Knowledge-Based Assembly Simulation for Virtual Prototype Modeling

Bernhard Jung, Marc Latoschik, Ipke Wachsmuth

Faculty of Technology, University of Bielefeld
PO Box 100 131
D-33501 Bielefeld
{jung, marcl, ipke}@techfak.uni-bielefeld.de

Abstract -- **The idea of Virtual Prototyping is the use of realistic digital product models for design and functionality analysis in early stages of the product development cycle. The goal of our research is to make modeling of virtual prototypes more intuitive and powerful by using knowledge enhanced Virtual Reality techniques for interactive construction of virtual prototypes from 3D-visualized, CAD-based parts. To this end, a knowledge-based approach for real-time assembly simulation has been developed that draws on dynamically updated representations of part matings and assembly structure. The approach has been implemented in an experimental system, the CODY Virtual Constructor, that supports a variety of interaction modalities, such as direct manipulation, natural language, and gestures.**

I. Introduction

Virtual Prototyping aims at the construction of realistic digital product models for design and functionality analysis in early stages of the product development cycle. The main goal of virtual prototyping is to reduce the number of physical prototypes fabricated in the product design process with the benefit of shortened development cycles [1, 5].

Virtual Reality (VR) techniques have so far mostly been applied for visual evaluation of the virtual prototype. In contrast, the preceding modeling of the virtual prototype relies on „open-loop“ interaction CAD-based methods: The designer models the prototype within the CAD system and only upon completion the geometry data is exported to a separate VR system for inspection of the digital mock-up. If a construction fault is discovered, the time-consuming design-and-inspection cycle is started over again. VR interfaces, on the other hand, offer a more intuitive, „closed-loop“ human-computer-interaction: The designer can manipulate the scene objects directly in the virtual environment and receives immediate visual feedback about model changes. However, closed-loop VR interaction cannot be readily applied to modeling of virtual prototypes. The main problem is that VR representations of the virtual prototype are mainly geometric and, accordingly, system support for manipulation of scene objects is uninformed of task-level constraints. Thus, e.g. in mechanical engineering scenarios, assembly constraints like part matings cannot be preserved.

In our approach to virtual prototype modeling, we aim at combining closed-loop VR interaction techniques with expert system techniques to achieve an intelligent task-level interface that relieves designers from the burdens of complex command interfaces. To this end, a knowledge-based approach for real-time assembly simulation of CAD-based parts has been developed that draws on dynamically updated knowledge representations of part matings and assembly structure. Key factors in this approach include:

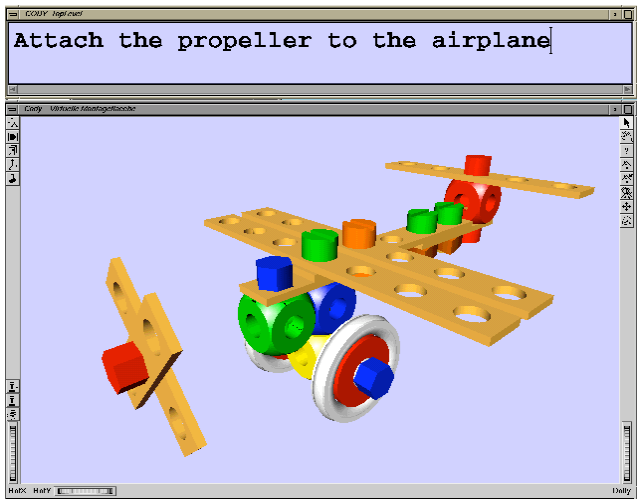


Figure 1. Interactive assembly simulation with the CODY Virtual Constructor.

- Task-level human-computer-interaction using direct manipulation, natural language, and gestures that is focused on communicating assembly operations rather than low-level geometry scene changes.
- Knowledge-based, real-time assembly simulation: Knowledge-based descriptions of the objects‘ mating possibilities enable the simulation of part assembly and disassembly as well as modification of aggregates.
- Dynamic scene conceptualization: The assembly state is step-keepingly matched against a model knowledge base of a target aggregate. Dynamic knowledge representations are created when constructed aggregates are recognized as assembly groups of the target aggregate. Due to dynamic scene conceptualization, verbal instructions can always refer to the current state of the assembly and can make use of functional names and properties.

Figure 1 shows a sample interaction with a desk-top version of our demonstrator system, the CODY Virtual Constructor.

Figure 2: The virtual assembly environment can be projected onto a wall-size display. Speech and gestures are used to communicate scene changes.

II. Human Computer Interaction

Interactivity is the most important asset to support closed-loop virtual prototyping. The user should be able to get a grasp on the model and change it as if it were real. A comfortable, task-level human-computer interface shall keep the designer free from technical considerations such as planning of geometric detail.

In the Virtual Constructor, we experiment with different kinds of visual displays (desktop to large-screen projection) and explore different interaction modalities, such as natural language, gestures, and direct manipulation in order to create easy-to-use interfaces for virtual prototype modeling. Figure 2 shows a setting where speech and gestures are used to assemble a virtual prototype of a „Citymobil“ vehicle whose components are projected onto a wall-size display.

A. Natural Language Instructions

Interaction modalities in the design and exploration of 3D computer graphics can be enhanced by „intelligent“ (symbolic) communication. We argue that interaction modalities should include language as means of communication. An important issue is that the system has to know about the spatial structure as it is perceived and experienced by the human user. When using verbal interaction, we need to be aware of the fact that the way we refer to details in a scene is „situated.“ For instance, it may depend on the objects themselves where we would speak of „front“ and „back,“ e.g., an airplane or a mobile platform impose local reference structures on space, and they may have intrinsic „front“ parts. We may use still different notions of „front“ and „back“ - and also of „left“ and „right“ - when making reference to our current aspect of view. When a deictic reference is involved („here“), the point of anchoring a reference frame depends on the current position of the speaker which could be identified with the camera position in a virtual environment. Knowledge of the current position of the observer (camera) can help to resolve ambiguity in instructions.

B. Direct Manipulation

When the virtual assembly environment is visualized on a desktop display, direct manipulation of scene objects using the mouse or space-mouse is an effective interaction means familiar to untrained users. We have developed special object manipulators supporting assembly and disassembly of the visualized scene objects using the mouse or other pointing devices. With the assembly manipulator, for example, the user move and rotate an object in all three dimensions; when brought close enough to another object, a knowledge-based „snapping“ process completes the assembly operation. Another set of manipulators enables the modification of assemblages along translational and/or rotational degrees of freedom for different types of object connections. For example, Figure 3 shows an object manipulator for assembly operations and Figure 4 an object manipulator used for changing the relative orientation of two assembled parts.

C. Gestures

When the virtual scene is visualized on large screen displays, a direct manipulation of the virtual scene is no longer appropriate. Mouse and keyboard obviously restrict the user when standing upright in front of the working area. Here, analyzing the user‘s gestural expression seems much more convenient as an input method. In our virtual construction application, electromagnetic trackers and data-gloves are used for gesture recognition. Gesture-based interaction is broken down in several tasks; more concretely, we have to:

- reference objects,
- reference locations, and
- detect desired manipulations

Objects and locations are referenced with simple pointing gestures to the object‘s virtual representation or to the desired point in space. After a selection is done, rotating and translating the hand is interpreted as an analogical manipulation of the pre-selected object, allowing the user to change the objects‘ spatial position and orientation [11].

D. Speech and Gesture Integration

For humans, the combination of speech and gestures is a very natural form of communication. Combination of speech and gestures also has advantages over gesture input alone in situations where (pointing) gestures are vague or where similar objects are arranged spatially close together. A coarse pointing gesture accompanied by a statement like „...take that red wheel...“ in most situations provides enough information to identify an unambiguous reference object. In addition, there is a temporal relationship between semantically related parts in the gesture- as well as in the speech-stream. In the above example, the climax of the pointing gesture has most likely occurred shortly before or during the time when the word „...that...“ was spoken. This information is used to strengthen hypotheses about detected gestures and identify a reference object.

E. Multi-Modal Presentation

Besides immediate visual feedback, we also found it useful to provide the user with audio feedback about the actions taken. Each action type is associated with a characteristic sound generated upon the successful completion of the action. For example, each assembly step is accompanied by a „click“ sound. If a user instruction cannot be completed, e.g. due to the user not anticipating an object collision resulting from an assembly operation, a special „bong“ sound is generated indicating failure. In this way, the user is always informed about the success or failure of intended assembly steps.

Related Work: Special aspects of natural language processing for instruction of virtual environments have been investigated in [2] and [4]. An overview of current research in gesture-based interfaces can be found in [17]. The combination of pointing gestures and speech input was realized first in the *Put That There System* [3]. In current work, we are extending combined speech/gesture input to include further natural gestures besides pointing.

III. Knowledge-Based Assembly Simulation

To enable an interactive assembly simulation in the virtual environment, internal object models of assembly objects must not restrict to data structures describing object geometry, such as their shape and location, but should also include information about their functional properties and about how they interact with other objects in the environment. Further, at least part of physics should be in effect, for instance, it should not be possible for objects to pass through one another, or, to move an object further than permitted by a physical boundary.

Central to our approach to assembly simulation is the notion of connection *ports*. A port represents a place within an object, that allows the object to mate with other objects. Thus, ports represent the primary object functionality in assembly, namely the ability to be connected with other objects. We have defined several kinds of ports that enable different kinds of part matings inducing, among other characteristics, different transformational degrees of freedom between the connected parts. Port types are modeled in a knowledge

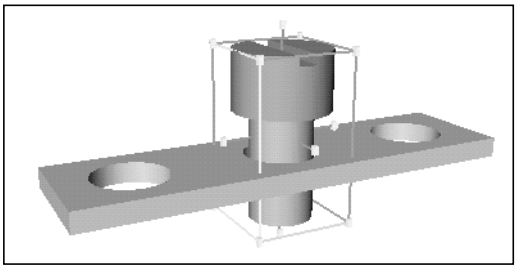


Figure 3: Peg-in-hole-like connections can be modeled by Extrusion Ports. The box indicates a manipulator used for interactive insertion of the bolt using pointing devices such as the mouse.or space-mouse.

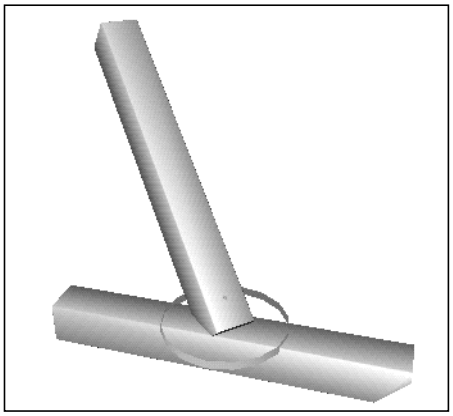


Figure 4: Plane Ports model connections between co-planar object faces. The circle indicates a special manipulator that can be used to reorient the parts along the rotational degree of freedom of Plane Port connections.

base [9], whose top-level port concepts are the following:

1. Extrusion Ports can be used to model peg-in-hole-like insertions of objects. An extrusion is a three-dimensional sweep geometry where a two-dimensional profile is extruded along a spine normal to the profile. Extrusion ports are further refined into female ports, where the sweep describes a hole, and male ports, where the sweep describes a solid (e.g. a shaft). A more general port concept, sweep ports, is also available. Connections between Extrusion Ports allow for one translational and one rotational degree of freedom. Figure 3 shows an example.

2. Plane Ports are used to model two-dimensional, planar connection areas. Connections between plane ports allow for two translational and one rotational degree of freedom. An example is shown in Figure 4.

3. Point Ports can model point-like object connections that induce no translational and at most one rotational degree of freedom when objects are connected. Point Ports can be understood as borderline cases of both extrusion and plane ports but introduce a separate concept that has conceptual and computational advantages. See Figure 5 for an example.

Besides the taxonomy of port concepts, the knowledge base also contains different kinds of formal connection relations

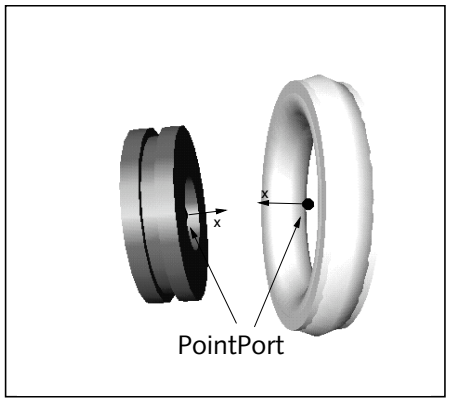


Figure 5: The connection between the wheel and the tire that allows for no translational degrees of freedom is modeled with Point Ports. One rotational degree of freedom is preserved.

that define further constraints concerning the relative movement of connected objects. E.g. screw-in-hole insertions can be defined with either independent or dependent rotational and translational degrees of freedom (in the latter case the screw needs to be rotated in order to be inserted into the hole). Or, a welding-type of connection can be defined as one that allows no transformational degree of freedom at all. These types of connection relations are formally modeled by using an extension of Roth's Freiheitsmatrizen [14].

Finally, the part model knowledge base contains generic descriptions of each kind of mechanical object considered in a specific assembly task. For each mechanical object, its connection ports, their locations within the object and their admissible connection types are defined. Optionally, so-called „hot-spots" can be defined that describe preferred locations for part matings within extended ports such as Extrusion and Plane Ports.

During an assembly task in the virtual environment, each visualized mechanical object is represented by an instance of a knowledge base concept. The now available object information about their location, ports, and connectablities suffices to simulate assembly and disassembly of parts and the modification of connections along transformational degrees of freedom. For example, simulation of assembly operations proceeds in the following four steps:

1. The objects to be mated including their connection ports are selected. The ports must be compatible with each other and provide enough capacity (unconsumed by connections with other objects) to mate with each other.
2. The parts are tentatively mated based on their port descriptions. Since currently consumed port capacities are taken into account, the result is a locally collision-free mating of the two parts. However, if the two objects are already part of larger aggregates, interpenetrations of other parts might occur as side-effect.
3. If a collision has occurred in step 2., the part mating is slightly changed along its induced transformational degrees of freedom until a globally collision-free state is reached.
4. The new assembly state is now visualized. The object port representations are updated concerning their new capacities and connection relations. Note that the assembly of two aggregates may result in several part matings which are all detected by the system.

Similarly, the simulation of other operations such as disassembly and rotations is based on the combined evaluation of knowledge-based port descriptions and collision avoidance strategies, thus, guaranteeing globally collision-free assembly states. Also, the simulation of assembly-related operations is computationally inexpensive and can be performed in real-time.

Related Work: An overview of assembly mating conditions in mechanical engineering including several modern CAD-systems can be found in [6]. Our approach is, however, more similar to the so-called (robot) assembly-languages, e.g. RAPT [13], in that is based on conceptual representations („features") rather than low-level topological entities such as edges and faces. Our Port-taxonomy is inspired by research in both mechanical engineering and computer graphics and is a novel contribution.

IV. Dynamic Conceptualization of Objects and Aggregates

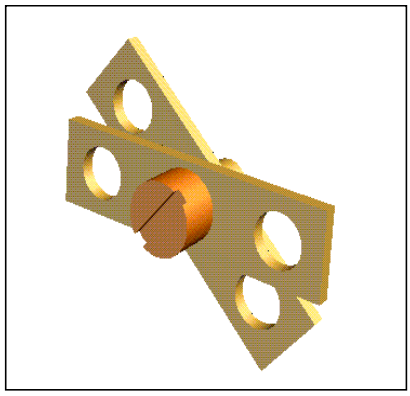
In the course of an on-going assembly task, aggregates are constructed in the virtual environment which might form meaningful subassemblies of a target aggregate. There are (at least) two reasons, why an intelligent design support system should maintain a dynamic internal model of the changing environment: First, the human user is likely to refer to constructed assemblies in natural language instructions and, thus, the system should have an understanding of what objects the user is referring to. And, second, the user will have expectations about how the visualized objects and assembly groups are typically combined in larger assemblies. For example, in Figure 1, the user instructs the system „attach the propeller to the airplane" where both the propeller and the airplane are complex aggregates. In order to process such an instruction, the system must step-keepingly infer which complex objects are a propeller or, respectively, an airplane. Also, the user will expect that the propeller is attached to the airplane in a particular fashion, namely facing front. Thus, the system needs to have some prior model knowledge of the target aggregate, and, must dynamically create, modify, and delete its conceptual representations of the objects and assemblies in the virtual environment. We call this *dynamic conceptualization* [16].

In order to operationalize dynamic conceptualization, we have developed the knowledge representation and reasoning framework Coar („Concepts for Objects, Assemblies, and Roles"). The Coar framework reflects three fundamental design issues for intelligent agent reasoning in (virtual) assembly environments [7]:

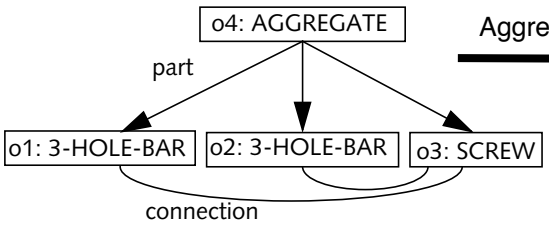
1. In Coar, a structured model of the target aggregate can be defined that describes its functional decomposition into subassemblies and primitive parts. During an on-going assembly task in the virtual environment,

```
PROPELLER
    is-a: ASSEMBLYGROUP
    part has-blade-1 #1: PROPPLERBLADE
    part has-blade-2 #1: PROPELLERBLADE
    part has-spinner #1: SPINNER
    pp-constraint connection  <has-blade-1>
                              <has-spinner>
    pp-constraint connection <has-blade-2>
                              <has-spinner>
    pp-constraint not parallel_x <has-blade-1>
                              <has-blade-2>
```

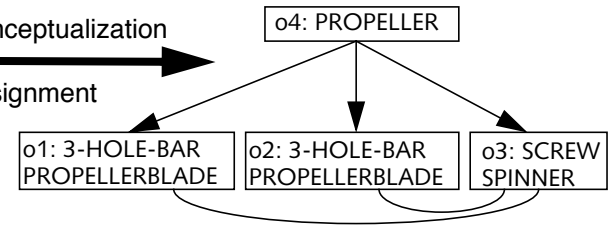
(a) COAR definition of a generic propeller of the Baufix airplane. The propeller consists of two PROPELLERBLADEs (a role type that can be assumed by, e.g., 3-hole-bars), and one SPINNER (a possible role type of screws). Further, each of the screws must be connected to the spinner, and the two blades must not be parallel to each other.



(b) A Baufix propeller is assembled in the virtual environment.



(c) In the situation shown in (b), the assembly simulation component notifies the COAR reasoner about new port connections. These connection relations are propagated to short-term concepts of mechanical objects; aggregation creates a short-term concept representing an unstructured aggregate.



(d) Aggregate conceptualization infers that the so far unstructured aggregate o4 matches the definition of the Baufix propeller. Role assignment reclassifies the propeller components according to their functional role within the Baufix propeller.

Figure 6: Dynamic conceptualization of the assemblies constructed in the environment within the Coar framework.

evolving aggregates are matched against this structural model and conceptual representations of constructed assembly groups are added to the dynamic model of the assembly scene.

2. If construction kits with multi-functional parts are used in an assembly task, these parts may assume specific functional roles when becoming components of assembly groups. In the Baufix airplane (see Figure 1), for example, ten screws are part of the complete construct, of which two are used as axles. In COAR, a distinction is made between *object types* that model multi-functional objects and assembly groups and *role types* that model specific functional aspects of objects and aggregates. COAR reasoning dynamically reclassifies object representations w.r.t. the underlying role type taxonomy when objects are used as components of larger assemblies.

3. Within the larger system architecture of the virtual assembly environment, COAR representations complement the 3D geometric scene description used for visualizing the assembly scene. The 3D geometric scene description contains information about the spatial arrangement of the scene objects, such as location, distance and size. COAR representations can access this spatial information when-needed and, thus, conceptual reasoning over COAR representations is closely intertwined with spatial reasoning over the geometric scene description [8].

The COAR knowledge representation formalism is a Frame-based/semantic-network-type of language. Generic descriptions, or classes, of objects and assemblies are called *long-term concepts*. Long-term concepts are divided into object types and role types (see above). Individual objects and aggregates in the environment are represented by *short-term concepts*. Short-term concepts are dynamically created and deleted during an assembly task; they are instances of long-term concepts but - in contrast, e.g., to conventional object-oriented programming languages - might change their classification w.r.t. their role type during their life-cycle.

Reasoning in Coar includes the following domain-independent inferences which are used for updating the short-term concept knowledge base in response to changes of the assembly state in the virtual environment:

- Propagation of connections: Connection relations are established between short-term concepts of composite

objects provided that a connection relation has already been established betwen some of their parts.

- Aggregation: Connected objects are grouped together by the creation of new short-term concepts representing unstructured, „flat“ aggregates.

- Aggregate conceptualization: Short-term concepts of „flat“ aggregates are matched against the structured model of the target assembly. New short-term concepts representing assembly groups are created if the match is established.

- Role assignment: Short-term concepts representing components of assembly groups are reclassified w.r.t. the role type hierarchy of long-term concepts in order to reflect the changing functional role of objects when used in different assemblies. Also, new context-dependent attributes are assigned to component representations by this inference.

Figure 6 shows an example of knowledge representation and reasoning in COAR, where the propeller of the Baufix airplane is assembled in the virtual environment and recognized as such by the COAR reasoner.

Related Work: Dynamic Conceptualization is a novel contribution to virtual environment research. COAR's aggregate conceptualization builds on similar inferences used in pattern recognition, e.g. the ERNEST system [10], and description logics [12]. The distinction between objects types and role types is based on a similar distinction in CONCEPTUAL GRAPHS [15].

VI CONCLUSIONS

The main goal of our research is to create easy-to-use interfaces for interactive construction of virtual prototypes from 3D-visualized, CAD-based parts. We have identified three key factors for creating knowledge-enhanced VR-interfaces that can relieve users from technical detail: (a) natural, task-level human-computer-interaction, (b) knowledge-based simulation, and (c) dynamic conceptualization of the virtual environment. Our approach is implemented in an integrated system, the CODY Virtual Constructor, that realizes novel contributions to each of these key factors.

VI ACKNOWLEDGMENTS

The research described here is partly supported by the German National Science Foundation (DFG) and by the Federal State of North-Rhine Westphalia. The authors are indebted to Martin Hoffhenke and Stefan Kopp for numerous contributions to the CODY Virtual Constructor.

REFERENCES

[1] J.A. Adam. Virtual reality is for real. *IEEE Spectrum*, 30(10):22-29, October 1993.

[2] N. I. Badler, B.L. Webber, J. Kalita, J. and J. Esakov. Animation from Instructions. In N.I. Badler, B.A. Barsky, and D. Zeltzer (eds): *Making them Move: Mechanics, Control, and Animation of Articulated Figures*. Morgan Kaufmann, 1991.

[3] R.A. Bolt. Put-that-there: Voice and gesture at the graphics interface. In *ACM SIGGRAPH-Computergraphics*, 1980.

[4] Y. Cao, B. Jung, and I. Wachsmuth: Situated Verbal Interaction in Virtual Design and Assembly, IJCAI-95 Videotape Program. Abstract in: *Proc. Fourteenth International Joint Conference on Artificial Intelligence*, 1995, 2061-2062.

[5] F. Dai and M. Göbel. Virtual Prototyping - an approach using VR-techniques. *Proceedings of the 14th ASME International Computers in Engineering Conference*, 1994.

[6] B.W. Henson and N.P. Juster. Information requirements for the support of assembly mating conditions. *Proceedings DECT'97, ASME Design Engineering Technical Conferences,* 1997.

[7] B. Jung. Reasoning about Objects, Assemblies, and Roles in On-going Assembly Tasks. *Proceedings DARS'98: International Symposium on Autonomous Robotic Systems*, Springer, 1998 (to appear).

[8] B. Jung and I. Wachsmuth: Integration of Geometric and Conceptual Reasoning for Interacting with Virtual Environments. *Proc. 98'AAAI Spring Symposium on Multimodal Reasoning,* 1998, 22-27.

[9] S. Kopp. *Ein wissensbasierter Ansatz zur Modellierung von Verbindungen für die virtuelle Montage*. Diplomarbeit, Technische Fakultät, Universität Bielefeld, 1998.

[10] F. Kummert, H. Niemann, R. Prechtel, and G. Sagerer. Control and explanation in a signal understanding environment. *Signal Processing*, 32:111-145, 1993.

[11] M. Latoschik and I. Wachsmuth: Exploiting distant pointing gestures for object selection in a virtual environment. In Ipke Wachsmuth and Martin Fröhlich (eds.): *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Volume 1371, 185-196, Springer-Verlag, 1998.

[12] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In *Principles of Knowledge Representation and Reasoning*, pages 485-496. Morgan Kaufmann, San Francisco, CA, 1994.

[13] R.J. Popplestone, A.P. Ambler, and I.M. Bellos. An interpreter for a language describing assemblies. *Artificial Intelligence*, 14(1):79-107, 1980.

[14] K. Roth. *Konstruieren mit Konstruktionskatalogen*. Band 2, 2. Auflage, Springer, 1994.

[15] J. Sowa. Using a lexicon of canonical graphs in a semantic interpreter. In M. Evens (ed.): Relational Models of the Lexicon. Cambridge University Press, 1988.

[16] I. Wachsmuth and B. Jung: Dynamic Conceptualization in a Mechanical-Object Assembly Environment, *Artificial Intelligence Review*, 10 (3-4) (pp 345-368).

[17] I. Wachsmuth and M. Fröhlich, *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Volume 1371, Springer-Verlag, 1998.