Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

**Research
Report**
RR-91-25

Incremental Syntax Generation with Tree Adjoining Grammars

Karin Harbusch Wolfgang Finkler Anne Schauder

October 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Philips, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Incremental Syntax Generation with Tree Adjoining Grammars

Karin Harbusch Wolfgang Finkler Anne Schauder

DFKI-RR-91-25

A version of this paper has been published in the Proceedings of the 4th International GI Congress: "KNOWLEDGE-BASED SYSTEMS" Distributed Artificial Intelligence and Cooperative Work, October 23rd and 24th, 1991.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8901 8).

Incremental Syntax Generation with Tree Adjoining Grammars

Karin Harbusch, Wolfgang Finkler, Anne Schauder

Authors' Abstract

With the increasing capacity of AI systems the design of human–computer interfaces has become a favorite research topic in AI. In this paper we focus on aspects of the output of a computer. The architecture of a sentence generation component – embedded in the WIP system – is described. The main emphasis is laid on the motivation for the incremental style of processing and the encoding of adequate linguistic units as rules of a Lexicalized Tree Adjoining Grammar with Unification.

Contents

1 Introduction

The acceptance of an AI system directly depends on its user interface facilities. For the next generation of user interfaces it is no longer acceptable to use a restricted input mode (e.g., an inflexible sequence of prompts of the system and required user input) or inadequate canned text as output (e.g., "Rule 31 was chosen because of precondition 2.").

An AI system with user interface should be capable to play the role of the producer as well as the role of the consument in conversational situations. Therefore it must be able to analyze user input and to present new information in an adequate way. To be adequate human–computer dialogue must share the properties of human–human dialogue. This means for the generation of ouput that e.g., size and granularity of the presented information should depend on the communicative context (novices must be informed in a more detailed manner than skilled persons).

The central topic of this paper is the generation of natural language output. Since natural language analysis and generation share a lot of properties, they could be seen as two directions of the same process. In order to motivate the extra research on natural language generation we have to differentiate between the two topics. One example for such a difference is given by the used search spaces: During analysis all syntactic and semantic ambiguities of an input sentence (e.g., "Time flies like an arrow.") must be resolved in order to find the intended meaning. Generation means to choose between an infinite number of utterances which are possible but more or less adequate in a situation ("Would you please close the window?", "Shut the window!", "It is cold here.", "I already had three influenzas this year.", ...).

Our generation component is embedded in the WIP[1] system whose architecture is presented in Section 2. The decision for an incremental processing mode is essential for the architecture of the generation component as well as for the determination of size and shape of processed linguistic units. We motivate this processing mode with psycholinguistical and computational arguments in Section 3. In Section 4 we argue that the formalism of Lexicalized LD/LP–Tree Adjoining Grammars with Unification is well suited for the representation of the knowledge used during verbalization (grammar and lexicon). The overall architecture of our sentence generation module is explained in Section 5. It is a system of concurrently and cooperatively working objects which manage the composition and linearization of the linguistic units.

2 The WIP System

The aim of the WIP project (see, e.g., [Wahlster et al. 91]) is to contribute in basic research in the field of 'Intelligent User Interfaces'. With increases in the amount and

[1] WIP is the acronym for "**W**issensbasierte **I**nformationspräsentation" which means knowledge–based presentation of information. The WIP project is supported by the German Ministry of Research and Technology under grant ITW8901 8.

sophistication of information that must be communicated to the user of a complex technical system, new ways to present that information flexibly and efficiently become necessary. Since in many situations information is presented efficiently only through particular combinations of communication modes (e.g., graphics and text), the automatic generation of multimodal presentations is an important task of such a system. One basic principle of the WIP system is that the various constituents of a multimodal presentation should be generated dynamically from a common representation, i.e. no predefined pictures or texts are used. This is the presupposition for flexibility (e.g., change of perspective) and interrelationship in the different modes of output (e.g., the use of cross–modal references).

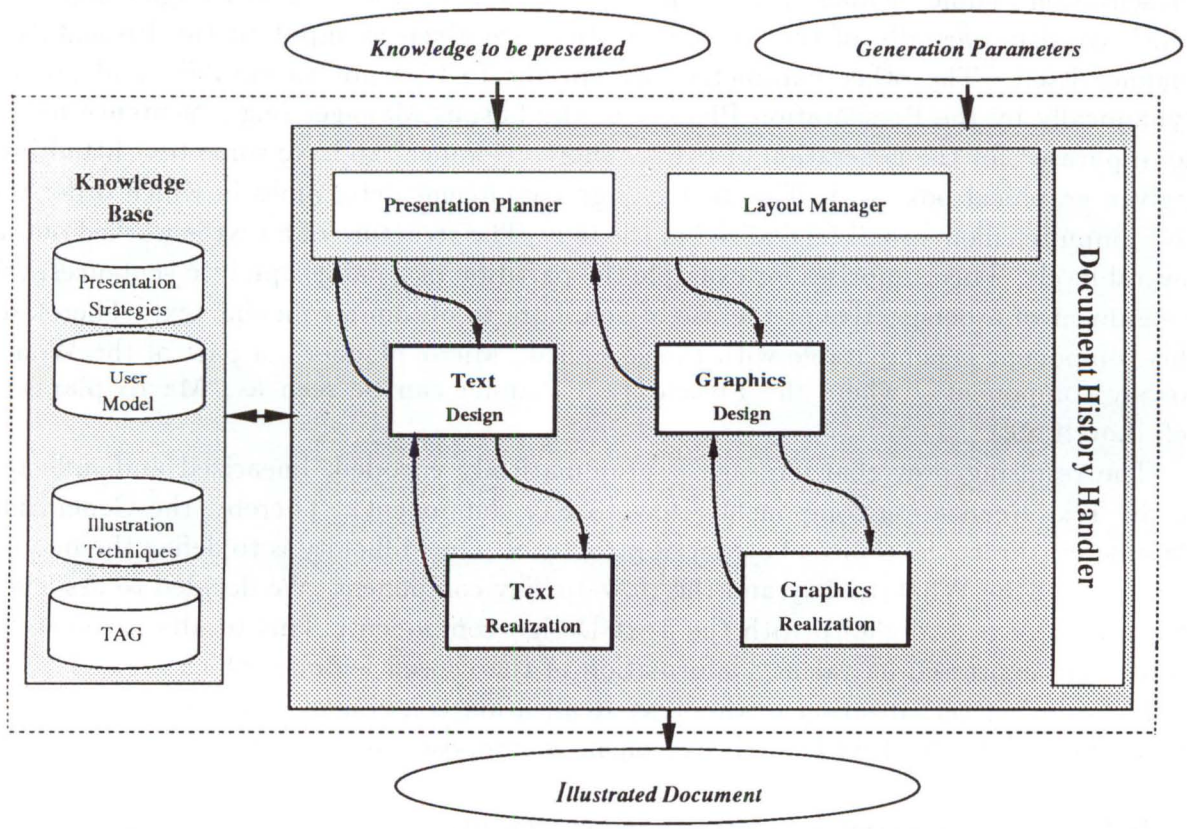

Figure 1: The Architecture of the WIP System

The *Presentation Planner* (see Figure 1) is the component that is responsible for the selection of contents. Furthermore, it decides which parts of the presentation shall be realized as text and which as graphics. The input for the planner consists of the knowledge to be presented together with some information about the communicative context in form of so–called *Generation Parameters*. A second control instance of the system is the *Layout Manager*. It uses semantic and pragmatic relations specified by the planner to arrange the graphical and textual fragments produced by the two mode–

specific generators (see W. Graf and W. Maaß in this volume for detailed information about the Layout Manager). The two generator components are divided into a design and a realization part. They produce textual and graphical parts of output without direct communication. References and influences between the two generators are nevertheless required for multimodal output: The *Document History Handler* as a central blackboard is the shared medium for communication between the different components. Here the need for a common representation for all components becomes obvious again. In the following we focus on the text generation part of WIP.

The Text Design component receives as input from the Presentation Planner exactly that piece of knowledge, which was chosen to be presented as text. Furthermore the Presentation Planner defines an adequate set of parameters for the Text Design component which consists partially of the parameters that are given as input to the Presentation Planner itself. The other parameters are specific to text not to graphics and are set dynamically by the Presentation Planner or the Layout Manager (e.g., "sentence mode: noun phrase" for the generation of titles, "short utterance" to have some text fitted into a given graphical box, ...). The Text Design component determines in which order the given input elements shall be realized in the text. The structure of a text is worked out at several levels. This comprises for example the partition of a paragraph into sentences, the assignment of a perspective or the use of anaphora to obtain a coherent text. Therefore, this component is comparable with the so-called "Micro-planner", a part of the What-to-Say component - where the Presentation Planner can be seen as "Macro-planner" (cf. [Levelt 89]).

The resulting preverbal message is grammatically encoded, linearized and inflected in the Text Realization component (How-to-Say component). Thereby, the Generation Parameters direct the choice of syntactic structures. One difficulty is to define the boundary between the What-to-Say and the How-to-Say component. We decided to associate the process of lexical choice with the Text Design component. This results in syntactic constraints expressed as valency information of the chosen lemmas which could lead to conflicts during verbalization in the Text Realization component. To be able to report these problems to the Text Design component we propose a model with feedback between the two modules.

In the following section we motivate the incremental style of processing during generation which is supported by such a model of a cascade with feedback.

3 Incremental Natural Language Generation

Incrementality stands for piecemeal processing of information. For the text generation task which is assumed to be executed on several stages of processing, this means an interconnection between the stages: Instead of working sequentially on the whole input the components work on partial input and produce parts of output for the next stage as soon as possible.

The idea of incrementality in natural language generation is motivated by psycholinguistic studies (see, e.g., [Kempen 78]). Humans often start speaking before they know exactly what the whole contents of their utterance will be. When describing a scene where a big man and a smaller man are visible and the big man does something, it is possible to focus on this man without knowing how to describe his activity. Thereby incremental processing is possible because the disambiguating modifier "big" can be added to the noun "man" without regarding the actions of the two persons. This observation can be interpreted as follows: The time that passes during the articulation of the first parts of the utterance can be used to process further parts of information which shall be realized in the utterance (e.g., that the action can be described as "laughing"). Special kinds of human speech errors and empirical results of other psycholinguistical experiments give evidence for this interpretation.

Advantages of incrementality in human language generation are that less information must be kept in mind, that pauses between parts of the utterance become shorter and consequentially that the danger of being interrupted by the dialogue partner decreases. Most important for AI generation systems are the first two facts.

From a computational point of view the main advantage of an incremental approach is the improvement of efficiency. A reduction of response time enhances the user's acceptance of an information system. Different components of the text generation system (see Section 5) can start working as soon as they obtain their first partial input. They should produce their output in a piecemeal fashion, too. Therefore, efficiency is gained by *parallelism* in the cascade of text generating components. On the contrary, a strictly sequential mode of operation would force them to work one after the other; their processing times would have to be added up.

There is a need for *incremental presentation* in WIP, especially, if knowledge to be presented is not completely available when the processing in the WIP system starts. A scenario can be imagined where information is continuously supplied by the application system (e.g., data obtained from measuring instruments) and where such information must be simultaneously presented in a condensed form to assist human decision–makers. In addition, an *online presentation mode* that aims to instantaneously illustrate the development of the document on a screen can be more appropriate for users even in situations where all information to be presented is available before the system starts. The advantage is that the user receives the reaction of the system earlier, i.e. before the output is completed.

For the transfer of an incremental approach to a computational model basically two questions must be examined: Firstly, how can an incremental and parallel mode between the components of the system (here, e.g., Presentation Planner, Text Design, ...) be realized, and secondly, can this kind of processing be furthermore enhanced by parallelism inside the components (e.g., parallel processing of parts of a sentence in the Text Realization component)?

Our answer to the first question is the previously mentioned model of a cascade for the

architecture of the WIP system. "Coarse grained" parallelism between the text generator, the graphics generator and the planning component is possible.

In this paper, we will try to answer the second question with regard to the Text Realization component. A simultaneous activity on existing parts of the syntactic structure would support the incremental processing of such parts (cf. [Finkler & Neumann 89]). A discussion about the degree of parallelism possible inside a component ends up in a discussion about the degree of independence that can be observed between the computed parts. On the other hand, this question has some influence on the determination of the size of entities the sentence generator should work with.

Our ideas to solve these problems are presented in the next two sections. We will translate the incremental mode of processing into a parallel model of cooperating objects which deal with entities from our knowledge base: structures of a Lexicalized LD/LP–Tree Adjoining Grammar with Unification.

4 The Grammar of the Text Realization Component

The task of our generator consists in finding a syntactic realization for the preverbal message which has been given as input. The additional information contained in the parameter settings has likewise to be integrated. Therefore we must have a look at the structure of syntactic knowledge and find a good representation for it.

We refine the problem stepwise. First an adequate formalism for the representation of syntactic knowledge is presented (Tree Adjoining Grammars with some extensions). Then the appropriateness of the TAG extension 'Lexicalized LD/LP-UTAG' for grammatical encoding is evaluated.

4.1 Lexicalized LD/LP–TAGs with Unification

Tree Adjoining Grammars

In 1975, the formalism of Tree Adjoining Grammars (TAGs) was introduced (see [Joshi et al. 75]. Since then, a wide variety of properties – formal properties as well as linguistically relevant ones – were studied (see, e.g., [Joshi 85] for a good overview).

A Tree Adjoining Grammar is a tree generation system. It consists of two different sets of trees, *initial* and *auxiliary trees*, which together form the set of *elementary trees*. Intuitively the set of initial trees can be seen as context–free derivation trees. The start symbol of the grammar is the root node, all inner nodes are nonterminals and all leaves are terminals (e.g., tree α in Figure 2). Auxiliary trees (e.g., tree β in Figure 2) can be combined with initial (or already modified initial) trees by replacing an inner node of the latter. This combination operation is called *adjoining* or *adjunction*. The result of an adjoining must again have the form of a derivation tree (e.g., the rightmost tree in

Figure 2). This demand restricts the structure of auxiliary trees: In order to replace a node labelled X, the root node of the auxiliary tree must also be labelled with X and one of its leaves, called the *foot node*, serves as the new root for the former subtree below the adjoining node. Another restriction for auxiliary trees is the requirement, that such a tree must derive at least one terminal. This disallows the repetition of arbitrary many adjunctions without growth in the terminal string.

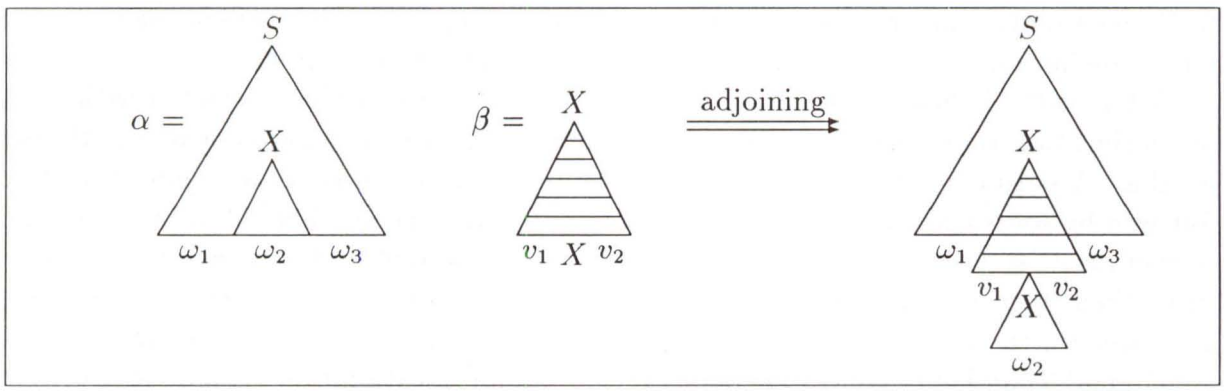

Figure 2: Elementary Trees and the Adjoining Operation in a TAG

The most obvious property of TAG trees which arises from the close relation with context–free derivation trees is the easy way to write and understand syntactic rules. The advantage over CFGs (context–free grammars) is that related facts can be described inside one rule (see, e.g., [Kroch & Joshi 85]). TAGs are more powerful than CFGs, they are mildly context–sensitive (cf. [Weir 88]). In the linguistic community, it is discussed controversially, how powerful a linguistic formalism should be (see, e.g., [Pullum 84]). Our decision for the TAG formalism is oriented at the thesis that natural language can be described very well by a mildly context-sensitive formalism.

One disadvantage of the TAG formalism is the redundancy of subtrees as a consequence of the relatively large size of the structures. One attempt to weaken this disadvantage is the introduction of a second combination operation – *substitution* – into the TAG formalism. This operation allows nonterminal leaves of elementary trees to be substituted by *substitution trees*. A substitution tree looks like an initial tree but the root node may be labelled with any nonterminal symbol. Substitution does not increase the power of the formalism because it can be eliminated by context–free preprocessing. Substitution nodes always mark places of obligatory expansions of a tree. A derivation tree is not completed until all substitutions have taken place.

The extensions of the TAG formalism that are described in the following are primarily linguistically motivated.

TAGs with Unification

Tree Adjoining Grammars – same as the CFGs – lack the necessary structures to express and compute complex attributes in the linguistic application domain, e.g., agreement. The encoding of such attributes (like person, number and case for nouns) in the category name (e.g., NP1plnom) leads to combinatory explosion of the grammar. In the framework of CFGs, this disadvantage is removed by combining the CFG with feature structures thereby defining a Unification Grammar (see [Shieber 86] for an introduction to Unification Grammar). This technique can also be applied to the TAG formalism. The formal definition is given in [Buschauer et al. 89] and in [Harbusch 90].

TAGs with Unification (UTAGs) allow each node of a tree to be associated with a list of specification rules. Such a rule consists either of a pair of two paths or of a path and a value. A path consists of a number uniquely referring to a node of an elementary tree followed by some feature names, e.g., ((0 pers) (01 pers)) means that the value of feature *person* must be shared by node 0 and node 01. If the second part of a rule is an atomic value then the rule specifies a 'definition'. All rules associated with nodes of a TAG tree are unified if there exists no contradiction between them, e.g., ((01 pers) 3) unified with ((001 pers) 2) and ((01 pers) (001 pers)) fails. The specification rules of a node are often represented as DAG (directed acyclic graph), where common prefixes are only represented once and different sons of a common prefix become brothers.

The trees in Figure 3 describe an example UTAG which allows the propagation of some syntactic information (here just the value of the feature *person*) from the lexical item "Mann" upwards. For reasons of simplification the determiner which is obligatory

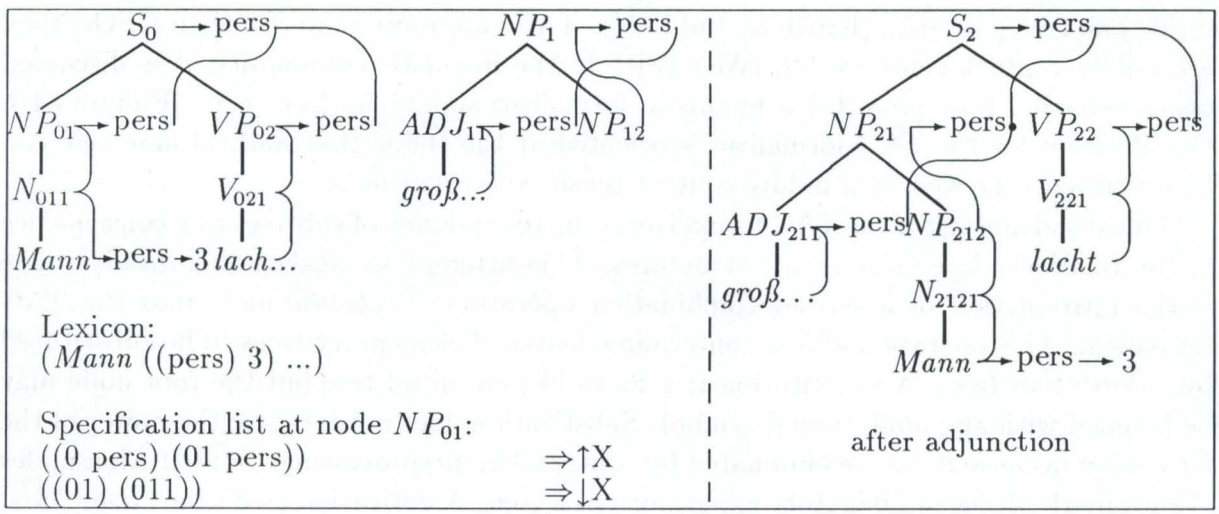

Figure 3: Trees of a TAG with Unification

for the represented german sentence is left out. The example illustrates that information specified in DAGs at nodes can be propagated upwards as well as downwards in the tree.

Thus there may be links (points of unification) between the DAGs all over the elementary tree, graphically represented as arcs in Figure 3. The question is how to manage these links during adjunction. If a node becomes an adjoining node it is replaced by an auxiliary tree. Thereby existing links from this node into the tree have to be cut and newly connected with parts of the auxiliary tree in order to fit in this tree correctly.

More formally spoken, we divide the specification lists of the adjoining node X into the three sets ↑X, ↓X and ∘X, stressing the different partner nodes mentioned in the rule: ↑X is the set of all specifications which relate X to its father or brothers in the tree, ↓X is the set of all specifications which relate X to its sons, ∘X is the set of value definitions for X. During the adjoining with unification at a node X, first this node is replaced by the auxiliary tree. All links of X upwards and downwards in the tree are cut–off. Then the DAG of the root node of the auxiliary tree is unified with ↑X and the DAG of the foot node is unified with ↓X (see the links in the resulting tree in Figure 3). The value definitions of ∘X are reintroduced at positions in the auxiliary tree, which are computed by a specific inheritance examination (see [Buschauer et al. 89]), to avoid unexpected unification failures. It becomes clear that TAGs can be naturally combined with the extending unification to form a representation for specific linguistic terms.

LD/LP–TAGs

Another disadvantage of the TAG formalism as well as for context–free grammars is that both formalisms cannot handle the problem of free word order efficiently. Each constellation of leaves has to be expressed by an individual tree. For CFGs this disadvantage was eliminated by defining two sets: *Immediate Dominance Rules (ID–rules)* contain information about the sons of a node without specifying their ordering (e.g., (NP → DET, ADJ, N), which specifies the constituents of a noun phrase as determiner, adjective and noun, corresponds to the context–free rules which result from the permutation of the three constituents on the right side). *Linear Precedence rules (LP–rules)* restrict the set of permutations (e.g., DET < ADJ produces only the context–free rules NP → DET ADJ N, NP → DET N ADJ and NP → N DET ADJ). This idea can be used for TAGs as well but with a slight modification. For CFGs the LP–rules are defined globally on the base of the nonterminals of the grammar. In the TAG formalism the unique node numbers are used to express the constraints on the elementary trees locally: Trees are interpreted as mobiles. In the name *LD/LP–UTAG* LD stands for Local Dominance (trees as domains of locality).

Lexicalized TAGs

The motivation for the use of the last extension of the formalism is the assumption that the process of syntactic generation is lexically guided (see [Kempen & Hoenkamp 87]): The adequate strategy for the verbalization of a preverbal message seems to be lexicalization, i.e. the choice of lemmas which are suitable to represent some elements of

the message followed by the construction of syntactic structures which is controlled by syntactic constraints introduced by the lemmas.

The principle of lexical guidance can best be supported by the grammar if it is lexicalized, that means that each rule has an anchor in the lexicon. The constraint for trees of *Lexicalized TAGs* (cf. [Schabes et al. 88]) is that each structure must be associated with at least one lemma which has to be the lexical head of the represented phrase (e.g., the noun of a noun phrase). The other way round, each input lemma from the Text Design component can directly be associated with a tree of the Lexicalized TAG.

4.2 Adequacy of the Formalism for Incremental Generation

The extended domain of locality within the basic TAG formalism (in contrast to CFGs) allows to express many linguistic phenomena by single trees. This property remains valid in all of the presented extensions of the formalism. TAGs with Unification allow a detailed declarative description of all kinds of syntactic phenomena (e.g., the formulation of agreement between subject and predicate of a sentence). However, it must be mentioned that the use of unification results in a formalism which is no longer context–sensitive and probably too powerful for the representation of natural language. The LD/LP-extension that divides the grammar rules into a set of mobiles and a set of order restricting constraints allows a more compact description of syntactic structures and their possible linearizations. Finally lexicalization demands that TAG trees contain at least (and for practical purpose often just) their lexical head as a leaf so that the focus lies on the description of all obligatory parts required for this lexical entry, i.e. the subcategorization frame.

The adequacy of Lexicalized LD/LP-UTAGs for sentence generation in the WIP system must be examined with respect to the incremental mode of processing and the incorporation of the generation parameters.

The consequence of incremental processing through the components of the text generator is that the input to the Text Realization component – the lemmas – are given step by step, as soon as the corresponding concepts are chosen by the Presentation Planner and the lexical choice inside the Text Design component is done. To gain best efficiency these input elements should be processed by the Realizer as soon as possible. This presupposes that with each given input lemma a syntactic rule (a TAG tree) can be chosen at once. The lexicalization property of our formalism allows such a one to one correspondence between lexical items and syntactic structures whose respective heads they are. Remember the example of the conceptual addition of a modifier to a noun in Section 3. As soon as the noun "man" is given to the Text Realization component it can be mapped to a substitution tree with root *NP* and sole son *N*. This tree describes the complete phrase opened by the noun at this moment. The modifier "big" can be mapped to an auxiliary tree with leaf *ADJ* and root and foot node *NP* because it modifies a noun phrase.

With the given example it should be intuitively clear that each lemma can be associated with an appropriate lexicalized structure. The combination of these structures depends on the defined relations between the lemmas (which are also given as input) and can be

handled by applying the grammar internal operations adjoining and substitution. In our example the auxiliary tree representing the modifier can be adjoined into the NP node of the substitution tree.

Notice, that a given lexical item can be associated with a whole set of alternative syntactic representations instead of exactly one lexicalized tree. Another possible realization of the modifier "big" could be a relative clause with the verb "to be" ("which is big"). A specific choice on such sets of rules is possible by using the given parameter settings, so the size of knowledge base entities seems adequate again.

5 The Architecture of the Syntax Generation Component

The integration of the syntax generation component into the WIP system imposes some specific requirements upon its architecture. They are described in the next section as well as some more general software engineering demands. The effects of an incremental processing mode on the architecture are shown in Section 5.2.

5.1 Requirements and Design Criteria for our Generator

To produce interrelated multimodal output – the overall task of the WIP system – its two mode specific generators must influence each other. They must be able to communicate. For this purpose the Document History Handler serves as a shared medium. One example for *crossmodal references* between the different modes is the generation of "middle knob" versus "left knob" for the same object in the knowledge representation according to its different graphical realizations, where the same knob can be seen either between two other knobs or – with another perspective in an enlargement of a part of the picture – as the leftmost of two knobs (cf. [Wahlster et al. 91]).

Another relation to surrounding components of the WIP system is indicated by the *feedback* arrow from the Text Realization component back to the Text Design component shown in the architecture in Figure 1. The syntax generator must be able to signal back a fail during its processing which can lead to revision processes in higher components.

By means of the incoming parameter settings the WIP system should be able to trigger the production of different utterances. This results in a *flexible* output medium which can be used to produce appropriate presentations for different users and which can be easily expanded by further parameter values, e.g., new categories of users.

The use of the WIP system in its online presentation mode where illustrated documents are shown on the screen – in contrast to the production of a whole document in batch mode – demands that the response time of the system should be acceptable for the user. The response time can be shortened and better *efficiency* can be gained by *incremental processing* which itself can be supported by parallelism (see Section 3). This demands of

the generator to process its input in a piecemeal fashion and to be able to integrate later specified input into already produced partial processing results.

In the following we describe some more general aspects which are important for the architecture of our generator.

By identifying different processes within the overall task of natural language generation they can be treated in distinct modules of the system (see Figure 4) leading to a *modular design*. They differ with respect to the used knowledge and/or methods. This separation makes it easier to test and exchange single components. Examples for tasks which can be isolated are the inflection of lexemes, linearization or the selection of grammar trees.

A strict separation of knowledge sources and operations, i.e. a declarative formulation of the knowledge sources, supports the transparency of the descriptions and the *extendability* and *adaptability* of the system. For the grammar and the lexica this is very common. In addition to that there are further knowledge sources (see, e.g., the Meta–Rule box described in Section 5.2), which are declaratively specified. Another advantage of this technique is that the operations can be developed without a detailed consideration of the actual content of the knowledge sources.

Of course, the produced utterances should obey the grammar rules of the target language, resulting in *grammatically correct* surface sentences.

5.2 The Individual Components

The individual components of the generator and their relations are shown in Figure 4.

The left side of the architecture shows the connections of the generator to other components of the WIP system. In the following they are shortly described together with the modules of the syntax generator (on the right side of the picture) which use them. For more details see [Harbusch et al. 91].

The goal of the utterance contains content words and their functional relations. It is passed stepwise to the *Interface* which selects the corresponding elementary trees for the lemmas. This selection varies according to the setting of the parameters. For example, the parameters which concern resources like space and execution time of the utterance can lead to the selection of "less expensive" trees (e.g., if a modifier of a noun shall be verbalized, an adjective phrase is preferred to a relative clause). Combinations of parameters must be explicitly controlled because they can contradict or strengthen each other. The selection relative to the parameter setting is done by declarative rules in the left Meta–Rule box.

An object in a *distributed parallel model* is created for each input element in order to process the chosen tree set. Trees in a lexicalized TAG have an adequate size which allows the objects to manage exactly the local syntactic and semantic information of the incoming concepts. The task of verbalization is distributed among several active objects which try to work as independent as possible. This introduces "fine–grained" parallelism into our generator in addition to the "coarse–grained" parallelism resulting from the cascaded architecture of the text generator as a whole.

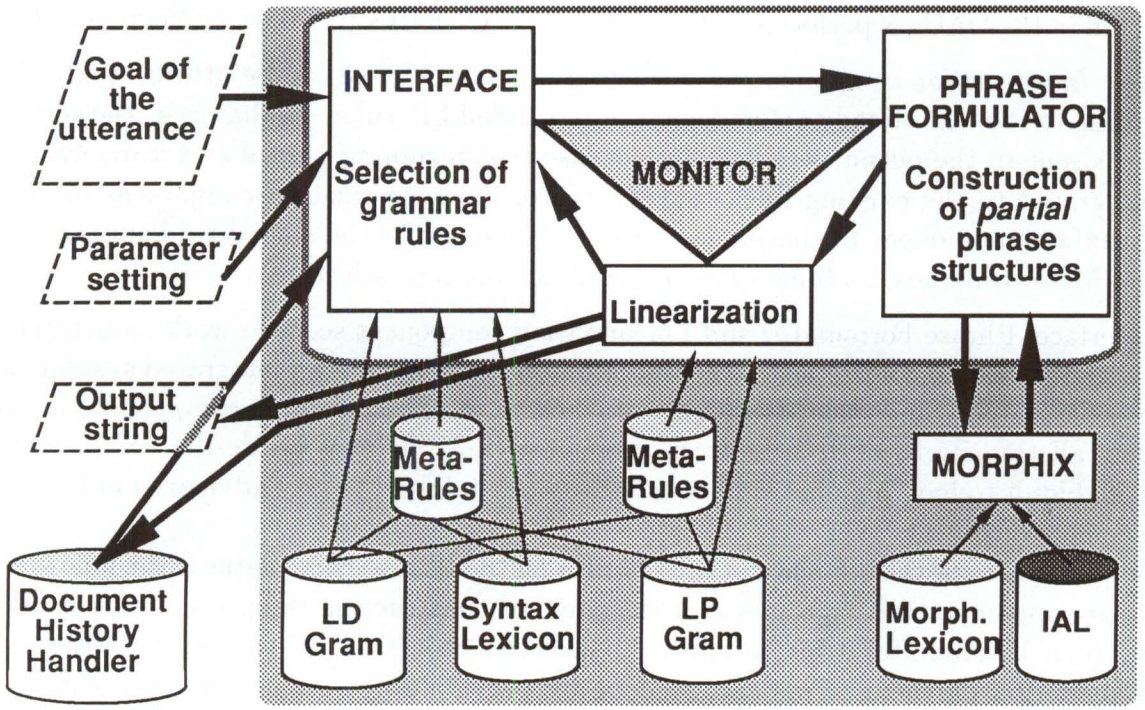

Figure 4: The Architecture of the Syntax Generator

In the *Phrase Formulator* the objects attempt to combine their locally represented trees with those of other objects according to the specified conceptual information using the TAG operations substitution and adjunction. According to the separation of the grammar into an LD- and an LP-part the combination operations are performed without regarding linear precedence constraints.

Since a distributed parallel model is used and the objects have to communicate and exchange data during these attempts, it follows that the overall task of verbalization cannot optimally be parallelized. Optimal parallelism would mean that the objects could run totally independent of each other. However, partial phrase structures can be simultaneously assembled at different nodes. To avoid inconsistency the following restriction is imposed on the objects: They must not participate in more than one communication at a time. Each object controls an own history of its represented structures which is expanded after a successful communication. This history will be used if a combination of structures must be retracted (e.g., in the case of syntactic dead-ends which are likely to occur during incremental generation).

The fact that objects represent sets of trees enables the objects to create several alternative partial structures as paraphrases at the same time. The decision between the alternatives must be made when an object hands over some structure to the Linearization component.

The inflection of the lemmas according to morphosyntactic information which has been collected in the DAGs is performed by the package MORPHIX ([Finkler & Neumann 88]).

The *Linearization* component tests whether a partial phrase can be uttered next – i.e. according to the text already uttered and the associated LP–rules. In that case, the surface string is sent to the output window. In the case that incoming partial structures cannot be integrated to the existing linearized structure, the Linearization component informs the Interface component to choose new phrase structures for the embedded lemmas. The second Meta–Rule box contains rules which direct the new selection.

Interface, Phrase Formulator and Linearization component seem to work sequentially. But with the central control instance – the *Monitor* component – an integrated system has been constructed. Since the Monitor is attached to each of the three components it can identify all transitions of data between them as well as changes inside the components. This enables it to have a global view of the states of affair and to give advices to individual components or objects.

One example for the influence of the Monitor on the Phrase Formulator is the following: If the parameter setting expresses that the time for constructing the presentation is very short, then the monitor tries to speed up isolated objects which are still waiting for communication partners. This can be done by providing default information, e.g., the case "nominative" is set for a noun phrase whose functional relation has not yet been specified. Another possibility is to make the Interface create an object which represents a dummy lemma of a certain category. Assuming that a verb "to burn" and a modifier "huge" have been associated with objects and cannot combine because a noun is missing, the Monitor can force the creation of an object for the default noun "thing" which results in the alarming utterance: "A huge thing burns."

6 Conclusions

Lexicalized LD/LP-TAGs with Unification provide a set of adequate linguistic units as elementary structures. The trees can be chosen and processed incrementally and are combined to a whole syntactic tree, whose parts are linearized and uttered as soon as possible. Thereby an incremental mode is realized within input, processing and output of the Text Realization component. The textual output of an AI system can be speeded up by using an incremental style of processing.

We have implemented a first prototype (see [Schauder 90]) separated from other components of the WIP system. The input is simulated and there is just one fixed combination of parameters (target language German, no time or space restrictions, free style). All central components of the generator – Interface, Phrase Formulator, Linearization component and Monitor – are basically implemented, but with restricted features. A grammar is prototypically realized which covers a small part of German. The inflection component MORPHIX is embedded as a complete module.

Our experience with the prototype is very promising. The separation of generation tasks into the presented components has shown to be an adequate solution. The definition of active objects in a distributed parallel model has turned out to be realizable and will be used more intensively within the system.

Acknowledgements

We would like to thank Tilman Becker and Christoph Kilger for their valuable suggestions.

References

[Buschauer et al. 89] B. Buschauer, P. Poller, A. Schauder, K. Harbusch: *Parsing natürlicher Sprache: Tree Adjoining Grammars mit Unifikation*, "AI-Laboratory" Memo in press, Department of Computer Science, University of the Saarland, Saarbrücken, 1989

[Finkler & Neumann 88] W. Finkler, G. Neumann: *MORPHIX: A Fast Realization of a Classification-Based Approach to Morphology*, Proceedings of the fourth ÖGAI, Wiener Workshop Wissensbasierte Sprachverarbeitung, Wien, 1988

[Finkler & Neumann 89] W. Finkler, G. Neumann: *POPEL–HOW: A Distributed Parallel Model for Incremental Natural Language Production with Feedback*, Proceedings of the 11th IJCAI, Detroit, 1989

[Harbusch 90] K. Harbusch: *Constraining Tree Adjoining Grammars by Unification*, Proceedings of the 13th International Conference on Computational Linguistics (COLING'90), Helsinki, 1990

[Harbusch et al. 91] K. Harbusch, W. Finkler, A. Schauder, T. Becker, B. Buschauer, P. Poller: *Generation with Tree Adjoining Grammar in the WIP project*, forthcoming, German Research Center for Artificial Intelligence, Saarbrücken, 1991

[Joshi et al. 75] A. Joshi, L. Levy, M. Takahashi: *Tree Adjunct Grammars*, Journal of the Computer and Systems Science, Volume 10, No 1, 136-163, 1975

[Joshi 85] A. Joshi: *An Introduction to Tree Adjoining Grammars*, Technical Report MS-CIS-86-64, LINC-LAB-31, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, 1985

[Kempen 78] G. Kempen: Sentence construction by a psychologically plausible formulator, in R. Campbell, P. Smith (eds.): *Recent advances in the psychology of language*, Plenum Press, New York, 1978

[Kempen & Hoenkamp 87] G. Kempen, E. Hoenkamp: *An incremental procedural grammar for sentence formulation*, Cognitive Science, Volume 11, 201-258, 1987

[Kroch & Joshi 85] A. Kroch, A. Joshi: *Linguistic Relevance of Tree Adjoining Grammars*, Technical Report MS-CIS-85-16, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, 1985

[Levelt 89] W. Levelt: *Speaking: From Intention to Articulation*, The MIT Press, Cambridge, Massachusetts, 1989

[Pullum 84] G. Pullum: *On two Recent Attempts to show that English is not a CFL*, Computational Linguistics, Volume 10, No 4, 1984

[Schabes et al. 88] Y. Schabes, A. Abeillé, A. Joshi: *Parsing Strategies with Lexicalized Grammars: Application to Tree Adjoining Grammar*, Proceedings of the 12th International Conference on Computational Linguistics (COLING'88), Budapest, 1988

[Schauder 90] A. Schauder: *Inkrementelle syntaktische Generierung natürlicher Sprache mit Tree Adjoining Grammars*, Master's Thesis, Department of Computer Science, University of the Saarland, Saarbrücken, 1990

[Shieber 86] S. Shieber: *An Introduction to Unification-based Approaches to Grammar*, CSLI Lecture Notes, No. 4, Stanford University, Stanford, California, 1986

[Wahlster et al. 91] W. Wahlster, E. André, W. Graf, T. Rist: *Designing Illustrated Texts: How Language Production is influenced by Graphics Generation*, Fifth Conference of the European Chapter of the ACL, Berlin, 1991

[Weir 88] D. Weir: *Characterizing mildly context-sensitive grammar formalisms*, PhD Thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, 1988

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-01
Franz Baader: Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert: A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel, Nelson M. Mattos: Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder, Werner Nutt: Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader: A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder: Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist: Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel: A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo: Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann
Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung
114 Seiten

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka :
On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt:
The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost
X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
Wolfgang Wahlster, Elisabeth André, Winfried Graf, Thomas Rist: Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation.
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents
A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist
WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta
RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke
A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel
Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-12
J.Mark Gawron, John Nerbonne, and Stanley Peters
The Absorption Principle and E-Type Anaphora
33 pages

RR-91-13
Gert Smolka
Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-15
Bernhard Nebel, Gert Smolka
Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann
Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

RR-91-17
Andreas Dengel & Nelson M. Mattos
The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner
FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-23
Prof. Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
Akquisition und Repräsentation von technischem Wissen für Planungsaufgaben im Bereich der Fertigungstechnik
24 Seiten

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder
Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

DFKI Technical Memos

TM-89-01
Susan Holbach-Weber: Connectionist Models and Figurative Speech
27 pages

TM-90-01
Som Bandyopadhyay: Towards an Understanding of Coherence in Multimodal Discourse
18 pages

TM-90-02
Jay C. Weber: The Myth of Domain-Independent Persistence
18 pages

TM-90-03
Franz Baader, Bernhard Hollunder: KRIS: Knowledge Representation and Inference System -System Description-
15 pages

TM-90-04
Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich: Terminological Knowledge Representation: A Proposal for a Terminological Logic
7 pages

TM-91-01
Jana Köhler
Approaches to the Reuse of Plan Schemata in Planning Formalisms
52 pages

TM-91-02
Knut Hinkelmann
Bidirectional Reasoning of Horn Clause Programs: Transformation and Compilation
20 pages

TM-91-03
Otto Kühn, Marc Linster, Gabriele Schmidt
Clamping, COKAM, KADS, and OMOS: The Construction and Operationalization of a KADS Conceptual Model
20 pages

TM-91-04
Harold Boley
A sampler of Relational/Functional Definitions
12 pages

TM-91-05
Jay C. Weber, Andreas Dengel and Rainer Bleisinger
Theoretical Consideration of Goal Recognition Aspects for Understanding Information in Business Letters
10 pages

DFKI Documents

D-89-01
Michael H. Malburg, Rainer Bleisinger: HYPERBIS: ein betriebliches Hypermedia-Informationssystem
43 Seiten

D-90-01
DFKI Wissenschaftlich-Technischer Jahresbericht 1989
45 pages

D-90-02
Georg Seul: Logisches Programmieren mit Feature -Typen
107 Seiten

D-90-03
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Abschlußbericht des Arbeitspaketes PROD
36 Seiten

D-90-04
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: STEP: Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch
69 Seiten

D-90-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Formalismus zur Repräsentation von Geo-metrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells
66 Seiten

D-90-06
Andreas Becker: The Window Tool Kit
66 Seiten

D-91-01
Werner Stein , Michael Sintek
Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause
RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp
Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause
Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 pages

D-91-09
David Powers and Lary Reeker (Eds)
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.)
MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
Thilo C. Horstmann
Distributed Truth Maintenance
61 pages

Incremental Syntax Generation with Tree Adjoining Grammars

Karin Harbusch, Wolfgang Finkler, Anne Schauder

RR-91-25
Research Report