimination of Complete Trees

The iterative elimination must be initialized by the first set of so called innermost trees, i.e., auxiliary trees in which no adjoining has taken place. At least one such tree must exist in each iterative step. After the correct elimination of all complete elementary trees, this condition holds again. Therefore the elimination procedure is iterated until nothing but initial trees remain.

no adjunction has taken place. The elimination of a complete elementary tree means the removal of the tree which makes its adjunction undone. This procedure can be iterated and at least one complete tree must exist in each iteration step [Harbusch 89].

This initialization procedure is organized as a recursive top–down traversal of all context–free derivations. Context–free derivation steps that belong to the same elementary tree can be identified by using the unique node numbers.

After the initialization of the first set of complete trees in the context–free parsing result, their iterative elimination is initiated in order to make all adjoinings undone until only initial trees remain. If this step succeeds all correct TAG–derivations are identified.

Obviously the elimination of elementary trees is done only virtually, otherwise we would risk loosing the completeness of the parsing result. So an elimination of a complete tree is organized as the definition of an immediate neighbourhood between the root and foot node of an auxiliary tree. In this way complete elementary trees that have been eliminated by this method are simply skipped inside the consequent iteration steps.

Finally an input sentence is accepted, if and only if, nothing but initial trees remain after the iteration.

3.2 The Translation Algorithm

This section shows how this parsing algorithm can be extended to detect synchronous adjoining and produce all synchronously derived trees. Since the TAG–parser explicitely identifies the individual derivation steps in the source grammar it is possible to extend the identification and elimination of a complete elementary tree in the source grammar by checking the synchronicity condition. The adjunction node must have a link and the partner tree of the adjoined tree in the source grammar must be adjoinable in the node the link impinges on. If the synchronicity condition does not hold, the complete tree in the source grammar is ruled out. So, doing elimination and synchronicity check at the same time has the advantage that invalid synchronous derivation steps can be ruled out immediately.

If the synchronicity condition holds, the step of eliminating a complete elementary tree in the source grammar is extended to produce a so called *building instruction* for the synchronized derivation step in the target grammar. The main advantage of building instructions in favour of producing all synchronous derivations simultaneously after each elimination in the source grammar is that intermediate target–structures can be shared instead of spelling them out. So the time complexity is not increased by this additional computation. The production of such building instructions is described in the following.

Suppose the adjunction of a tree β_1 in a tree α_1 at node x_1 has been identified in the source grammar. If the synchronicity condition of this adjunction holds, a building instruction for the synchronized adjoining of the following form is constructed:

$$adjoin\ (\alpha_2,\ x_2,\ \beta_2).$$

This instruction implies that the synchronous derivation step is the adjunction of the tree β_2 into the tree α_2 in node x_2 — where (α_1, α_2) and (β_1, β_2) are tree pairs and there is a link between the node x_1 in α_1 and x_2 in α_2.

This instruction is propagated inside the further iteration steps. So, once the next complete elementary tree in the source grammar is identified, its elimination is carried out in the same way. The propagated building instruction thereby serves as the synchronous tree of the currently found complete tree in the source grammar. In this way we get such embedded building instructions like:

$$adjoin\ (\gamma_2,\ y_2,\ adjoin\ (\alpha_2\ ,x_2,\ \beta_2)).$$

This means that during the iteration, the individual building instructions are combined step by step in a similar way as derivation trees for TAGs are specified in order to share subdescriptions in an efficient manner. For reasons of efficiency, all building instructions remain uninstantiated until the iteration has terminated successfully. At the end of the iteration only those complex building instructions that are attached to valid readings in the source grammar are evaluated.

A main advantage of using a two level analysis approach — where the second step follows the derivation definition in an inverse manner — for IS–TAGs is that the system always operates on *related* subresults according to the source and the target grammar.

Another feature of our system is that it allows switching between source and target languages. The system is therefore *reversible* as long as no ϵ–trees, i.e. elementary trees with an empty terminal string, occur in the selected source grammar.[5] The reason for this restriction is that empty trees in the source grammar could lead to an infinite number of derivations for the input sentence. This in turn leads to an infinite number of translation results thus making it unclear what the translation of the input sentence should be.

In general, ambiguous input sentences require further strategies to select the best alternative. As we describe in section 4, the solution computed first is taken as the translation result in our system. Of course, better strategies — e.g. methods concerning additional knowledge about the input sentence (e.g., a semantic

[5]Reversibility is an overall goal in VERBMOBIL. However, the analysis is realized by another formalism other than TAG. On the long-term, a synchronization between this formalism and TAG is planned so that the anytime mode can run interleaved with "free" analysis. As mentioned above with respect to generation, this is an important reason which determines the formalisms to be selected. Currently we have some basic ideas, on how this combination can be stated using a *mixed synchronization* of two different formalisms, although their expressivity should be just the same here.

representation) — should be preferred. Since we haven't implemented any knowledge based decision making, we will return to this task in the final section, where future work is mentioned.

4 Applying IS–TAGs in VERBMOBIL

In the VERBMOBIL translation scenario, Isomorphic S–TAGs are used as an additional transfer component in *anytime processing* in the following contexts. Synchronized structures in german/japanese and english allow for rudimentary translation results in case of an early anytime interruption of the system (e.g., quick and dirty translation of keywords). Consequently, a broad coverage of the source language must be represented in the synchronous framework.

On the other hand, direct translation of special constructions (e.g., conventionalized speech acts or idioms) by synchronized structures speeds up the system thus preserving more time for the processing of difficult phenomena in the source utterance. In this case, the final translation result is only partially processed in the synchronous framework so that the synchronous framework has to only cover a sublanguage.

In the second case however, an elaborate structural representation is crucial. It allows for the integration of phrases produced by the synchronous framework into, loosely speaking, "freely" generated phrases in order to form a correct translation result. This consideration rules out an approach on the basis of "flat" templates in favour of a "deep" grammatical representation.

Necessarily, anytime processing has effects on the overall architecture. The results can be preliminary in two different respects:

- The result is *underspecified* (e.g., an NP without determined syntactic function). The effect on the architecture of the overall system is that each component must be flexible enough to additionally deal with underspecified input. This flexibility can be realized by providing several sub–components for the same task depending on the type of underspecification (e.g., a pure semantic NP–analyzer if the syntactic function is unknown), or it results from one component which works with incomplete knowledge — similar to an *incremental* system [Kilger & Finkler 95].

- The result is *guessed*, e.g., by estimating defaults, statistics etc. Since the result can be wrong, the next components must be able to identify contradictions, which require backtracking in the input–providing components and thus revising their own computation as soon as revised input specifications arrive.

This paper focusses on the first alternative. We describe a component that can translate with rudimentary knowledge in specific situations. In the following situations, there is only rudimentary knowledge to start with:

- the syntactic component has not yet been activated. So terminals, i.e., the best chain of the speech analyzer, are associated with their syntactic category but remain unrelated to each other.
- A basic hypothesis of the content of the dialogue contribution is given by the VERBMOBIL–dialogue component (e.g., salutation or accept–date) using statistac methods [Alexandersson et.al. 1995].

A main advantage of such a behaviour instead of detailed processing is that a result is always provided (interrupts). Furthermore, simple constructions do not waste time (realtime processing).

Currently, we have an implementation of a flat translation system from german into english in the formalism of IS–TAGs. As mentioned before, flat translation consists of the following steps. First the input sentence is analyzed on the source grammar with respect to synchronicity constraints and, if successful, its synchronous derivations are generated in the target grammar. The terminal string of the solution computed first is taken as the translation result (best–first strategy). This means that we have not yet implemented an elaborate strategy to choose the best alternative for ambiguous sentences (see also the final section).

Figure 1 exemplifies flat transfer rules for *((wunder-)* schönen) Tag (Herr Miller)* and *Hello (Mister Miller) How are you*, respectively. The example illustrates the necessity of empty trees. In english only the conventionalized greating "hello, how are you" is possible. In german however, an infinite number of "wunder-" may modify "schoen" or "Tag". Addressing the dialogue partner by name is facultatively possible in both languages.

Our system covers all typical dialogue contributions in the VERBMOBIL scenario in order to allow for early interrupts. The grammar consists of approximately 500 trees separated into 10 different dialogue–act classes. This allows the system to switch between individual subgrammars in order to restrict the number of grammar rules to be regarded in a structural transfer step (for a compact description of the individual dialog–acts distinguished in our system see Appendix A together with the examples in Appendix B). Peter Poller's parser, implemented in COMMON LISP on an MAC–IVORY machine, was initially extended to run Synchronous TAGs without multiple links (respective nodes are duplicated in the current grammar). In the average case, a structural transfer step takes 0.5 sec on a SYMBOLICS MAC–IVORY machine.

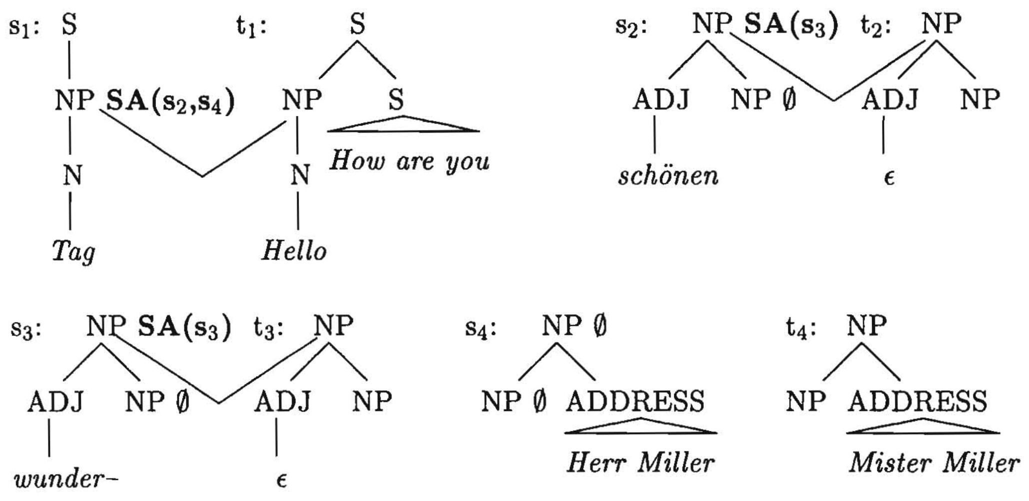

Figure 1: Example trees in a Synchronous TAG (with Constraints).

5 Future Work

In the last section, we mentioned the problem of ambiguous translation results and the unsatisfying best–first strategy to handle it. A more sophisticated strategy should allow for a better selection of the best alternative for ambiguities in the target language. The new strategy we are currently testing is to define probabilities for the individual competing structures.

As outlined in the motivation section, VERBMOBIL is designed to work in a system–wide anytime fashion. Up to now, we have realized a specific line of processing on the syntactic level. Obviously, for *semantic* and *pragmatic transfer*, the same algorithm can be applied. We have started to realize a transfer grammar containing information on the semantic/pragmatic level. Currently, we have no experience with the performance of such a grammar. This must be studied in future work. A further question concerning all levels of processing remains open: the intregation of those directly coupled lines of processing with the elaborate processing of other parts of the dialogue contribution. Basically, we favour a technique of sharing structures efficiently.

Finally, we address a more theoretical problem. As mentioned before, the general definition of S–TAGs is more powerful than that of TAGs. In order to find further formalisms beside IS–TAGs, that are equivalent to tree–set–local MC-TAGs, we are currently studying *Synchronous Context-free Grammar* (*S–CFG*).

Figure 2 depicts the example of an S–CFG producing the language a^n b^n c^n d^n in one component. Note that multiple links are propagated to the leftmost nonterminal daugther here without loss of generality.

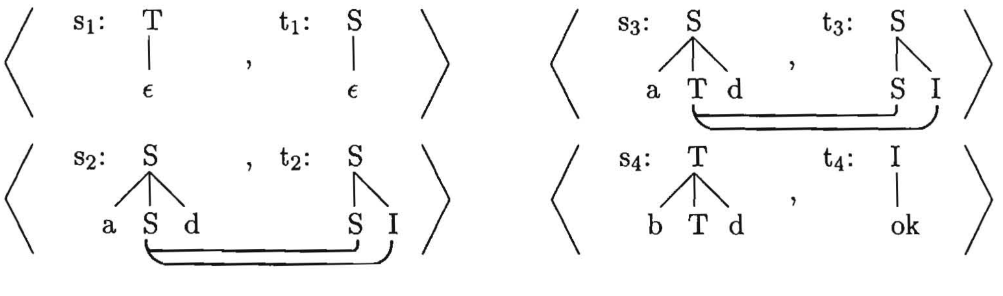

Figure 2: Two synchronized CFGs for a^n b^n c^n d^n

We assume that it is possible to show the equivalence of TAGs and S–CFGs. If so, S–CFGs can be an interesting formalism for the specification of natural–language grammars. Furthermore, we hope that a direct parser for S–CFGs will have a better time complexity — at least in the average case, because the control of the synchronous contextfree derivation in the target grammar can be managed more locally than for TAGs. These points are open questions to be addressed in the future.

As outlined in [Shieber 94] constrastive phonomena exist in different languages which cannot be expressed by isomorphic S–TAGs whenever they require the synchronization of non–isomorphic TAG–derivations. For example in the sentences

- Le docteur lui soigne les dents,
  The doctor treats his teeth,

the pronoun "lui" is a modification of the main verb "soigne" while "his" modifies "teeth". It's not possible to describe this correspondency with IS-TAGs because the two derivations of the sentences above are non–isomorphic as figure 3 shows.

In order to express such non–isomorphic correspondencies, we propose a relaxation on the synchronicity constraint. This allows for *dynamic links* that may be defined between trees even if they do not form a tree pair in the synchronous TAG. The dynamics then come in automatically during a synchronous derivation because such a link between non paired trees is only applicable if both trees are part of the current two derivations of the components, i.e. it requires adjunctions to have taken place before. The new formalism resulting from this extension is called *Non–isomorphic Synchronous TAG* (*NIS–TAG*, [Harbusch & Poller 96]).

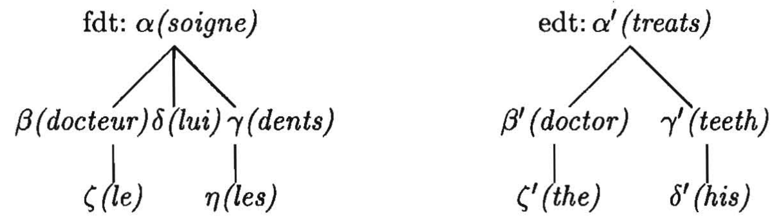

Figure 3: Non–isomorphic synchronous derivation trees

Similar to S–TAGs, in NIS–TAGs constraints may also have an indirect effect on synchronous adjoinings. The reason is that the satisfaction of a constraint in one component may influence derivation steps in the other component because of the synchronicity condition. NIS–TAGs also allow for the formation of a non tree–adjoining language in one component although this component is specified by a TAG. Furthermore, in NIS–TAGs, the two nodes at which an adjoining takes place should not necessarily belong to trees of a tree pair anymore. So the parsing problem for NIS–TAGs is similar to that of S–TAGs. Nevertheless, a parser for NIS–TAGs that bases on a TAG–parser for the source grammar is available and is described in [Harbusch & Poller 96]. The construction of the synchronized target derivation however, needs exponentional runtime because links can be applied dynamically. Metaphorically speaking, the construction of valid target derivations is a puzzle comprising individual target derivation steps, whose solution are all target derivation trees, in which all puzzle pieces are used.

Since we have no implementation of the new formalism, we are not in a position to compare it with our IS–TAG implementation. A comparison based on an average case is expected to help in making a decision, as to which grammar type should be further developed for use in the shallow translation module of the VERBMOBIL project.

References

[Alexandersson et.al. 1995] J. **Alexandersson**, E. **Maier**, und N. **Reithinger**. 1995. *A Robust and Efficient Three-Layered Dialog Component for a Speech-to-Speech Translation System.* In Procs. of the 7th EACL, Dublin, Ireland, pp. 188-193.

[Dean & Boddy 88] Th. **Dean**, M. **Boddy**. 1988. *An Analysis of Time-Dependent Planning.* In Procs. of the 7th AAAI, pp. 49–54.

[Earley 70] J. **Earley**. 1970. *An Efficent Context-free Parsing Algorithm.* Communications of the Association for Computing Machinery (ACM), 13:2, pp. 94–102.

[Harbusch 89] K. **Harbusch**. 1989. *Effiziente Strukturanalyse natürlicher Sprache mit Tree Adjoining Grammars.* Dissertation, Universität des Saarlandes, Saarbrücken.

[Harbusch & Poller 96] K. **Harbusch**, P. **Poller**. *Non-Isomorphic Synchronous TAGs.* to appear in CSLI Lecture Notes.

[Kilger & Finkler 95] A. **Kilger**, W. **Finkler**. 1995. *Incremental Generation for Real-Time Applications.* Technical Report RR–95–11, DFKI, Saarbrücken.

[Nirenburg 93] S. **Nirenburg** (ed). 1993. *Forum Issue: Current Research in Machine Translation.* Machine Translation 7:4, pp. 229–331.

[Poller 94] P. **Poller**. 1994. *Incremental Parsing with LD/TLP–TAGs.* Computational Intelligence 10:4, pp. 549–562.

[Schabes & Shieber 90] Y. **Schabes**, S.M. **Shieber**. 1990. *Synchronous Tree Adjoining Grammars.* Procs. of the 13th COLING, Helsinki, Finland, Volume 2, pp. 253–258.

[Shieber 94] S.M. **Shieber**. 1994. *Restricting the weak-generative capacity of Synchronous Tree-Adjoining Grammars.* Computational Intelligence, Volume 10, Number 4, pp. 371–385.

[Vijay–Shanker 87] K. **Vijay–Shanker**. 1987. *A Study of Tree Adjoining Grammars.* Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA.

[Wahlster 93] W. **Wahlster**. 1993. *VERBMOBIL: Translations of Face-to-Face Dialogs.* In Procs. of the MT Summit IV, , Kobe, Japan, pp. 127–135.

A The Language defined by the Source Grammar

In this section, the covered languages of the individual partitions of the german grammar are described in terms of regular expressions. The following abbreviations are used:

Notation	Meaning
$(x)^{(0\|1)}$	0 or 1 occurrences of x
$(x)^*$	0 up to n occurrences of x
(x|y)	either x or y
x.y	x and y in exactly this order
[x,y]	either x y or y x
$<x>$	is a template for a substructure
i,j	1. . .31

VM–INTRODUCTION (currently 32 tree–pairs (TP)):

((einen)$^{(0|1)}$. (((wunder-)* . schoenen)$^{(0|1)}$. guten)$^{(0|1)}$. Tag|Morgen|nAbend|Abend
| gruess . (Sie)$^{(0|1)}$. Gott)
. [(wuensche . ich . (Ihnen | Dir)$^{(0|1)}$)$^{(0|1)}$, (aus . Saarbruecken)$^{(0|1)}$,
(nach . Muenchen)$^{(0|1)}$, < *ANREDE* >$^{(0|1)}$]

< *ANREDE* > ::=
(((mein)$^{(0|1)}$. (lieber | sehr . geehrter))$^{(0|1)}$. (Hans | (Herr . (Doktor | Professor)$^{(0|1)}$
. Meier))
| ((meine)$^{(0|1)}$. (liebe | sehr . geehrte))$^{(0|1)}$. (Maria | (Frau . (Doktor | Professor)$^{(0|1)}$
. Kunze)))$^{(0|1)}$

VM–INTRODUCTION–REACTION (45 TP):

(ebenfalls | (auch . (Ihnen | Dir)$^{(0|1)}$))$^{(0|1)}$
. (einen)$^{(0|1)}$. (((wunder-)*schoenen)$^{(0|1)}$. guten)$^{(0|1)}$. Tag | Morgen | nAbend
. [(wuensche . ich . (Ihnen | Dir)$^{(0|1)}$)$^{(0|1)}$, ((zurueck)$^{(0|1)}$. nach . Saarbruecken)$^{(0|1)}$
, (aus . Muenchen)$^{(0|1)}$, < *ANREDE* >$^{(0|1)}$]

(ebenfalls | gruess . (Sie)$^{(0|1)}$. Gott) . [((zurueck)$^{(0|1)}$. nach . Saarbruecken)$^{(0|1)}$
, (aus . Muenchen)$^{(0|1)}$, < *ANREDE* >$^{(0|1)}$]
(PRON)

VM–TOPIC (26 TP):

[(Sie . wissen . schon)$^{(0|1)}$, (weshalb . ich . anrufe)$^{(0|1)}$]
. (((wir . (muessen | wollten) . (doch . (noch)$^{(0|1)}$)$^{(0|1)}$. (einen | diesen) . Termin
. $<REISE>^{(0|1)}$. (machen | festlegen)))
| (es . geht . um) . (einen | diesen) . Termin . $<REISE>^{(0|1)}$)
. $<NACHFRAGE-WANN>^{(0|1)}$

$<REISE>$ ::=
fuer . (die | unsere) . Reise . [(Sie . wissen . schon)$^{(0|1)}$
, ((in . die . Schweiz) . (zu . unsern . Geschaeftspartnern)$^{(0|1)}$)$^{(0|1)}$]

$<NACHFRAGE-WANN>$ ::=
(((wann . waere . es . (Ihnen | Dir)$^{(0|1)}$. (am . ehesten)$^{(0|1)}$. recht)
| (wann . waere . es . ((am . ehesten)$^{(0|1)}$. geschickt) | (am . geschicktesten)))
. bei . (Ihnen | Dir))

VM–TOPIC–REACTION (35 TP):

(oh)$^{(0|1)}$. (ja)$^{(0|1)}$. (prima|toll)$^{(0|1)}$
. (((dann)$^{(0|1)}$. (lassen . Sie | lass) . uns . doch
. (einen | einen . solchen | solch . einen | diesen) . Termin . ausmachen)
| ((dann)$^{(0|1)}$. (lassen . Sie | lass) . uns . doch . (mal)$^{(0|1)}$. schauen))
. [$<NACHFRAGE-WANN>^{(0|1)}$, ((haben . Sie) | (hast . Du)
. (einen | Deinen | Ihren | den) . Kalender . (gerade)$^{(0|1)}$. (vorliegen | da)$^{(0|1)}$)$^{(0|1)}$]

VM–PROPOSAL (65 TP):

$<INTRO>^{(0|1)}$. wie . waere . es . ((mit . dem | am) . i-ten . Januar | vom
. i-ten . bis . j-ten . Januar). $<EINSCHRAENK>^{(0|1)}$
. $<NACHFRAGE-GEHT>^{(0|1)}$

$<INTRO>^{(0|1)}$. ich . koennte . (Ihnen | Dir)$^{(0|1)}$. ((den . i-ten . Januar)
| (vom . i-ten . bis . j-ten . Januar)) . anbieten . $<EINSCHRAENK>^{(0|1)}$
. $<NACHFRAGE-GEHT>^{(0|1)}$

$<INTRO>^{(0|1)}$. am . (naechsten)$^{(0|1)}$. Dienstag . ((den . i-ten . Januar)
| (vom . i-ten . bis . j-ten . Januar))$^{(0|1)}$. haette . ich . einen . Termin . frei
. $<EINSCHRAENK>^{(0|1)}$. $<NACHFRAGE-GEHT>^{(0|1)}$

< *INTRO* > ::=
(oh)$^{(0|1)}$. ja | nein | aber | (mal . sehen) | (mal . schauen)

< *EINSCHRAENK* > ::=
aber . (nur | nicht) . (nachmittags | (am . Nachmittag))

< *NACHFRAGE – GEHT* > ::=
((geht . das . bei . (Ihnen | Dir)) | (waere . (Ihnen | Dir) . das
. (recht | geschickt)))$^{(0|1)}$

VM–PROPOSAL–REACTION (19 TP):

((oh)$^{(0|1)}$. (ja | ja . doch | nein))$^{(0|1)}$
. (da . kann . ich . (nicht | schlecht | prima)$^{(0|1)}$
| das . geht . (prinzipiell)$^{(0|1)}$. nicht
| (das | Dienstag)$^{(0|1)}$. ist . (prinzipiell)$^{(0|1)}$. (ganz)$^{(0|1)}$. (schlecht | prima))
. [(bei . mir | (fuer . mich))$^{(0|1)}$, (in . meinem . Terminkalender)$^{(0|1)}$, (mit . meinen
. Terminen)$^{(0|1)}$]

VM–AGREEMENT (8 TP):

(prima | gut | ok)$^{(0|1)}$. ((ich . trage . (es | (den . Termin)) . (bei . mir)$^{(0|1)}$. ein)
| (ich . trage . mir . den . Termin . ein)
| (halten . wir . den . Termin . fest))

VM–BYE (6 TP):

(bis . dann)$^{(0|1)}$. (in . der . Schweiz)$^{(0|1)}$. ((auf . Wiedersehen) | tschuess) . (dann)$^{(0|1)}$

VM–BYE–REACTION (6 TP):

(bis . dann)$^{(0|1)}$. (in . der . Schweiz)$^{(0|1)}$. ((auf . Wiedersehen) | tschuess) . (dann)$^{(0|1)}$

B Examples

In the following are examples of translations from german into english. They are organized in subsections that reflect the predicted dialogue act and its corresponding grammar partition used for the translation. The respective execution time on a MACIVORY Lisp–Machine is given in brackets.

B.1 VM-INTRODUCTION

Tag $\Longrightarrow$ hello how are you **(0.466152 sec)**

Morgen $\Longrightarrow$ good morning how are you **(0.496051 sec)**

Abend $\Longrightarrow$ good evening how are you **(0.433755 sec)**

gruess Gott $\Longrightarrow$ hello how are you **(0.502105 sec)**

gruess Sie Gott $\Longrightarrow$ hello how are you **(0.491554 sec)**

guten Tag $\Longrightarrow$ hello how are you **(0.543846 sec)**

guten Morgen $\Longrightarrow$ good morning how are you **(0.571269 sec)**

guten Abend $\Longrightarrow$ good evening how are you **(0.545040 sec)**

einen guten Tag $\Longrightarrow$ hello how are you **(0.594746 sec)**

einen guten Morgen $\Longrightarrow$ good morning how are you **(0.591639 sec)**

einen guten Abend $\Longrightarrow$ good evening how are you **(0.627422 sec)**

Tag wuensche ich $\Longrightarrow$ hello how are you **(0.712472 sec)**

Tag wuensche ich Ihnen $\Longrightarrow$ hello how are you **(0.985506 sec)**

Tag Hans $\Longrightarrow$ hello Hans how are you **(0.684729 sec)**

Tag mein lieber Herr Professor Meier $\Longrightarrow$ hello my dear professor Meier how are you **(1.015726 sec)**

Tag liebe Frau Kunze $\Longrightarrow$ hello dear Misses Kunze how are you **(0.932308 sec)**

Tag sehr geehrter Herr Meier $\Longrightarrow$ hello dear Mister Meier how are you **(1.275869 sec)**

Tag sehr geehrter Herr Doktor Meier $\Longrightarrow$ hello dear doctor Meier how are you **(1.216402 sec)**

schoenen guten Tag $\Longrightarrow$ hello how are you **(0.593036 sec)**

schoenen guten Morgen $\Longrightarrow$ good morning how are you **(0.608246 sec)**

schoenen guten Abend $\Longrightarrow$ good evening how are you **(0.609144 sec)**

einen schoenen guten Tag $\Longrightarrow$ hello how are you **(0.635377 sec)**

einen schoenen guten Morgen $\Longrightarrow$ good morning how are you **(0.636726 sec)**

einen schoenen guten Abend $\Longrightarrow$ good evening how are you **(0.633265 sec)**

wunder- schoenen guten Tag $\Longrightarrow$ hello how are you **(0.704340 sec)**

wunder- schoenen guten Morgen $\Longrightarrow$ good morning how are you **(0.731259 sec)**

wunder- schoenen guten Abend $\Longrightarrow$ good evening how are you **(0.692479 sec)**

einen wunder- schoenen guten Tag $\Longrightarrow$ hello how are you **(0.723414 sec)**

einen wunder- schoenen guten Morgen $\Longrightarrow$ good morning how are you **(0.757820 sec)**

einen wunder- wunder- schoenen guten Abend $\Longrightarrow$ good evening how are you **(0.888301 sec)**

einen wunder- schoenen guten Tag wuensche ich Dir $\Longrightarrow$ hello how are you **(1.236423 sec)**

einen wunder- schoenen guten Tag wuensche ich Ihnen $\Longrightarrow$ hello how are you

(1.218007 sec)

Tag wuensche ich Dir $\Longrightarrow$ hello how are you **(0.970725 sec)**

Tag wuensche ich Ihnen $\Longrightarrow$ hello how are you **(0.988851 sec)**

Tag wuensche ich aus Saarbruecken $\Longrightarrow$ hello from Saarbrucken how are you **(1.001368 sec)**

Tag aus Saarbruecken wuensche ich $\Longrightarrow$ hello from Saarbrucken how are you **(1.239584 sec)**

Tag lieber Hans $\Longrightarrow$ hello dear Hans how are you **(0.833119 sec)**

Tag sehr geehrter Hans $\Longrightarrow$ hello dear Hans how are you **(1.056448 sec)**

Tag sehr geehrter Herr Doktor Meier $\Longrightarrow$ hello dear doctor Meier how are you **(1.392789 sec)**

Tag meine liebe Frau Professor Kunze $\Longrightarrow$ hello my dear professor Kunze how are you **(1.042731 sec)**

B.2 VM-INTRODUCTION and VM-INTRODUCTION-REACTION

ebenfalls $\Longrightarrow$ thank you fine how are you **(0.545043 sec)**

ebenfalls einen wunder- schoenen guten Tag $\Longrightarrow$ fine how are you **(1.093603 sec)**

auch Ihnen einen wunder- schoenen guten Tag mein lieber Herr Professor Meier $\Longrightarrow$ fine my dear professor Meier how are you **(2.070439 sec)**

ebenfalls einen wunder- schoenen guten Tag zurueck nach Saarbruecken $\Longrightarrow$ fine in Saarbrucken how are you **(1.433055 sec)**

Tag zurueck nach Saarbruecken $\Longrightarrow$ fine in Saarbrucken how are you **(0.883770 sec)**

Tag nach Saarbruecken $\Longrightarrow$ fine in Saarbrucken how are you **(1.562337 sec)**

Tag wuensche ich Ihnen $\Longrightarrow$ fine how are you **(1.201394 sec)**

Tag wuensche ich nach Saarbruecken $\Longrightarrow$ fine in Saarbrucken how are you **(1.212710 sec)**

Tag nach Saarbruecken wuensche ich $\Longrightarrow$ fine in Saarbrucken how are you **(1.219446 sec)**

Tag wuensche ich Ihnen nach Saarbruecken $\Longrightarrow$ fine in Saarbrucken how are you **(1.574126 sec)**

B.3 VM-TOPIC

wir muessen einen Termin machen $\Longrightarrow$ we have to make a date **(0.784800 sec)**

wir muessen einen Termin machen wann waere es recht $\Longrightarrow$ we have to make a date when would it suit you best **(1.336077 sec)**

wir muessen einen Termin festlegen $\Longrightarrow$ we have to determine a date **(0.785351 sec)**

wir muessen doch einen Termin machen $\Longrightarrow$ we have to make a date **(0.972783 sec)**

wir muessen doch noch einen Termin machen $\Longrightarrow$ we have to make a date **(1.396464 sec)**

wir muessen doch noch diesen Termin machen $\Longrightarrow$ we have to make this date **(1.135776 sec)**

wir wollten doch noch einen Termin machen $\Longrightarrow$ we wanted to make a date **(1.107128 sec)**

wir wollten doch noch diesen Termin machen $\Longrightarrow$ we wanted to make this date **(1.259343 sec)** wir wollten doch noch diesen Termin fuer die Reise machen $\Longrightarrow$ we wanted to make this date for the journey **(1.475527 sec)**

wir wollten doch noch diesen Termin fuer unsere Reise machen $\Longrightarrow$ we wanted to make this date for our journey **(1.466356 sec)**

weshalb ich anrufe wir muessen doch noch einen Termin machen $\Longrightarrow$ the reason for my call is we have to make a date **(1.246564 sec)**

Sie wissen schon wir muessen doch noch einen Termin machen $\Longrightarrow$ you know we have to make a date **(1.531589 sec)**

wir muessen doch noch einen Termin Sie wissen schon machen $\Longrightarrow$ we have to make a date you know **(1.318011 sec)**

wir wollten einen Termin fuer die Reise in die Schweiz machen $\Longrightarrow$ we wanted to make a date for the journey to Switzerland **(1.899738 sec)**

wir wollten einen Termin fuer die Reise in die Schweiz zu unsern Geschaeftspartnern machen $\Longrightarrow$ we wanted to make a date for the journey to Switzerland to talk to our partners **(2.304599 sec)**

Sie wissen schon wir wollten einen Termin fuer die Reise in die Schweiz zu unsern Geschaeftspartnern machen $\Longrightarrow$ you know we wanted to make a date for the journey to Switzerland to talk to our partners **(3.473346 sec)**

wir wollten doch noch diesen Termin fuer die Reise festlegen $\Longrightarrow$ we wanted to determine this date for the journey **(1.511424 sec)**

wir wollten doch noch diesen Termin fuer unsere Reise machen $\Longrightarrow$ we wanted to make this date for our journey **(1.504470 sec)**

B.4 VM-TOPIC-REACTION

oh ja dann lassen Sie uns doch mal schauen $\Longrightarrow$ well then let us see **(1.879303 sec)** ja lassen Sie uns doch mal schauen $\Longrightarrow$ well let us see **(1.298260 sec)**

ja lassen Sie uns doch mal schauen hast Du Deinen Kalender vorliegen $\Longrightarrow$ well let us see do you have your agenda available **(2.899873 sec)**

ja lassen Sie uns doch mal schauen haben Sie Ihren Kalender gerade vorliegen $\Longrightarrow$ well let us see do you just have your agenda available **(2.581686 sec)**

ja lassen Sie uns doch mal schauen hast Du Deinen Kalender gerade da $\Longrightarrow$ well let us see do you just have your agenda available **(3.241916 sec)**

lass uns doch diesen Termin ausmachen $\Longrightarrow$ let us just fix this date **(1.065608 sec)**

lass uns doch einen Termin ausmachen $\Longrightarrow$ let us just fix a date **(1.084223 sec)**

lass uns doch einen solchen Termin ausmachen $\Longrightarrow$ let us just fix such a date **(1.212981 sec)**

lass uns doch solch einen Termin ausmachen $\Longrightarrow$ let us just fix such a date **(1.265757 sec)**

lassen Sie uns doch einen solchen Termin ausmachen $\Longrightarrow$ let us just fix such a date **(1.259212 sec)**

lassen Sie uns doch einen Termin ausmachen $\Longrightarrow$ let us just fix a date **(1.166236 sec)**

lassen Sie uns doch solch einen Termin ausmachen $\Longrightarrow$ let us just fix such a date **(1.203139 sec)**

prima dann lassen Sie uns doch diesen Termin ausmachen $\Longrightarrow$ fine then let us just fix this date **(1.609818 sec)**

ja lassen Sie uns doch mal schauen wann waere es recht $\Longrightarrow$ well let us see when would it suit you best **(2.319408 sec)**

ja lassen Sie uns doch mal schauen wann waere es Ihnen recht $\Longrightarrow$ well let us see when would it suit you best **(2.079591 sec)** ja lass uns doch mal schauen wann waere es Dir recht $\Longrightarrow$ well let us see when would it suit you best **(2.085727 sec)**

ja lass uns doch mal schauen wann waere es Dir am ehesten recht $\Longrightarrow$ well let us see when would it suit you best **(2.318236 sec)**

B.5 VM-PROPOSAL

ich koennte Ihnen den 3-ten Januar anbieten $\Longrightarrow$ I would propose January the 3rd **(1.997028 sec)**

ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ I would propose January the 3rd to the 10th **(2.885111 sec)**

mal sehen ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ well I would propose January the 3rd to the 10th **(3.182099 sec)**

mal schauen ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ well I would propose January the 3rd to the 10th **(4.019270 sec)**

oh ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ I would propose January the 3rd to the 10th **(4.026952 sec)**

aber ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ but I would propose January the 3rd to the 10th **(3.091083 sec)**

ja ich koennte Ihnen vom 3-ten bis 10-ten Januar anbieten $\Longrightarrow$ well I would propose January the 3rd to the 10th **(3.749869 sec)**

wie waere es mit dem 7-ten Januar aber nicht am Nachmittag $\Longrightarrow$ what about January the 7th but not in the afternoon **(6.088878 sec)**

wie waere es mit dem 7-ten Januar aber nur am Nachmittag $\Longrightarrow$ what about January the 7th but only in the afternoon **(8.750999 sec)**

wie waere es mit dem 7-ten Januar aber nicht nachmittags $\Longrightarrow$ what about January the 7th but not in the afternoon **(8.357143 sec)** wie waere es mit dem 7-ten Januar aber nur nachmittags $\Longrightarrow$ what about January the 7th but only in the afternoon **(5.223726 sec)**

am naechsten Dienstag haette ich noch einen Termin frei $\Longrightarrow$ I would have time on the following Tuesday **(2.027760 sec)**

am Dienstag den 9-ten Januar haette ich noch einen Termin frei $\Longrightarrow$ I would have time on Tuesday January the 9th **(4.455410 sec)**

B.6 VM-PROPOSAL-REACTION

nein Dienstag ist schlecht $\Longrightarrow$ oh no Tuesday is bad **(0.860921 sec)**

oh nein Dienstag ist schlecht $\Longrightarrow$ well oh no Tuesday is bad **(1.071660 sec)**

ja Dienstag ist prima $\Longrightarrow$ oh yes Tuesday is fine **(0.878548 sec)**

ja Dienstag ist ganz prima $\Longrightarrow$ oh yes Tuesday is absolutely fine **(0.959341 sec)**

ja Dienstag ist prima fuer mich $\Longrightarrow$ oh yes Tuesday is fine with me **(1.138630 sec)**

ja Dienstag ist ganz prima mit meinen Terminen $\Longrightarrow$ oh yes Tuesday is absolutely fine with my appointments **(1.422854 sec)**

nein da kann ich nicht $\Longrightarrow$ oh no that does not work with me **(2.031872 sec)**

nein da kann prinzipiell ich nicht $\Longrightarrow$ oh no basically that does not work with me **(1.476331 sec)**

ja das ist ganz prima bei mir mit meinen Terminen $\Longrightarrow$ oh yes that is absolutely fine with me with my appointments **(1.750520 sec)**

B.7 VM-AGREEMENT

gut ich trage es bei mir ein $\Longrightarrow$ fine I am taking down the date **(0.716945 sec)**

prima ich trage es bei mir ein $\Longrightarrow$ fine I am taking down the date **(0.669607 sec)**

ok ich trage den Termin bei mir ein $\Longrightarrow$ ok I am taking down the date in my agenda **(0.742390 sec)**

halten wir den Termin fest $\Longrightarrow$ I am taking down the date **(0.535127 sec)**

B.8 VM-BYE

bis dann in der schweiz auf wiedersehen $\Longrightarrow$ see you in Switzerland good bye **(0.656059 sec)**

tschuess bis dann $\Longrightarrow$ bye bye see you **(0.375373 sec)**

auf Wiedersehen bis dann $\Longrightarrow$ good bye see you **(0.414960 sec)**

auf Wiedersehen $\Longrightarrow$ good bye **(0.297690 sec)**

B.9 VM-BYE-REACTION

bis dann in der schweiz auf wiedersehen $\Longrightarrow$ see you in Switzerland good bye **(0.656059 sec)**

tschuess bis dann $\Longrightarrow$ bye bye see you **(0.375373 sec)**

auf Wiedersehen bis dann $\Longrightarrow$ good bye see you **(0.414960 sec)**

auf Wiedersehen $\Longrightarrow$ good bye **(0.297690 sec)**

B.10 Reversibility

As mentioned before, the grammar is reversible as long as it doesn't contain empty trees in the source grammar. Thereby the subgrammars VM-INTRODUCTION, VM-INTRODUCTION-REACTION, VM-TOPIC, VM-TOPIC-REACTION, VM-PROPOSAL, VM-BYE and VM-BYE-REACTION are not reversible! The following reverse examples originate from VM-PROPOSAL-REACTION.

VM-PROPOSAL-REACTION:

oh no Tuesday is bad $\Longrightarrow$ nein Dienstag ist schlecht **(0.896618 sec)**

oh yes Tuesday is fine $\Longrightarrow$ ja Dienstag ist prima **(0.780978 sec)**

oh yes Tuesday is fine with me $\Longrightarrow$ ja Dienstag ist prima fuer mich **(1.177638 sec)**

oh no basically that does not work with me $\Longrightarrow$ nein das geht prinzipiell nicht fuer mich $\Longrightarrow$ nein nein da kann prinzipiell ich nicht **(1.605726 sec for both readings)**