Grammars and Discourse Theory to Describe and Recognize Mechanical Assemblies

Christian Bauckhage, Susanne Kronenberg,
Franz Kummert, and Gerhard Sagerer

Technische Fakultät, AG Angewandte Informatik
Bielefeld University, P.O. Box 100131, 33501 Bielefeld, Germany
{cbauckha,susanne}@techfak.uni-bielefeld.de

Abstract. In recent years numerous approaches to automatic reasoning about mechanical assemblies have been published. CAD techniques, graph-based approaches and semantic methods to model assembled objects were proposed. However, all these methods are difficult to employ for visually supervising assembly processes since they lack the flexibility, generality or ease required in practice. This paper presents the idea of using syntactic methods known from discourse theory in order to model classes of mechanical assemblies. Moreover, a criterion to derive recursive grammatical production rules is introduced so that the representation of a class of assemblies becomes especially compact. This general representation scheme allows to automatically derive hierarchical structural descriptions of individual assemblies by means of computer vision.

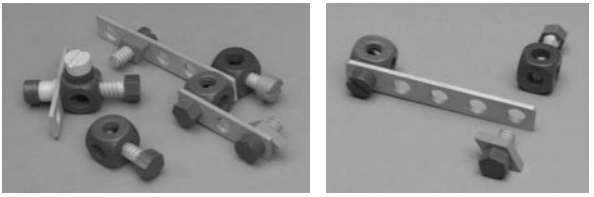
1 Introduction

The work presented in this paper is embedded in a research project studying advanced human-machine communication [4]. The project's purpose is to develop a robot which is able to process visual and acoustic data in order to recognize and manipulate arbitrarily positioned objects in its environment according to the instructions of a human.The domain was chosen to be the cooperative construction of a toy-airplane using parts of a wooden construction-kit for children (see Fig. 1). As it is customary for every mechanical construction process this scenario implies that objects from a set of separate parts are assembled to form more complex units. These units are called *mechanical assemblies* and are defined to be sets of solid parts with a distinct geometric relation to one another [8]. *Subassemblies* are subsets of parts consisting of one or more elements in which all parts are connected. Thus, in a construction process subassemblies are sequentially assembled and the resulting products typically show a hierarchical structure.

Supervising robotic assembly by means of a computer vision system requires a representation of knowledge about feasible assemblies in order to recognize complex objects in image data. Concerning flexibility, our system must not be specialized on the construction of airplanes. Thus, a compact representation is required which enables to describe any of the numerous feasible toy-assemblies.

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 173–182, 2000.

Throughout the last decade there was extensive research concerning flexible assembly. In geometry based modeling CAD techniques are used to describe elementary objects and assemblies [14]. Graph-based methods model certain relations between elementary parts [3], whereas semantic approaches make use of manually created hierarchical descriptions [2]. There also are sensor based approaches to monitor assembly processes which, however, employ highly specialized sensing tools like laser-range-finders [7].



(a) (b)

Fig. 1. Examples of feasible assemblies in the construction scenario.

All these approaches either depend on detailed geometric information or require structural knowledge about individual assemblies. Our system, however, should react to instructions in reasonable time but visually determining the geometric structure of an assembly is rather time consuming. Moreover, it is impossible to provide a detailed structural description for every imaginable assembly in our scenario. To cope with these difficulties we propose a grammar-based method to model assembled objects which is described in the following.

2 Assembly Modeling by Discourse Grammars

2.1 Motivation

Syntactical approaches to pattern recognition problems are used for many years. They allow to classify complex patterns of interrelated primitives and yield a structural description simultaneously. Furthermore, the possibility to define recursive production rules generally leads to particular compact representations of classes of patterns [5].

Grammatical methods have also been introduced to computer aided manufacturing applications like workpiece or machine tool design [1]. However, to the best of our knowledge grammar-based approaches have not yet been used to qualify the internal structure of mechanical assemblies.

Our efforts towards this end start from the following consideration: putting mechanical parts together means to connect them via some mechanical features. These *mating features* are well known in assembly modeling where their geometrical aspects are of primary concern [14]. An examination of their functional aspects reveals, however, that mating features also allow to specify recursive production rules: from a mating feature point of view an assembly consists of several *functional units* each providing features necessary to realize stable connections. Similarly, an assembly is composed of several subassemblies which –as pointed

out in the introduction– may be elementary parts or assembled objects. Thus, from a functional viewpoint elementary parts and assemblies can be treated equally: both must provide mating features to be assembled.

This observation immediately leads to a recursive method to describe mechanical assemblies. Consider for example the bolt-nut type assemblies depicted in Fig. 1. Objects like rings, bars or sockets (called miscellaneous objects in the following) can be put onto bolts and are fastened using nut-type objects like cubes or rhomb-nuts. As Fig. 1(b) indicates an assembly consists at least of a bolt and a nut and optionally of some miscellaneous objects and may serve as a bolt, a miscellaneous object or a nut itself. Thus, understanding assemblies to be composed of a *bolt-part*, an optional *miscellaneous-part* and a *nut-part* results in the following grammar for bolted assemblies:

$$
\begin{aligned}
\text{ASSEMBLY} &\rightarrow \text{BOLT\_PART}\ \ \text{NUT\_PART} \mid \\
&\quad\ \ \text{BOLT\_PART}\ \ \text{MISC\_PART}\ \ \text{NUT\_PART} && (1)\\
\text{BOLT\_PART} &\rightarrow \text{ASSEMBLY} \mid \mathit{BOLT} && (2)\\
\text{NUT\_PART} &\rightarrow \text{ASSEMBLY} \mid \mathit{CUBE} \mid \mathit{RHOMBNUT} && (3)\\
\text{MISC\_PART} &\rightarrow \text{ASSEMBLY} \mid \mathit{BAR} \mid \mathit{FELLY} \mid \mathit{RING} \mid \mathit{SOCKET} \mid \\
&\quad\ \ \text{ASSEMBLY}\ \ \text{MISC\_PART} \mid \mathit{BAR}\ \ \text{MISC\_PART} \mid \mathit{RING}\ \ \text{MISC\_PART} \mid \\
&\quad\ \ \mathit{FELLY}\ \ \text{MISC\_PART} \mid \mathit{SOCKET}\ \ \text{MISC\_PART} && (4)
\end{aligned}
$$

Production rule (1) describes the composition of a bolted assembly and reflects the fact that the miscellaneous-part is optional. The second and third rule state that the bolt- and the nut-part consist either of a (sub)assembly or a corresponding elementary object. Rule (4) describes how the miscellaneous-part of an assembly is constructed: it is either given by a single part or a subassembly or a possibly infinite sequence of those. In reality, of course, the number of miscellaneous objects depends on the length of the corresponding bolt. This fact is neglected here but is captured by a unification process (see section 2.3).

In fact, this grammar produces linear structures and is rather simple. Even though sophisticated techniques like graph grammars seem more appropriate we will show in the following that this simple grammar is nevertheless sufficient to describe 3D objects resulting from a construction process. Moreover, grammatical assembly models are not restricted to cases with one class of mating features or to binary mating relations. Figure 2 shows another (rather didactic) assembly domain to illustrate how the principles deduced for bolted assemblies can be generalized to n-ary mating relations and several types of connections. A hammer as depicted in Fig. 2(c) consists of subassemblies which are connected via different mating features. To construct the handle a ternary mating relation comprising a chock-part a ring-part and a shaft-part has to be established while the head of the tool is connected to the handle via a snap fit mechanism.

2.2 Discourse Grammars - Assembly Grammars

As mentioned above this work is associated with research in man-machine communication where speech processing is of especial importance. It were these fortunate circumstance that revealed a similarity between discourse and assembly structures. In this section some common known facts of discourse analysis

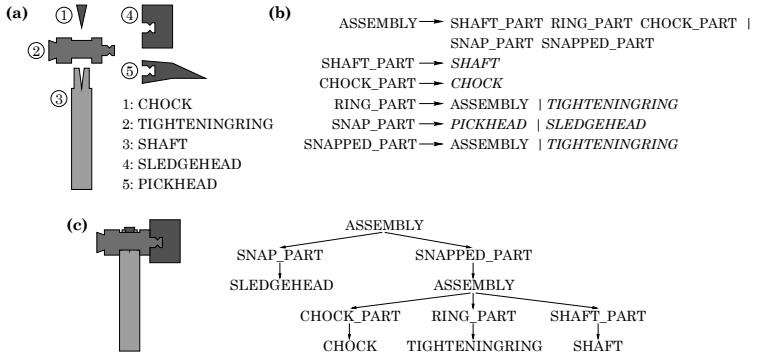


Fig. 2. Another assembly domain.

are outlined to motivate that the underlying principles for processing discourse can be directly transformed to a computational framework for reasoning about assemblies. Intuitively, discourse grammars specify how structural units are combined due to some tools for investigating discourse structure and discourse relations [13]. It is generally agreed that discourse has a recursive structure i.e. a discourse structure is recursively built based on discourse relations which can be stated between smaller discourse segments [6]. Therefore, a discourse grammar consists of a sentence grammar, mostly chosen to be unification based, and a set of (binary) discourse grammar rules, i.e. a set of rules describing discourse relations. This defines a framework in which both intra- and intersentential constraints can be expressed, i.e. a framework which integrates discourse constraints together with syntactic and semantic constraints. Moreover, often the processed discourse structure is only partially available for further processing. This means that only subparts with special properties are labeled as admissible parts for subsequent interpretation [15]. Summarizing, discourse grammars generate a structure based on semantic and syntactic knowledge which captures various discourse phenomena [9].

To show that discourse theory can be transformed into a framework for the description of the internal structure of assemblies the structural properties of assemblies must meet the requirements of discourse. That is, assemblies must have a recursive structure in that bigger units can be obtained by recursive sequencing and embedding of smaller ones. Additionally, this recursion must be defined by a relation which holds between subassemblies. As outlined in section 1 recursion is naturally given for mechanical assemblies due to the properties of the construction process. Furthermore, a relation can be stated between subassemblies which is given by common elementary objects: subassemblies can be conjoined to yield larger units iff these subassemblies share common elementary objects.

2.3 The Processing Model

The approach for recognizing assembled objects is based on a LR(1)-parser and a unification grammar [11]. The advantage of this approach is that the unification grammar puts most of the syntactic information that is standardly captured in

context free phrase structure rules into the lexicon. Every word is represented in the lexicon by a feature structure which specifies the values for various attributes. In the assembly approach the lexicon contains all elementary objects and the feature structure for each object specifies values for various attributes describing the structural properties of this object. A LR(1)-parser standardly uses two actions: shift and reduce. The reduce action reduces the right hand symbols of a grammar rule to the left hand symbol. Our model is based on a LR(1)-parser which is augmented with the facility to handle feature structures [10]. This implies that every reduce action includes a unification of the corresponding feature structures yielding structural descriptions of subassemblies. The shift action pushes the input symbols from the input string onto the processing stack. In order to 'parse' three dimensional objects an order to shift recognized objects of a given scene onto the processing stack must be defined, i.e. a multi dimensional signal must be transferred into a linear sequence of primitives. Generally, this order depends on the domain but as a rough estimate it is always stated on necessary elementary objects. These are objects that must be found in every subassembly (e.g. bolts in bolted assemblies) and constrain the range within the signal where parsing has to proceed next (for a detailed description of our linearization heuristic see section 3).

The basic idea of our processing model is to try to find small units in the input signal (similar to discourse segments in speech parsing) and to conjoin them with earlier found ones, if possible. This means: based on the shift and reduce actions of the LR(1)-parser the process of parsing subparts of the signal yields derivation trees (or structural descriptions) of subassemblies which are conjoined with already derived descriptions, if necessary. Subsignal parsing terminates if the signal is processed completely or if the unification fails. A unification error occurs if two objects should be assembled due to the LR(1)-parser grammar but the subcategorization frame of the corresponding feature structure of one of these objects is already satisfied (e.g. a bolt and a cube will not be combined if the cube cannot absorb another bolt since all its holes are used already). If parsing a subpart of the signal terminates two different cases can occur:

1. No other subassembly was recognized so far. Therefore, the parsing process restarts at another part of the signal, i.e. with another necessary elementary object, if existing, otherwise the complete assembly is recognized.
2. Other subassemblies were already recognized. Therefore, it is tested if the subassemblies recognized earlier can be conjoined with the recent one to form a larger assembly. After that, the parsing process is continued at another necessary elementary object, if existing.

In order to merge subassemblies a *conjoin* relation is stated between derivation trees (also denoted *assembly trees*) to integrate already derived trees (which are called *current assembly trees* according to discourse theory terminology) into *incoming assembly trees* (which is another denotation for the recently derived assembly tree).

Definition: The *conjoin* relation $\triangle$ combines two assembly trees X and Y if both share an elementary object. The assembly tree X is integrated into Y at

node n which is labeled with the common elementary object and which occurs at the frontier of the incoming assembly tree Y. This means in Y$\triangle$X the node n of Y is substituted by the assembly tree X (see Fig. 3(b), 3(c) and 3(d)).

In our processing model this relation is realized by an additional LR(1)-parser action, so-called *conjoin*. If an assembly tree is derived the conjoin action first searches for common elementary objects in the current assembly trees and then –if common elementary objects are found– conjoins both assembly trees. Since the head node of every subassembly tree is labeled with the elementary objects contained in the highest hierarchical level of the subassembly only nodes along the path given by the head node have to be searched for common objects. This conjoin action is performed on every elementary object of the incoming assembly tree and for all current assembly trees [1] Three different cases can occur:

1. No common elementary object is found in the current assembly trees, i.e. only subassemblies were recognized so far which are not connected to the incoming one. (see Fig. 3(a) and 3(b)).
2. The conjoin relation can be stated between a current assembly tree and the incoming assembly tree. Therefore, the current assembly tree will be integrated into the incoming one (see Fig. 3(d)).
3. More than one common elementary object can be found in a current assembly tree. Consequently, every node labeled with an elementary object in the incoming assembly tree will be substituted by the current assembly tree. Because the same current assembly tree is integrated several times into the incoming one, all occurrences of the current assembly tree in the incoming one are identical [2]. This corresponds to cases where assemblies are connected via several elementary objects. Therefore, all occurrences of a current assembly tree in an incoming assembly tree are unified to one singleton subtree. An example of this case is shown in Fig. 5(g).

3 Recognition

In this section we illustrate the processing model for the visual recognition of assembled objects by means of the example depicted in Fig. 5(a) (note that the object labels were added manually to facilitate the discussion).

As a syntactic method the assembly recognition procedure depends on primitives which in our case are elementary objects. Following a hybrid approach that combines neural and semantic nets Kummert et al. [12] realized a fast and reliable method to recognize such primitives yielding labeled image regions as shown in Fig. 5(b). Since assembled objects are physically connected they appear as clusters of adjacent regions once the recognition of single objects is completed. In order to recognize assemblies by parsing clusters of regions their elements have to be examined in a sequential manner. Generally, a method to

[1] After a conjoin action is performed the comparison process is continued on the conjoined assembly tree, for than the new incoming tree, and the remaining current trees, including the just conjoined current tree, in case that it contains several common elementary objects.

[2] This can be easily verified by the head nodes of each integrated current assembly tree, because they all are labeled with the same elementary objects.

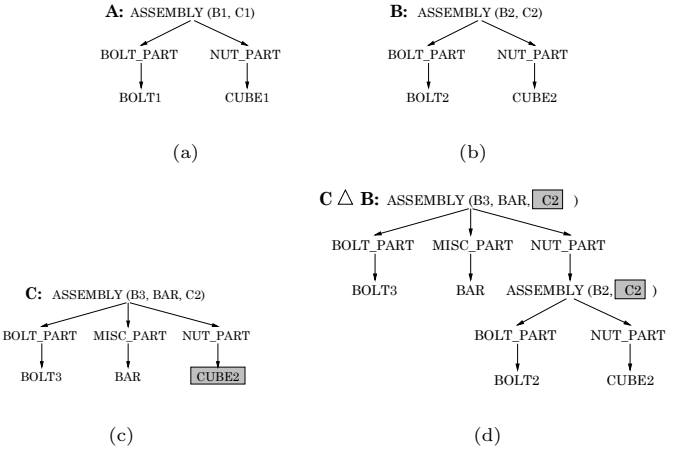


Fig. 3. 3(a), 3(b) and 3(c) Assembly trees of singleton subassemblies. 3(d) Subassemblies C and B share object CUBE2 thus the corresponding trees are conjoined. All structures describe subassemblies of the assembly shown in Fig. 5(a).

order two-dimensionally distributed primitives is domain dependent and has to be found heuristically. In our application, simple topological considerations lead to a straightforward linearization technique: objects that are put onto bolts are linearly arranged in space, thus the centers of mass of regions representing single objects are ordered approximately linear (see the crosses in Fig. 5(b)). Therefore, starting parsing at regions representing bolts and then choosing an adjacent region provides a search direction. If there are several adjacent regions different alternatives for parsing must of course be considered. After examining a certain region in the parsing process the next object to be examined must be adjacent to the recent one and if there are several candidates the nearest one complying with the search direction is chosen. If an adjacent object depicts a bar this heuristic has to be modified since the positions of the holes have to be considered instead of the center of mass. Obviously, holes that have been used in construction are not visible, however, virtual positions as emphasized in Fig. 5(b) can be estimated by dividing the bar's main axis into equidistant ranges.

Figures 5(c)-5(f) show the intermediate results and the final outcome of the assembly recognition procedure for the example. After starting the process at BOLT1 the next object to be examined due to the heuristic is CUBE1. According to the rules of the grammar both objects form an assembly (see the corresponding assembly tree of subassembly A in Fig. 3(a)) thus a white polygon comprising both labeled regions is output (Fig. 5(c)). Then, another necessary elementary object, i.e. a bolt (here: BOLT2), is chosen as the starting point for a parsing process leading to the description of subassembly B in Fig. 3(b) and the recognition result depicted in Fig. 5(d).

Continuing at BOLT3 yields assembly tree C in Fig. 3(c) which shares an elementary element with subassembly B. Consequently, both descriptions are unified leading to the description C△B in Fig. 3(d) and the result depicted in

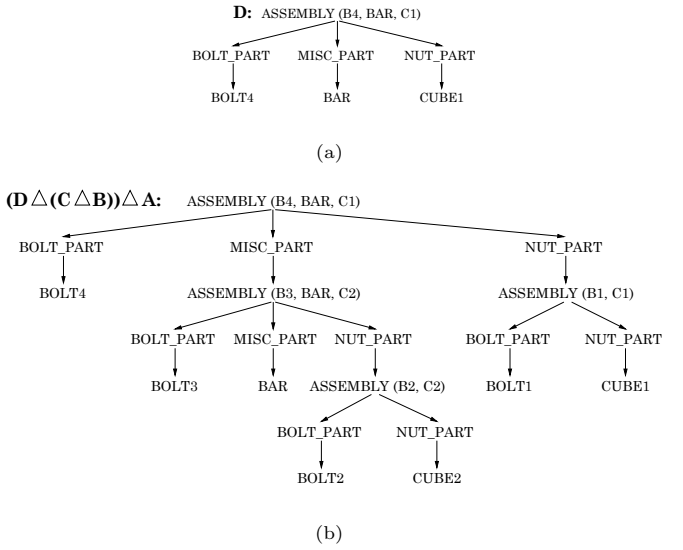


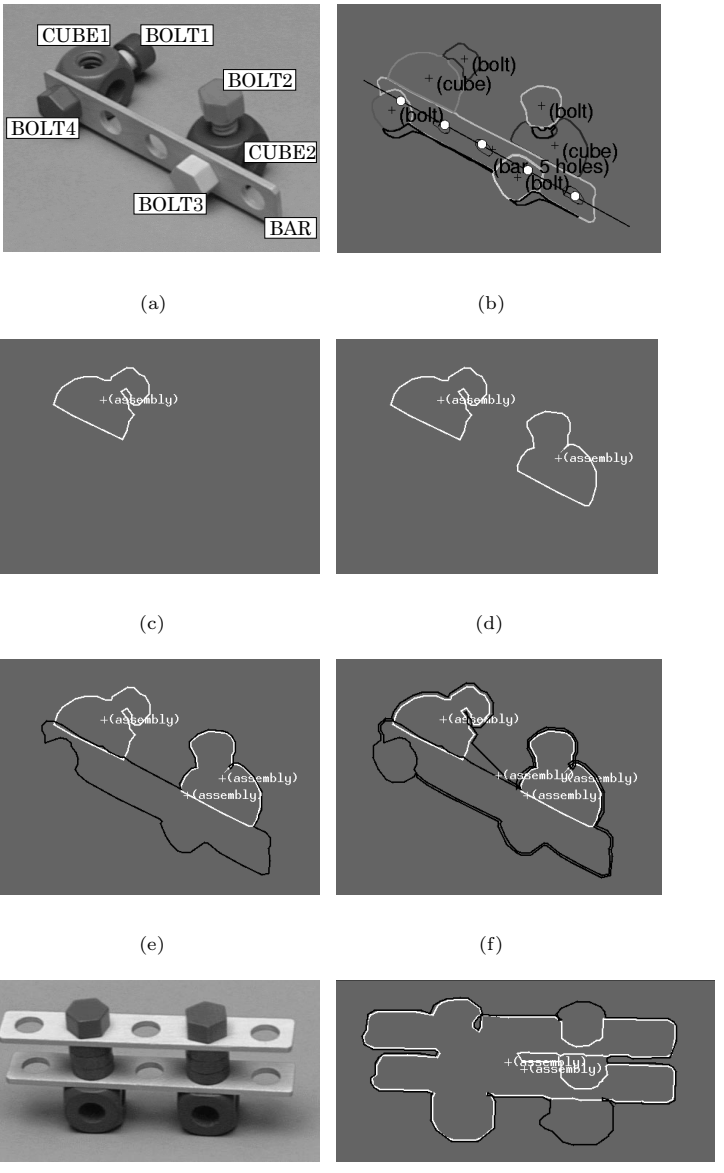
Fig. 4. Assembly trees describing a singleton subassembly and the whole assembly depicted in Fig. 5(a).

Fig. 5(e). Note that black polygons surround the objects of recently found *complex* subassemblies. Finally, continuing the parsing process at BOLT4 detects subassembly D shown in Fig. 4(a). Since D shares elementary objects with subassemblies A and C△B the corresponding conjoin operation leads to the assembly tree (D△(C△B))△A describing the whole assembly and a polygon comprising all the subassemblies is generated (see Fig. 5(f)).

4 Conclusion

Structurally analyzing mechanical assemblies typically yields a component hierarchy where higher-level components consist of lower-level parts which again may have internal structure. This paper presented the idea to use grammars in order to describe the hierarchical structure of individual assemblies. Analyzing mechanical mating features revealed a functional equivalence of elementary objects and subassemblies and resulted in compact grammatical rules describing whole classes of assemblies.

Grammars together with a suitably defined lexicon comprising facts about mechanical elements allow an automatic derivation of structural descriptions of assembled objects. Employing techniques known from discourse theory we introduced a parsing model to recognize assembled objects in image data. Based on a domain dependent heuristic clusters of recognized elementary objects are analyzed sequentially and structures of detected subassemblies are conjoined if necessary. Thus, if all elementary objects comprising an assembly are visible a reliable recognition of assembled objects is possible which allows to supervise the intermediate steps of a construction process.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Fig. 5. 5(a) An assembly with 4 bolts. 5(b) Results of a single object recognition procedure and highlighted portpoints of the bar. 5(c) Recognition of a simple subassembly. 5(d) Recognition of a second simple subassembly not connected to the first one. 5(e) Recognition of a further subassembly sharing a part with the second one, thus they are conjoined. 5(f) Detection of a subassembly sharing parts with the first and third one, i.e. recognition of the whole assembly. 5(g) and 5(h) Another feasible assembly and the recognition result.

Further work concerning grammar based assembly recognition must consider the problem of perspective occlusions. I.e. the parsing model must be extended such that fuzzier input data like incorrect or missing results of the elementary object recognition procedure does not affect the correct detection of assembled objects. To this end we hope to benefit from ideas to deal with incorrect or incomplete speech input [11].

References

1. K.N. Brown, C.A. McMahon, and J.H. Sims Williams. Describing process plans as the formal semantics of a language of shape. *Artificial Intelligence in Engineering*, 10(2):153–169, 1996.
2. S. Chakrabarty and J. Wolter. A Structure-Oriented Approach to Assembly Sequence Planning. *IEEE Transactions on Robotics and Automation*, 13(1):14–29, 1997.
3. T.L. De Fazio, S.J. Rhee, and D.E. Whitney. Design-Specific Approach to Design fo Assembly (DFA) for Complex Mechanical Assemblies. *IEEE Transactions on Robotics and Automation*, 15(5):869–881, 1999.
4. G.A. Fink, N. Jungclaus, F. Kummert, H. Ritter, and G. Sagerer. A Distributed System for Integrated Speech and Image Understanding. In *International Symposium on Artificial Intelligence*, pages 117–126, 1996.
5. K.S. Fu, editor. *Syntactic Pattern Recognition, Applications*. Springer-Verlag, Berlin, 1977.
6. C. Gardent. Discourse tree adjoining grammar. Technical report, University of Saarland, CLAUS report 91, 1997.
7. K. Ikeuchi and T. Suehiro. Towards an Assembly Plan from Observation Part I: Task Recognition with Polyhedral Objects . *IEEE Transactions on Robotics and Automation*, 10(3):368–385, 1994.
8. R.E Jones, R.H. Wilson, and T.L. Calton. On Constraints in Assembly Planning. *IEEE Transactions on Robotics and Automation*, 14(6):849–863, 1998.
9. S. Kronenberg and F. Kummert. Syntax coordination: Interaction of discourse and extrapositions. In *ICSLP*, volume 5, pages 2071–2074, 1998.
10. S. Kronenberg and F. Kummert. Soft unification: Towards robust parsing of spontanous speech. In *IASTED Int. Conf. on Artificial Intelligence and Soft Computing*, pages 381–385, 1999.
11. S. Kronenberg and F. Kummert. Structural dependencies between syntactic relations: A robust parsing model for extrapositions in spontaneous speech. In *ASRU: IEEE Int. Workshop on Automatic Speech Recognition and Understanding, [Online]*, 1999. Available HTTP: http://asru99.research.att.com/abstracts/4_9297.html.
12. F. Kummert, G.A. Fink, G. Sagerer, and E. Braun. Hybrid Object Recognition in Image Sequences. In *ICPR*, volume II, pages 1165–1170, 1998.
13. R. Scha and L. Polanyi. An augmented context free grammar for discourse. In *Proc. 12th Int. Conf. of the Association for Computational Linguistics*, pages 573–577, 1988.
14. W. van Holland, W.F. Bronsvoort, and F.W. Jansen. Feature Modelling for Assembly. In W. Straßer and F.M. Wahl, editors, *Graphics and Robotics*, pages 131–148. Springer-Verlag, Berlin, 1995.
15. Bonnie Lynn Webber. Structure and ostension in the interpretation of discourse deixis. *Language and Cognition Processes*, pages 107–135, 1991.