Reuse of Plan–Based Knowledge Sources in a Uniform TAG–based Generation System

Karin Harbusch & Jens Woch

Universität Koblenz–Landau; Abt. Koblenz
Rheinau 1, D–56075 Koblenz
e–mail: {harbusch, woch}@uni–koblenz.de

Abstract

In an uniform generation system all knowledge bases are specified in the same formalism and run the same processing component. The advantage of this behavior is that any order of applying the knowledge bases, i.e. a negotiation on revisions between the individual components, can easily be imposed on the system. Furthermore, the implementation of the overall system is simpler because only one algorithm must be developed and tested. In the project INTEGENINE *we specify all knowledge sources in the formalism of Schema–TAGs with Unification (SU–TAGs). A general paradigm of our work is to reuse existing knowledge bases, i.e. to transform various formats into a SU–TAG. For the syntactic and lexical knowledge the existing XTAG system has already automatically been transformed. In this paper we address the general question how to transform plan–based knowledge–sources which are frequently used in the what–to–say part of a generation system. As an instance of the general transformation model presented here, we show how to transform the knowledge sources of the plan–based system VOTE.*

The transformation component we describe in the following appertain to a *uniform generation system* based on *Schema–TAGs*. Let us first briefly address this system in order to motivate the serviceableness of the transformation component in the general system.

The idea of *uniform* or so called *integrated generation* was basically described in the system KAMP (Appelt, 1985). In this system a hierarchical action planner explores expressions based on the formalism of intensional modal logic. KAMP was not intended to be a psycholinguistic model of human behavior, although is reflects some aspects of human language production such as *incrementality*. This behavior directly results from the integrated model. Any knowledge base is supposed to become active at any time, i.e. as early as possible.

From this observation the question arises whether the uniform model can serve as a basis to remedy the generation gap (Meteer, 1990), i.e. the situation in which a sequential process (first what–to–say, then how–to–say) leads to dead end situations which cannot be solved by local modifications in the component in which the problem occurs. Our assumption is to extend the — in a sense demon–like — activation of knowledge bases towards a *parametrised model* which allows for recovery strategies to escape from local dead ends by imposing revisions of parameter–defined components. This means that parameters trigger the activation of specific knowledge bases and hence initiate overall revisions. Our claim is that this approach is able to build up any kind of communication model in a generation system.

As underlying formalism of our integrated generation model we have chosen *Schema–TAGs with Unification (SU–TAGs)*[1] because TAGs provide the necessary complexity to express any kind of concept in the what–to–say and how–to–say component (cf., e.g., (Stone & Doran, 1997), (Webber & Joshi, 1998), (Becker *et al.*, 1998), (Nicolov, 1998)). Schema–TAGs are especially

[1] In a *schematic elementary tree*, a *regular expression (RX)*, is annotated at each inner node of an elementary tree. This means, that the elementary schemata enumerate a possibly infinite set of elementary trees. RXs are inductively defined. Let α, β and $\beta_1, ..., \beta_n$ ($n \geq 2$) be Gorn addresses uniquely referring to daughters or arbitrarily complex RXs, then $\alpha.\beta$ (concatenation of branches), $(\beta_1 + ... + \beta_n)$ (enumeration of alternatives), α^* (Kleene Star) and α^+ (α^* without the empty repetition) are RXs. Finally, "–" allows to cut off a subtree at the end of a path. As an abbreviation we write $\alpha^{(n|m)}$ which enumerates $\sum_{i=0}^{m} \alpha^{n+i}$ ($n, m \geq 0$). Notice, "." binds stronger than "+". Notice also, that here the feature specifications are attached to the regular expression because the branches are licensed by RXs (cf. (Harbusch & Woch, 2000)).

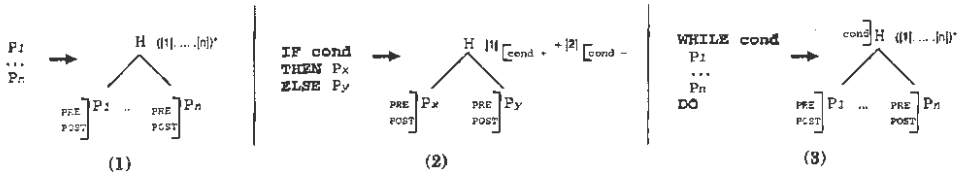

Figure 1: Transformation of sequences, alternatives, and repetitions into SU–TAGs

advantageous because they compress grammars in a manner that allows for the underspecified generation of substructures (cf. (Harbusch, 2000), (Harbusch & Woch, 2000)).
In order to provide a flexible generation system, the example domain is not of particular interest but only a necessary prerequisite of a demonstration system. On that account, we decide to *reuse* existing knowledge sources to circumvent the time–consuming task of developing a knowledge base from scratch. Thus, transformation algorithms for the individual knowledge bases of a generation system must be provided. Any TAG can automatically be transformed into a SU–TAG (Harbusch *et al.*, 1998). This is already done for the syntactic knowledge base XTAG (Doran *et al.*, 1994). The knowledge sources of SPUD (Stone & Doran, 1997), those of anchored L–TAGs (cf. (Webber & Joshi, 1998)), as well as the TAG transformed from an HPSG (Becker *et al.*, 1998) will be rewritten as SU–TAG next. Doing so, the generation system is extended towards a *generation workbench* which provides libraries with knowledge sources from which the user can select a personal generation system with self–defined parameters.
In this paper, we describe the transformation of *plan–based knowledge bases* into a SU–TAG. Here we only concentrate on the particular class of plans which is widely applied in what–to–say components of generation systems (e.g. VOTE (Slade, 1994)), i.e. the classical *plan–based plans* (cf. (Yang, 1997)). As an illustration a concrete plan of the system VOTE is transformed and the decision making on the basis of VOTE's further knowledge bases is presented. Finally, the interaction of plans with the system's knowledge about the domain — also specified as SU–TAG — is outlined in order to demonstrate basically, how the uniform generation works.

A *plan*[2] consists of n steps, any of them in turn may be an *action* or a plan again. Each step consists of *pre–* and *postconditions*, as well as controlling elements of a programming language (e.g. `IF-THEN-ELSE`, `WHILE`). A plan can be applied iff the overall *goal*, i.e. the input specification, matches the preconditions of the first step. A plan step can be applied iff the *current situation*, i.e. the postconditions of the previous step, or the input specifications respectively, match the preconditions of the currently considered plan step. If a plan step is atomic, i.e. an action, it is *performed* by replacing the preconditions with the postconditions, resulting in the new current situation. An overall plan can *successfully be applied in the current situation* iff the final postconditions can be computed according to the overall goal and the initial situation. Given that, the general idea of the transformation into a SU–TAG is as follows:

1. Each plan step in a sequence becomes an individual node of an elementary scheme under a common root node.
2. The chronological sequence of plan steps is rewritten via concatenation in the RX.
3. Pre– and postconditions at each node are wrapped up in feature specifications.
4. The conditions of concepts of the programming language are realized by unification too, whilst the branches and repetitions itself are transformed into RXs.

In the transformation of Fig. 1-1, the first three steps are illustrated. Each plan step P_1, ..., P_n is transformed into a daughter node. The regular expression at the root node enumerates the concatenation of all daughters from left to right and all pre– and postconditions are rewritten as feature specifications. Step 4 is illustrated by two example statements in Fig. 1-2 and 1-3. Basically, the conditions in the statements are checked by a feature "cond". For instance,

[2]For an illustration of a plan, see the strategy for decision making in VOTE (abstracting from technical notations, cf. (Slade, 1994), p. 140) on the left side of Fig. 2.

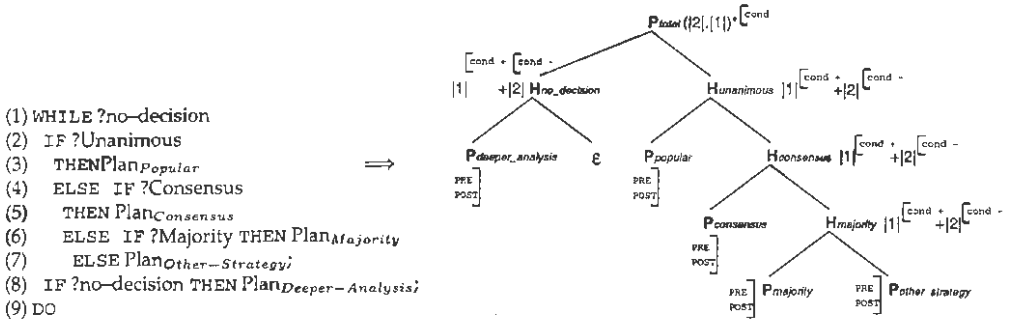

Figure 2: Transformation of the VOTE–Plan

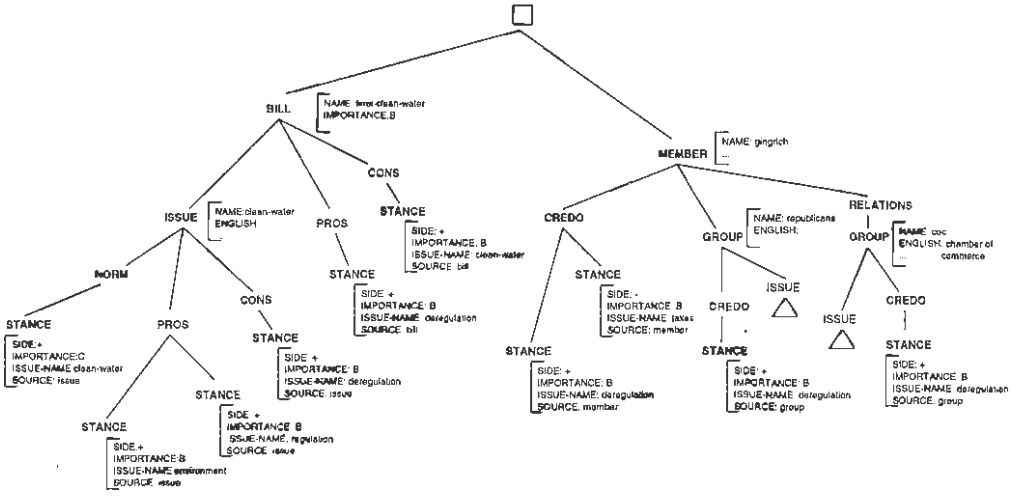

Figure 3: Part of VOTE's knowledge for MEMBER:gingrich and BILL:limit–clean–water

in the IF-THEN-ELSE statement a positive value for "cond" activates the THEN part and a negative value the ELSE part. The behavior of the steps 1, 2 and 4 is exemplified in Fig. 2. This plan describes the decision making process in the system VOTE. The actual knowledge in the pre– and postconditions is suppressed here for reasons of simplicity. As outlined in step 4, IF-THEN-ELSE statements are rewritten as sums, (i.e. representing the choice of one of the branches according to the instantiation of the condition) and the WHILE construction is rewritten as Kleene Star which stops according to the instantiation of the respective condition represented as feature specification. At the root node the concatenation represents the sequence of the two IF-THEN-ELSE statements in line (2) and (8) (step 1 and 3 result in |2|.|1| according to the order of branches). Here, the Kleene Star in the RX rewrites line (1).

Now we explain how pre– and postconditions are specified and tested in order to apply plans in this particular example. For this reason we must describe VOTE's further knowledge bases in more detail: VOTE consists of ISSUES (e.g. *gun control*), STANCES (PRO, CON, normal case), GROUPS (e.g. *ACLU*), RELATIONSHIPS, MEMBERS, BILLS and STRATEGIES. Fig. 3 shows the structure of what VOTE knows about a concrete BILL:limit–clean–water and the attitude of a concrete MEMBER:gingrich towards this bill. Let us presuppose here that the structures described in Fig. 3 can be produced by SU–TAG structures of the form outlined in Fig. 4. This is directly obvious because all mother–daughter relations in the instantiation are represented as elementary schemata. Furthermore, any scheme licenses the specification of any number of such relations by Kleene Star. In any plan of VOTE the pre– and postconditions are yet specified by unification about STANCEs of bills and members. For instance in Plan$_{popular}$, PRE = {Unify

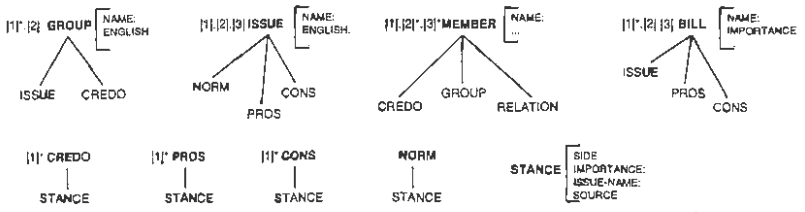

Figure 4: SU–TAG representation of the knowledge base classes BILL and MEMBER

all SIDE features of all STANCEs}, i.e. test whether they have the same value and POST = {Unify the feature DECISION at the root node with the SIDE feature of the uppermost STANCE of BILL}, i.e. vote in a popular manner. Hence, in the uniform framework the application of a plan imposes further constraints on the knowledge about bills and members. The reasoning about plans and domain knowledge is performed integratedly in a unform manner.
In general, pre– and postconditions can also be specified in first order predicate logic. Let us consider the STRIPS example in (Yang, 1997) p. 17. For instance, the plan return–brush with PRE = {have–brush(?b)} and POST = {¬ have–brush(?b)} is rewritten by the feature specifications PRE = {((b have–brush) +)} and POST = {((b have–brush) –)}. For reasons of space we cannot go into more details here (cf. (Otto *et al.*, 1998)).
The implementation of the above described transformation component has just begun using general parser generator concepts in the same way as for the TAG–to–STAG transformation.

References

A. ABEILLÉ, T. BECKER, O. RAMBOW, G. SATTA & K. VIJAY-SHANKER, Eds. (1998). *Procs. of the 4th International TAG Workshop*, IRCS-Report 98–12.

APPELT D. E. (1985). *Planning English Sentences*. Cambridge University Press.

BECKER T., FINKLER W., KILGER A. & POLLER P. (1998). An efficient kernel for multilingual generation in speech–to–speech dialogue translation. In (Isabelle, 1998), p. 110–116.

DORAN C., EGEDI D., HOCKEY B. A., SRINIVAS B. & ZAIDEL M. (1994). XTAG system — a wide coverage grammar for english. In M. NAGAO, Ed., *Procs of the 15th COLING*, Kyoto, Japan.

HARBUSCH K. (2000). Natural-language processing with Schema–TAGs. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree Adjoining Grammars: Formal Properties, Linguistic Theory and Applications*. CLSI Lecture Notes.

HARBUSCH K., WIDMANN F. & WOCH J. (1998). Towards a workbench for Schema-TAGs. In (Abeillé *et al.*, 1998), p. 56–61. IRCS-Report 98–12.

HARBUSCH K. & WOCH J. (2000). Direct parsing of Schema–TAGs. In H. C. BUNT, Ed., *Procs. of the Sixth International Workshop on Parsing Technologies (IWPT)*, p. 305–306, Trento, Italy.

P. ISABELLE, Ed. (1998). *Procs. of the 36th ACL and 17th COLING*. Université de Montréal, Canada, Morgan Kaufmann.

METEER M. W. (1990). *The Generation Gap – The Problem of Expressibility in Text-Planning*. PhD thesis, University of Massachusetts, Amherst, MA, USA.

NICOLOV N. (1998). Memoisation in sentence generation with lexicalized grammars. In (Abeillé *et al.*, 1998), p. 124–127. IRCS-Report 98–12.

OTTO F., NARENDRAN P. & DOUGHERTY D. J. (1998). Equational unification, word unification, and 2nd-order equational unification. *Theoretical Computer Science*, 198 (1–2 p.), 1–47.

SLADE S. (1994). *Goal–Based Decision Making: An Interpersonal Model*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates Inc.

STONE M. & DORAN C. (1997). Sentence planning as description using Tree Adjoining Grammar. In *Procs. of the 35th ACL and 8th EACL*, Madrid, Spain.

WEBBER B. L. & JOSHI A. K. (1998). Anchoring a Lexicalized Tree–Adjoining Grammar for discourse. In (Isabelle, 1998).

YANG Q. (1997). *Intelligent Planning*. Berlin, Germany: Springer-Verlag.