Practical, Template–Based Natural Language Generation with TAG

Tilman Becker
DFKI GmbH

1. Introduction

This paper describes a TAG–based (Joshi and Schabes, 1997), template–based approach to Natural Language Generation. It extends the idea of lexicalization to whole phrases, similar in style to the representation of idioms in a TAG grammar. In addition to this, there is a second type of templates: partial derivations.

The first phase in the generator, driven by planning rules, produces a derivation tree which is then used in a straightforward realizer to create the derived tree. This tree is then the basis for the input to a Concept–To–Speech synthesis component (Schweitzer, Braunschweiler and Morais, 2002).

There are two basic methods for constructing these two kinds of templates. The first and preferred one is based on an existing grammar where the templates represent shortcuts that would be used for reasons as simplicity, efficiency, or because the existing grammar does not contain an interface to the representation language, e.g., semantics, that is used as the generator‘s input. The second method which is used so far in the SMARTKOM project (Wahlster, Reithinger and Blocher, 2001) is necessary when no suitable grammar exists. It allows for an approach similar to rapid prototyping: only the templates that are needed are specified and the templates are initially kept as large as possible. Only in the following development steps, the templates are made smaller and smaller, eventually themselves becoming a grammar.

The following sections briefly present the SMARTKOM project and the generator architecture. Then the use of fully specified templates is discussed in this context, including their use for concept–to–speech synthesis. Finally we present the current work, including some tools that are under development.

2. The SmartKom Project

SMARTKOM is a multi-modal, dialogue system currently being developed at several academic and industrial partners (see `www.smartkom.org`). User input modalities are speech, including prosody, various gestures,[1] and an interpretation of the user‘s facial expression. The system output is a combination of graphical presentations and an animated, talking agent, Smartakus, plus various external effects such as controlling a VCR, sending e-mail, querying databases etc.

The key idea behind the system is to develop a kernel system which can be used within several scenarios and applications. Currently there exist three such scenarios – *public*, *home*, and *mobile* – with a number of applications such as EPG (electronic programming guide), scanning and sending documents, route finding, etc. They all are different in their external appearance but share most of the basic processing techniques. The system depicted in Figure 1 is the “public scenario” system. Within this scenario, an intelligent kiosk is developed with which one is able to access the various applications. Development is mainly for German, but some applications are also ported to English.

2.1. Architecture

Figure 2 shows the overall architecture of the generation system which follows a straightforward classical pipeline design. On the interfaces to input and output, XSLT stylesheet transformations adapt the various formats, while internally there are two important sub–components: Preplan, a simple planning engine that maps from the presentation goals to derivation trees and a TAG component that implements TAG trees with their operations, including full feature unification.

Preplan is a top–down rule expanding planner, implemented in Java. Rules have associated constraints that can refer to knowledge bases that are constructed from the input to the generator. The planner then matches parts of this input to select appropriate templates (i.e., partial derivation trees) and fills them with data from the input.

1. analyzed through an infrared camera, thus obliterating the need for a touch screen.

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 80–83. Universitá di Venezia.

Figure 1: Multi-modal interaction with the system.

e.g., the name of the user is taken from the input and inserted into an existing tree for a name, overwriting an uninstantiated lexical item in this tree.

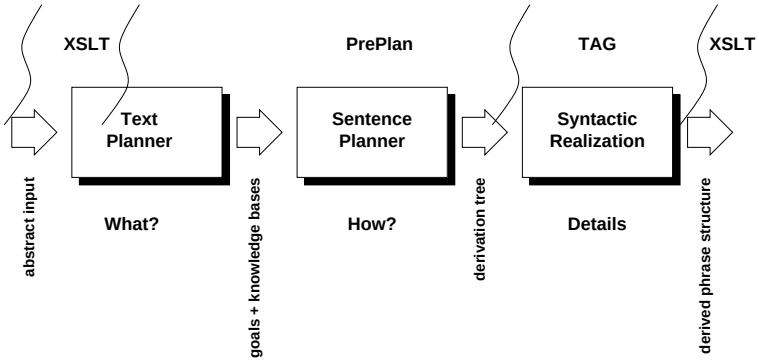

Figure 2: Classical pipeline architecture of the generator.

3. Fully Specified Templates

The design decision to use template–based generation in SMARTKOM, is driven by practical considerations: First, generator output was needed as early as possible in the project. Since there are a number of non–overlapping, but well–defined applications in SMARTKOM, output for a new application has to be created in a short time, suggesting the use of templates. On the other hand, simple string concatenation is not sufficient. E.g., for integrating Concept–to–Speech information, especially in the way the synthesis component of SMARTKOM is designed(Schweitzer, Braunschweiler and Morais, 2002), calls for an elaborate syntactic representation, i.e., phrase structure trees with features, to guide the decisions on prosodic boundaries. At least since (Reiter, 1995) (also see (Becker and Busemann, 1999)), the use of templates and "deep representations" is not seen as a contradiction. Picking up on this idea, the generation component in SMARTKOM is based on fully lexicalized generation (Becker, 1998), packing whole parts of a sentence together into one *fully specified template*, representing them not as a string but rather as a partial TAG derivation tree. See also figure 3. All nodes in the TAG trees carry top and bottom feature structures, see (Vijay-Shanker, 1987), which also contain discourse information as needed for concept–to–speech synthesis. The current setup uses fully inflected forms, but this is due to be changed soon and the changes will be minimal. A call to a morphology module will get inflectional information from the feature

structures and stem information from the leaf as, e.g., in (Becker, 1998).

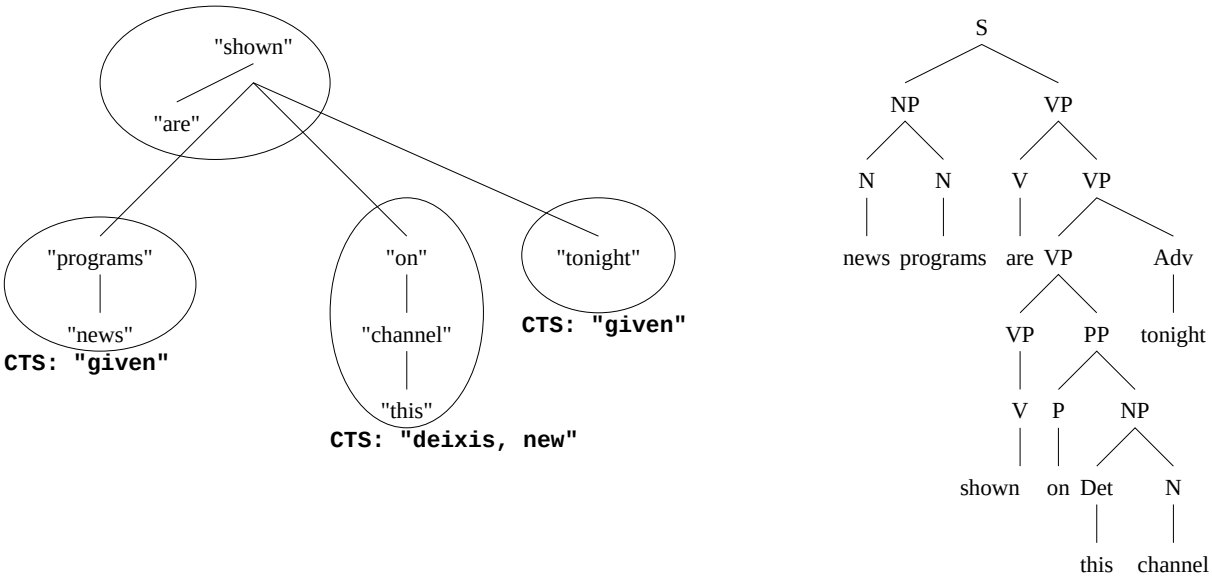

Figure 3: Derivation tree with CTS (Concept–to–Speech) markup. Each ellipse is a fully specified template. The sentence–planning process combines such templates to a complete derivation tree.

3.1. Specifying only Relevant Representations

With this approach to specify intermediate levels of representation (which commonly only exist inside the NLG module), the question remains whether *all* levels of representation have to be specified. Clearly, this is desirable, but not necessary in SMARTKOM. Thus only the level of dependency and phrase structure are represented fully. Dependency is necessary to guide the top–down generation approach, phrase structure is necessary for (Concept–to–)speech synthesis. However, there is nothing preventing the later inclusion of other levels of representation, e.g., adding a level of semantic representation (e.g., (Joshi and Vijay-Shanker, 1999)) which might be used by a sentence aggregation component.

3.2. Dependency and Speech–Markup

Specifying templates on the level of derivation trees rather than on derived trees or even strings has several advantages. In the context of Concept–to–Speech synthesis, it is necessary to add markup to parts of the string. This can be done easily by adding the information to the corresponding node in the derivation tree from where it is percolated and automatically distributed to the corresponding parts of the utterance when constructing the derived tree and the string from the derivation tree. Such markup relates to parts of the output that have to (de-)emphasized, parts that refer to objects in the graphical presentation and must be coordinated with a pointing gesture.

Figure 3 shows a derivation tree with speech-relevant markup on some nodes. Besides mere convenience in the markup[2] the additional power of TAG allows the distribution of semantically connected markup to discontinuous parts in the final string. Since formal complexity is a very different issue in generation than in parsing, we are open to the use of extensions from the standard TAG formalism as in (Rambow, 1994) or (Gerdes and Kahane, 2001) which might be necessary for German.

3.3. Tools and Current Work

Currently we have editors available for the planning rules and the TAG-tree templates. Both build on XML

2. E.g., XML-style opening and closing parentheses can be integrated into the trees and thus are realized by a single marked node vs. the situation in a classical context–free based string-expanding template generator, where opening and closing elements have to be denoted independently–a typical source for errors.

representations of the knowledge bases and present them in an easily accessible format: a directory structure[3] as known from the Windows Explorer for the set of trees and a graphical tree editor for the TAG-trees.[4]

Current work is centered around adding templates for new applications and has shown that managing a large set of templates can be problematic. Eventually we hope to switch to the first development method as mentioned in the introduction: We plan to extend the rule editor with a TAG-parser. To add a new template to the generator, the user will then type in an example sentence, have it parsed, select the correct parse, mark (delete) the variable parts, keeping the fixed part and add the remainder of the rule. Thus rules can be created without ever writing trees by hand.[5]

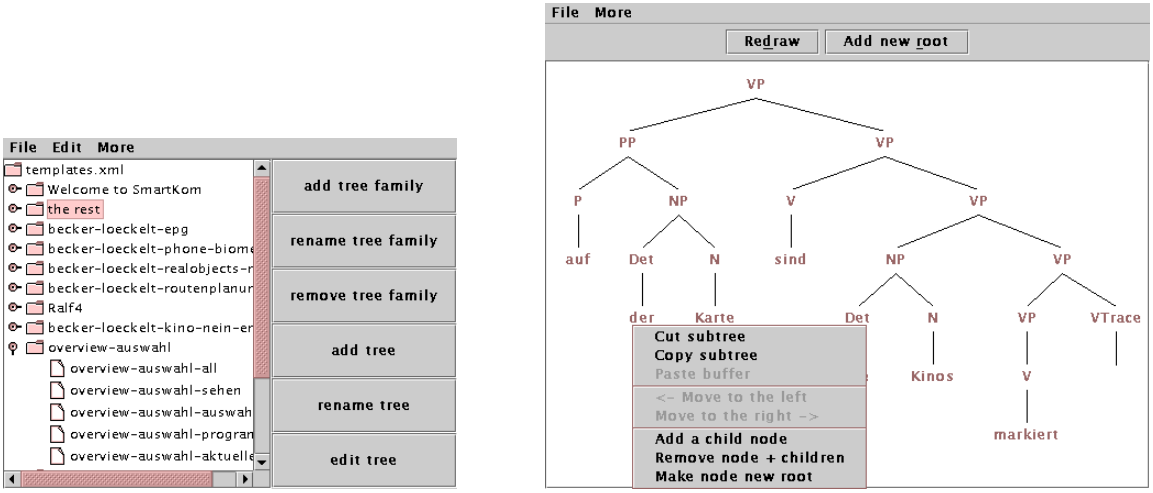

Figure 4: The grammar and tree editor tools.

References

Becker, Tilman. 1998. Fully lexicalized head-driven syntactic generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada, August.

Becker, Tilman and Stephan Busemann, editors. 1999. *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation.*, Bonn, September. Workshop at the 23rd German Annual Conference for Artificial Intelligence (KI '99).

Gerdes, Kim and Sylvain Kahane. 2001. Word order in German: A formal dependency grammar using a topological hierarchy. In *Proc. of ACL 2001*, Toulouse, France.

Joshi, A. and Y. Schabes, 1997. "Tree–Adjoining Grammars". In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–124. volume 3. Berlin, New York: Springer.

Joshi, A. K. and K. Vijay-Shanker. 1999. Compositional Semantics for Lexicalized Tree-Adjoining Grammars. In *Proc. of the 3rd International Workshop on Computational Semantics*, Utrecht, The Netherlands, January.

Rambow, Owen. 1994. *Formal and Computational Models for Natural Language Syntax.* Ph.D. thesis, University of Pennsylvania, Philadelphia. Available as Technical Report 94-08 from the Institute for Research in Cognitive Science (IRCS).

Reiter, Ehud. 1995. NLG vs. templates.

Schweitzer, Antje, Norbert Braunschweiler and Edmilson Morais. 2002. Prosody Generation in the SmartKom Project. In Bernard Bel and Isabel Marlien, editors, *Proceedings of Speech Prosody 2002*, pages 639–642, Aix-en-Provence, France.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Wahlster, Wolfgang, Norbert Reithinger and Anselm Blocher. 2001. SmartKom: Multimodal Communication with a Life-Like Character. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, September.

3. implemented with JTree
4. A transformation of the XTAG grammar into the XML format is in the works, the main problem is parsing and translating the feature equations. Eventually the feature macros need to be incorporated into the editor, too.
5. Which will also avoid inconsistencies in the set of templates.