Dynamic Conceptualization in a Mechanical-Object Assembly Environment

I. Wachsmuth and B. Jung
Faculty of Technology, University of Bielefeld
D-33501 Bielefeld, Germany

(Received ..... ; Accepted in final form .....)

Abstract. In an experimental setting of mechanical-object assembly, the CODY ("Concept Dynamics") project is concerned with the development of knowledge representations and inference methods that are able to dynamically conceptualize the situation in the task environment. A central aim is to enable an artificial agent to understand and process natural-language instructions of a human partner. Instructions may build on the current perception of the assembly environment on the one hand, and on the other on the knowledge-based understanding of grouped structures in the developing construct. To this end, a dynamic conceptualization must integrate information not only describing the types of the objects involved, but also their changing functional roles when becoming part of structured assemblies.

We have developed an operational knowledge representation formalism, COAR ("Concepts for Objects, Assemblies, and Roles"), by which processes of dynamic conceptualization in sequences of assembly steps can be formally reconstructed. Inferences concern the assertion or retraction of aggregate representations in a dynamic knowledge base, as well as the computation of role changes for individual objects associated herewith. The structural representations integrate situated spatial features and relations, such as position, size, distance, or orthogonality, which are inferred on need from a geometry description of the task environment. The capacity of our approach has been evaluated in a 3D computergraphics simulation environment.

Key words: Knowledge Representation, Hybrid Structural and Spatial Representations, Situated Communication

1. Introduction

Artificial agents that act in an environment must be able to maintain knowledge about objects undergoing change in the course of a situation. Let us assume an agent that has sensors to perceive input from a task environment (e.g., images and speech), and actuators to affect the outside world. Let us further assume that we want to communicate with the agent, for instance, issue natural language instructions to direct the agent's actions. For instance, we could want the agent to change a configuration of objects as this is typically encountered in robot assembly tasks. Then the agent has to build an internal representation of its environment which forms the basis of understanding what instructions refer to, and which can be manipulated to explore the effect of actions before they are actually attempted in the task environment.

In order to conceptualize a changing task environment, the agent must draw on certain world knowledge, for instance, knowledge which describes the possible roles that an object can take on in an assemblage. On the other hand, the world knowledge at the agent's disposal can never be complete with respect to the multitude of situations to which the agent could be exposed. Hence, being coupled to its environment by way of sensors and actuators, the agent needs to *exploit the actual situation*, as a source of information, for example, to integrate visual input while processing a natural language instruction. And it is crucial for the agent to modify its conception of the environment as it changes, in other words, the task environment must be "dynamically conceptualized" by the agent.

1.1. SFB 360 - Reference Situation

The work reported here is part of the Collaborative Research Centre SFB 360 *"Situated Artificial Communicators"* at the University of Bielefeld. One of the long-term aims of the research in SFB 360 is the construction of an artificial system that is able to communicate and cooperate with a human partner in the accomplishment of assembly tasks. In our reference situation from mechanical-object assembly, which touches all projects in SFB 360 (see site description, Rickheit & Wachsmuth, this volume), complex aggregates, such as a model airplane, are to be constructed from the components of a wooden construction kit, named "Baufix" (Figure 1). The human partner who has a diagram of the target aggregate takes on the part of an *instructor* and directs the system to carry out particular assembly steps. An artificial communicator - in the long range an assembly robot - takes on the part of a *constructor* and carries out the necessary actions. This reference situation is referred to as "instructor-constructor scenario".

Natural communication among human partners to a great extent grounds on their current perception of the situation in the environment, i.e., is situated. When both partners in the instructor-constructor scenario share an environment which they both can (at least partially) observe, verbal instructions will likely make reference to what can be seen in the current situation. What can be seen in an assembly environment could also include how constructed aggregates are perceived as structured assemblies by the cooperating partners. Furthermore, while some of an object's properties may remain unchanged in the course of an assemblage, others will depend on the actual situation. Thus, when processing the verbal instructions of the instructor, the constructor needs to draw on what is currently true in a changing situation.

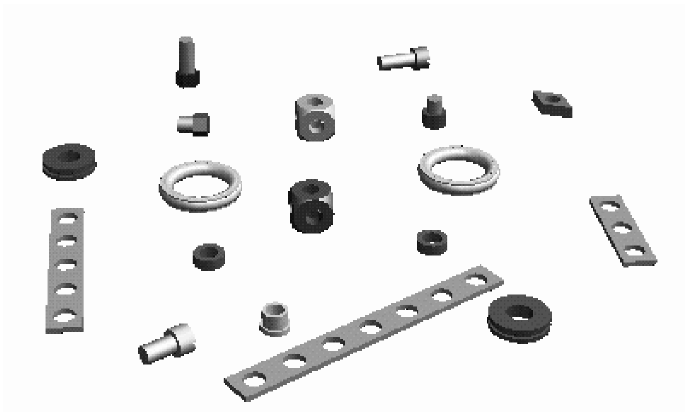

Figure 1. Baufix construction kit (sample parts)

If we want artificial communicators to make better profit of the ways humans communicate, the artificial communicator needs to be able to process situated verbal instructions by integrating what is *said* with what is *seen* (a *situated artificial communicator*). Much work in SFB 360 is spent on the reception, processing, and understanding of acoustic and visual information and on how the visually available information and spoken information are to be put in relation with each other (Fink et al., 1994; Fuhr et al., 1995; Moratz et al., 1995). Besides of sensorimotor and communicative abilities, the artificial communicator must have at its disposal a repertoire of suitable background knowledge and inference capabilities. Most crucially, the constructor must be able to process instructions and accomplish construction steps in a continually altered environment. This is why we need dynamic conceptualization.

1.2. Dynamic Conceptualization - The CODY Project

A conceptualization is an abstract, simplified view of a world represented for some purpose (Genesereth & Nilsson, 1987 ; Gruber & Olsen, 1994). By *dynamic conceptualization*, roughly, we mean that an agent maintains and updates an internal view of a changing environment (Jung & Wachsmuth, 1994). To accomplish this, internal representations must be coupled with the environment, and they must be created, modified, and deleted as the current situation demands. The purpose of dynamic conceptualization in SFB 360's instructor-constructor scenario is that instructor and constructor can communicate about a dynamic scene. While a construction is in progress, internal representations are to be updated dynamically by the constructor, in the same way as the human instructor is assumed to have a dynamic mental model which adapts to the changing characteristics of the task environment (Rickheit & Strohner, 1994). Thus, the knowledge representations needed by the artificial communicator (constructor) will change over time and involve information which is replaced or superseded.

In SFB 360, the CODY ("Concept Dynamics") project is concerned with the development of knowledge representations and inference methods that are capable of dynamically conceptualizing the situation in a task environment of mechanical-object assembly. To this end, internal representations must accumulate informations not only describing the types of the objects involved, but also the functional role of objects in larger assembly groups. For instance, a *bar* may take on the role of a *propeller blade* when becoming part of an airplane undercarriage. Thus, object conceptualizations cannot be modeled in the 'classical way', i.e. as instances of a long-term repertoire of generic concepts. Rather, they have a life cycle in which they undergo changing classifications, associated with changing feature attributions, where role aspects tend to prevail as construction is progressing.

As a core part of CODY, we have developed and implemented a knowledge representation formalism, COAR, by which processes of dynamic conceptualization in sequences of assembly steps can be formally reconstructed. COAR builds on ideas of semantic networks but provides more than classic instantiation: Internal representations are constructed, maintained and restructured along with the changes that occur when assemblies are constructed from a variety of multi-function construction objects. COAR representations integrate taxonomic, partonomic, and functional information of objects and aggregates used in assembly, and they can access situated spatial information from the task environment. *Long-term concepts*, serving as background knowledge, describe mechanical objects, assemblies, and roles. *Short-term concepts*, serving as temporal descriptions of individual entities in dynamic conceptualization, bear current classifications of entities with respect to object type and role type. Whereas object-type classification persists, role-type classification is dynamic, i.e. it may change over time and thus evoke changing feature attribution. Inferences on short-term concepts concern the assertion or retraction of representations for constructed aggregates in a dynamic knowledge base, and the computation of role changes for individual objects combined herewith.

The capacity of our approach is so far evaluated in a 3D computer-graphics simulation environment, the Virtual Constructor, in which the assemblage of model airplanes and similar constructs from a construction kit can be simulated. The Virtual Constructor is briefly described in Section 2, before we go into detail explaining the CODY representation language, COAR, in Section 3. The usage of COAR representations for dynamic conceptualization in Section 4, where coherence conditions specifying the creation, modification, and retraction of short-term concepts are described. In concluding, we discuss our achievements, their current and prospective applications, and where we want to go next.

2. The CODY Virtual Constructor

In order to probe our work in a task environment accessible to perception and action (not yet available as a full technical system integrating all necessary components), we currently work with a computergraphics environment, the Virtual Constructor (Jung et al., 1995). The CODY Virtual Constructor is a knowledge-based simulation system for the interactive assembly of complex aggregates from 3D objects. The external scene is reproduced, somewhat simplified, as a virtual scene, the *virtual assembly workbench*, on which the assemblage of model airplanes and similar constructs from a construction kit can be simulated.[1]

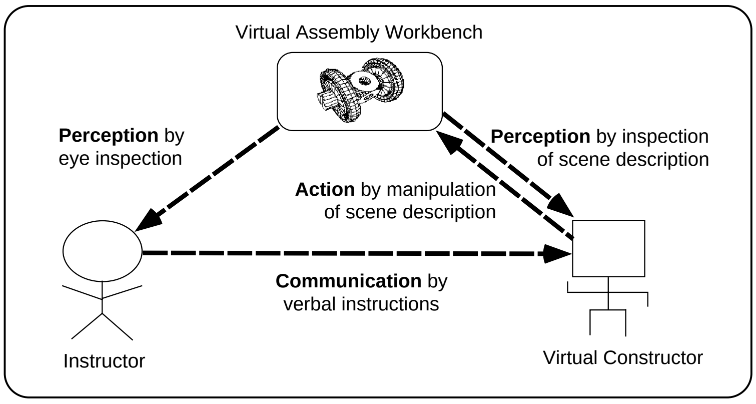
Virtual Assembly Workbench
Perception by eye inspection
Perception by inspection of scene description
Action by manipulation of scene description
Communication by verbal instructions
Instructor
Virtual Constructor

Figure 2. Instructor-constructor scenario with virtual assembly workbench

The virtual scene can be inspected by both communicators, i.e., is a commonly perceived world situation. The (human) instructor can view the scene from changing perspectives, whereby the (artificial) constructor has knowledge of the instructor's current field of view. The instructor can issue simple (written) natural-language commands. The Virtual Constructor can change the world situation by manipulating the geometry scene description (cf. Figure 2).

As a sample application for the Virtual Constructor we chose the assembly of a simple target aggregate - a model airplane - from the parts of construction kit (cf. SFB 360 reference situation, Section 1.1). These parts, e.g. bolts, blocks, and bars of different sizes, can be assembled in various ways. Any aggregate that can be built from Baufix building parts, but no aggregate that is physically impossible, can be assembled on the virtual assembly workbench. For instance, the partially assembled model airplane shown in Figure 3 (left) was produced on the virtual assembly workbench. In Figure 3, also, the effect of the instruction *"attach the propeller to the fuselage"* is illustrated (right).

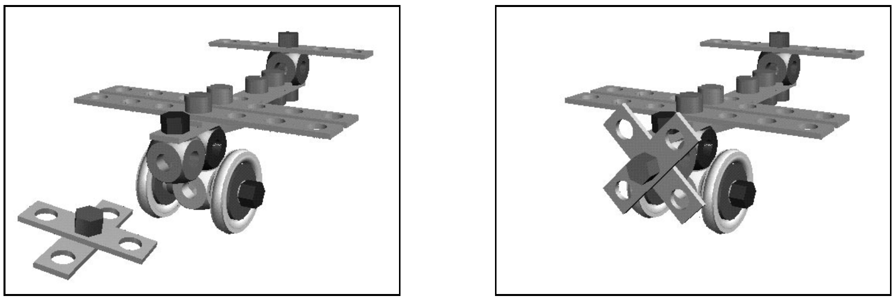

Figure 3. *"Attach the propeller to the fuselage"*

Instructions to the Virtual Constructor may refer to object types (e.g. *bolt*, *bar*, etc.), to objects' current functional roles in the evolving airplane (e.g., *axle*, *propeller blade*, etc.), to visual object properties such as color, form, and size, and finally, to object locations (e.g. *"upper left"*). Besides single parts, also assembled aggregates can be referred to, and any part or aggregate may be referred to as *"thing"*. In some cases, the instructor can alter the situation reached in the preceding action of the constructor by issuing a follow-up instruction which is correcting the previous one. Some sample instructions are given below.

> *put the left bolt through the red disc*
> *attach the thing to the left to the bottom of the block*
> *mount the wheel on the axle*
> *attach the long bolt to the top of the undercarriage*
> *detach the tire from the drum*
> *rotate the bar crosswise to the tail unit*
> *no, the other way round*

Instructions typically make reference to the current object configuration in the environment as it is perceived from the current perspective.

Spatial expressions in instructions can be evaluated by accessing an internal camera model describing the current user perspective and the geometric scene description of the virtual environment. The geometric scene description, contains among other information the position and orientation of objects and aggregates in three-dimensional space. To find reference objects for natural language expressions like *"the left bolt"*, the Virtual Constructor will evaluate the geometric scene description and the camera model to select the bolt that is most to the left as seen from the current user perspective.

While a key idea is to exploit the geometric scene description to the greatest extent possible for instruction processing, this alone does not suffice. Instead, the geometric scene description is enriched with a layer of structural symbolic representations that account for the Virtual Constructor's current conceptualization of the environment. They describe which aggregates are formed in the current situation and which functional roles objects have assumed in the evolving airplane. Only by integrating the geometric scene description (including the camera model) with these dynamic knowledge representations, natural-language expressions like *"the left axle"* or *"the wheel on the right"* can be understood. By using the Virtual Constructor, we are in the position to satisfactorily evaluate the effect of dynamic conceptualization, even when an integrated system with sensori-motor components is not yet available.

3. A Concept Language for Objects, Assemblies, and Roles

The concept language COAR ("Concepts for Objects, Assemblies, and Roles") was developed as a formal basis for dynamic conceptualization in assembly tasks. Concepts for mechanical objects, assemblies that can be built from these objects, and finally, the changing roles that objects can take on in larger assemblies can be modeled. Main design goals of COAR were to supply language constructs that enable the classification of constructed aggregates as instances of predefined assemblies and which allow the assignment of roles to object representations, according to their current aggregate context.

The language provides means to describe concepts by their attributes and to relate different concepts to one another by a small set of semantic relations. COAR, roughly, is a synthesis of the semantic-network language ERNEST (Sagerer, 1990; Niemann et al., 1990), which is used in a number of SFB 360 projects, and the description logics for part-of hierarchies introduced in (Padgham & Lambrix, 1994). COAR further allows the integration of spatial knowledge in concept definitions, in that descriptions of aggregates, besides of has-part and connected-to relations, may include geometric relations between parts, e.g. `orthogonal`$_x$ or `touches`, which would have to be evaluated in a geometric scene description when establishing an instantiation (Jung & Wachsmuth, 1995).

3.1. Object Types and Role Types

In COAR, a distinction is made between two kinds of concepts: *Object types* are used to model mechanical objects, their subparts (e.g., their connection ports), and assemblies. *Role types* are used to model the specific functional aspects of objects in larger assemblies.

One motivation for introducing the object type – role type distinction is to clearly separate an object's essential (context-independent) properties from its specific functional properties in larger wholes. A further motivation is to exploit this distinction in the design of inference methods which update an agent's internal model as the external situation changes. While the object type of an individual representation will always remain the same, its role type may change over time. Therefore, inference mechanisms for updating an agent's internal model must never consider the reclassification of individual representations w.r.t. their object type but only w.r.t. their role type.

Distinctions similar to our separation of object types and role types have been proposed by Sowa (1988), Way (1991), and Guarino et al. (1994). Sowa, for example, distinguishes between *natural types* and *role types*, the latter being "subtypes of natural types in some particular pattern of relation-ships" (Sowa, 1988, p.120). In the assembly tasks considered here, this particular pattern of relationships is an object's aggregate context, i.e. objects may assume specific roles when used in larger assemblies. Sowa (1988), and also Guarino et al. (1994), have proposed to model object types and role types in a single generalization hierarchy, with role types subordinated to objects types. This is different in COAR, where object types and role types are modeled in two distinct hierarchies. Concepts in the different hierarchies are not in a specialization relationship but they are linked through a special semantic relation *role-of.* This allows, for example, that the same role may be assumed by objects of different types.

3.2. The COAR Language

COAR is an operational knowledge representation formalism, by which processes of dynamic conceptualization in mechanical assembly can be formally reconstructed. The COAR language is used to model *long-term concepts*, serving as static background knowledge, and *short-term* concepts, serving as temporal descriptions of individual entities in dynamic conceptualization.

Long-term concepts are generic descriptions in a frame-like notation. The abstract syntax for long-term concepts is as follows:

long-term concept: *conceptname*
    specialization-of: *conceptname**
    [role-of *relationname*: *conceptname*$^+$]
    (part *relationname int*: *conceptname*)*
    (connection *min max*: *conceptname*)*
    (pp-constraint *relation* <*relationname*$^+$ ><*relationname*$^+$ >)*
    (attribute *attributename*: *valuedomain*)*

Short-term concepts are modeled as instances of long-term concepts; however, they can change classification during their life cycle. The abstract syntax for short-term concepts is as follows:

short-term concept: *individualname*
    instance-of: *conceptname*$^+$
    [role-of *relationname*: *individualname*]
    (part *relationname*: *individualname*)*
    (connection: *individualname*)*
    (attribute *attributename*: *value*)*

In detail, the language constructs in COAR expressions describing long-term concepts serve the following purposes:

The *specialization-of ("isa")* slot relates concepts to their superconcepts. All properties of a concept are inherited to its specializations. Multiple superconcepts can be defined. Object types will only be specializations of other object types, and role types will only be specializations of other role types.

The *role-of* slot relates role types with object types. That is role types, typically, will have their role-of slot filled with possibly several object types. Object types will have no role-of slot. If a role type is related to a certain object type, then all instances of this object type may be assigned that role type during their life cycle.

The *part* slot relates composite objects with their components. Different parts of composites are distinguished by their names. Also, number and type restrictions must be defined. Composites, typically, have a pp-constraint filled which defines structural conditions between their parts such that the existence of assembled composites can be inferred from the configuration of their parts. Parts of a composite can either be object types or role types. The latter case indicates that objects will assume a certain role when used as part of a composite.

The *connection* relation is used to define the possible physical connections between mechanical objects. Connection relations will only be defined for long-term concepts which are at the bottom of the part decomposition, i.e. in our case the mechanical objects' connection ports. For example, it can be defined that certain bolts can be fitted in the holes of certain bars by defining an appropriate connection relation between these objects' connection ports. Objects other than ports are assumed to be connectable to all other objects, where the establishment of actual connections between their instances will depend on connectability of their ports (and on collision tests checked in the geometric scene description).

With the *pp-(part-part)constraint* construct, relations can be stated which must hold between two parts of an instance. To specify these (possibly indirect) parts, two paths of part names are specified. Two kinds of constraints can be stated: (1) *connection constraints*, which require connection relations to be established between the corresponding instances; (2) *geometric constraints*, e.g. $\mathtt{orthogonal}_x$, which trigger tests in the geometric scene description on an if-needed basis. That is, spatial knowledge derivable from the geometric scene description can be integrated in COAR concept definitions.

The *attribute* slot allows the definition of concept-inherent properties, i.e. properties that hold for an object or role in isolation rather than expressing its relations to other entities. Typical attributes are color and shape.

COAR also comes with a number of predefined top-level (long-term) concepts. The concept OBJECTTYPE serves as most general object-type concept, and the concept ROLETYPE serves as the most general role type. Furthermore, a concept AGGREGATE is defined as a specialization of OBJECTTYPE representing unstructured, "flat" aggregations of connected objects. Any set of connected objects can such at least be conceptualized as an AGGREGATE. Finally, a concept ASSEMBLY-GROUP is defined as a specialization of AGGREGATE and serves as the most general concept describing structured assemblies. The reasoning processes described below will only attempt to infer the existence of instances of user-defined concepts that are specializations of the concept ASSEMBLYGROUP.

The language constructs for short-term concepts are to a large extent identical with the language constructs for long-term concepts. The *instance-of* slot indicates the current classification of short-term concepts w.r.t. long-term concepts. The instance-relation w.r.t. role-type concepts is dynamic and can change after any assembly step. When a short-term concept is classified as instance an of a long-term concept, then also all attributes defined in the long-term concept must be asserted and properly initialized in the short-term concept, i.e. short-term concepts can be assigned changing attribute sets during their life cycle. The *role-of* slot is used to relate short-term concepts representing the same external entity, where one short-term concept accounts for the entity's object-type conceptualization and the other one for its role-type conceptualization.

3.3. Using COAR for Modeling Long-term Concepts

Long-term concepts serve as static background knowledge for dynamic conceptualization of objects and assemblies in the task environment. In our current demonstration system, long-term concepts are defined in two knowledge bases. A knowledge base "Building Blocks" contains long-term concepts for the multi-functional mechanical objects, like bolts and bars, and their connection ports. A knowledge base "Airplane" contains long-term concepts describing the functional decomposition of the model airplane which serves as target aggregate in our scenario. The airplane's assembly groups may be further structured in subassemblies. Functional decomposition grounds with role-type concepts describing the function of single objects in assembly groups. Role-type concepts are linked to their corresponding object-type concepts in the knowledge base "Building Blocks" via the role-of relation.

The partitioning of the background knowledge into two knowledge bases reflects the different ontologies with respect to which the external objects can be conceptualized. As a further advantage, this modularization supports reuse of the modeled ontologies. For example, the knowledge base "Building Blocks" could be reused for other target aggregates, such as cars or scooters.

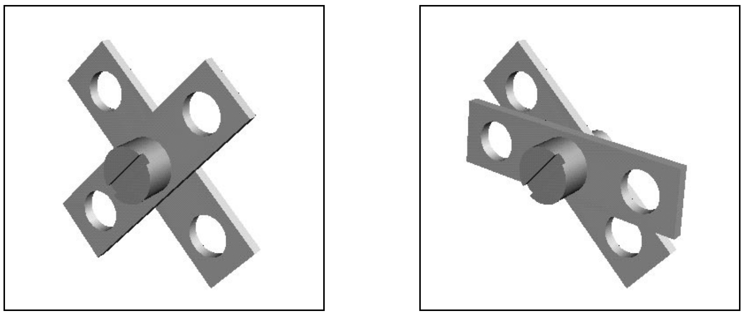

Figure 4. A propeller and an aggregate which is no propeller

In the following example, a long-term concept for the propeller of the model airplane is defined. As shown in Figure 4, a propeller consists of three parts: two propeller blades and the propeller hub. Furthermore, we want to define a propeller where the two propeller blades are orthogonal to each other. On the left of Figure 4, there is a propeller but the assembly on the right of Figure 4 is not considered an instance of the propeller concept since the two bars are not at a right angle.[2]

```
long-term concept: 3-HOLE-BAR                           // an object type
     specialization-of: BAR                             // in knowledge base
     part has-hole-1 #1: HOLE                           // "Building Blocks"
     part has-hole-2 #1: HOLE
     part has-hole-3 #1: HOLE
     attribute color: wood

long-term concept: PROPELLERBLADE                       // a role type
     specialization-of: AIRPLANEPART                    // in knowledge base
     role-of as-object: 3-HOLE-BAR                      // "Airplane"

long-term concept: PROPELLER                            // an object type
     specialization-of: ASSEMBLYGROUP                   // in knowledge base
     part has-propellerblade-1 #1: PROPELLERBLADE       // "Airplane"
     part has-propellerblade-2 #1: PROPELLERBLADE
     part has-propellerhub #1: PROPELLERHUB
     pp-constraint connection  <has-propellerblade-1.as-object.has-hole-2>
                               <has-propellerhub.as-object.has-shaft>
     pp-constraint connection  <has-propellerblade-2.as-object.has-hole-2>
                               <has-propellerhub.as-object.has-shaft>
     pp-constraint orthogonal_x <has-propellerblade-1> <has-propellerblade-2>
```

Figure 5. COAR definitions of long-term concepts (inherited slots not shown)

Figure 5 illustrates how such a propeller made of Baufix parts is modeled in COAR. The first concept, 3-HOLE-BAR, is a specialization of a general BAR and therefore, like the bar, an object-type concept. The 3-HOLE-BAR has three holes which are modeled as its parts. Each of the part slots has a unique name by which it can be distinguished from other parts. The PROPELLERBLADE is some role type which is a specialization of a general role type AIRPLANEPART. The role-of slot of the PROPELLERBLADE relates it to the 3-HOLE-BAR, which indicates that every instance of 3-HOLE-BAR can assume the role of PROPELLERBLADE. The concept PROPELLER is a specialization of the ASSEMBLYGROUP concept. Three parts are defined. The first part slot, for example, must be filled with exactly one instance of the PROPELLERBLADE. Also, three pp-constraint constructs are defined that describe the arrangement of the these parts. The first pp-constraint, for example, requires the propeller hub to be connected with the second hole of the first propeller blade. The third pp-constraint requires the two propeller blades to be in orthogonal orientation to each other.

Besides the two knowledge bases “Building Blocks” and “Airplane”, two additional knowledge bases are used to couple COAR instances to the geometric scene description. One of these additional knowledge bases provides generic geometry models of the mechanical objects used in the assembly environment. Besides containing purely presentational data like wireframe models, the generic geometry models also serve as analogical representations describing, e.g., the building blocks' intrinsic orientation and the exact positions and orientations of their connection ports. For instance, the rotation axis and the intrinsic orientation of bolts are kept with geometry models for each particular bolt type (e.g. short hexagon bolt). We are currently reworking the knowledge base to keep such information within generalized, parametric geometry models, called *imaginal prototypes* (e.g., of a general bolt). The geometric knowledge contained in these imaginal prototypes will be inherited to each particular geometry model; cf. Figure 6 for an illustration.

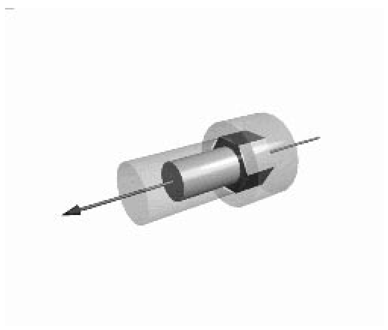

Figure 6. Imaginal prototype of a bolt superimposed on a concrete geometry model

The other additional knowledge base defines several qualitative spatial relations over the geometry models. In our current demonstrator system, the following spatial relations are defined: $\texttt{orthogonal}_x$, $\texttt{orthogonal}_y$, $\texttt{orthogonal}_z$ $\texttt{parallel}_x$, $\texttt{parallel}_y$, $\texttt{parallel}_z$, `touches`, and `near`. These spatial relations, as was described above, can be used in the pp-constraints of COAR concepts. Whether or not these relations hold between two individuals is tested in the geometric scene description. To this end, we use routines which keep track of varying situation parameters.

3.4. Using COAR for Modeling Short-Term Concepts

Short-term concepts serve as temporal conceptualizations of individual entities (objects and aggregates) in the current task environment. Dynamic conceptualization involves the step-keeping creation, modification, and retraction of short-term concepts as the external situation progresses. Short-term concepts account for the classification of entities with respect to object type and role type. They further mediate the derivation of situated spatial features of objects and aggregates, such as position, orientation, or size, and spatial relations between objects and aggregates, such as orthogonality or nearness, which are inferred on need from geometry descriptions in the task environment.

```
short-term concept: bar-1
      instance-of: 3-HOLE-BAR
      part has-hole-1: bar-1/hole-1
      part has-hole-2: bar-1/hole-2
      part has-hole-3: bar-1/hole-3
      connection: bolt-1
      attribute color: wood

short-term concept: airplanepart-1
      instance-of: PROPELLERBLADE
      role-of as-object: bar-1
      connection: airplanepart-2
```

Figure 7. COAR descriptions of short-term concepts (explanation see text)

For a simple example, we illustrate how an individual entity in a certain role is modeled by way of two COAR short-term concepts. The descriptions shown in Figure 7 concern a bar, currently used as a propeller blade, which is connected to a bolt (like one of the bars of the propeller in Figure 4). In this example, the short-term concept bar-1 is an instance of the object type 3-HOLE-BAR. A connection relation is established from bar-1 to the short-term concept bolt-1. Via the role-of link, the short-term concept airplanepart-1 is related to bar-1. That is, airplanepart-1 and bar-1 represent the same geometric object where the short-term concept airplanepart-1 conceptualizes the object according to its function in the evolving airplane. Currently, airplanepart-1 is conceptualized as instance of the role type PROPELLERBLADE. This classification w.r.t. the role-type hierarchy corresponds to the object's current aggregate context as part of a propeller and could change if the object would be detached and used in a different assembly.

All short-term concepts in dynamic conceptualization are *grounded* in the the task environment, as given by the geometric scene description. In general, short-term concepts directly represent some associated geometric entity, for example, an object, aggregate, or connection port. As opposed to primitive objects and connection ports, geometric aggregate representations (grouped structures) do not exist from the beginning but are created and deleted dynamically whenever corresponding short-term concepts are. In some cases, short-term concepts do not directly represent geometric entities. These short-term concepts, however, have grounded representations as parts. In our current system, the only such indirectly grounded short-term concepts represent holes. At this time, holes do not exist in the geometry models. Instead, the two openings of holes (only tunnel-like holes occur in our scenario) are marked in the geometry models to distinguish between the sides from which some object can be fitted into a hole.

Being grounded in the geometric scene description, short-term concepts can access the spatial knowledge contained therein. For instance, geometry models of objects and aggregates give their current positions and orientations, or the length of the bounding box can be used to compute their approximate sizes. Furthermore, as described before, short-term concepts can infer spatial information when needed which thus need not explicitly be represented in them.

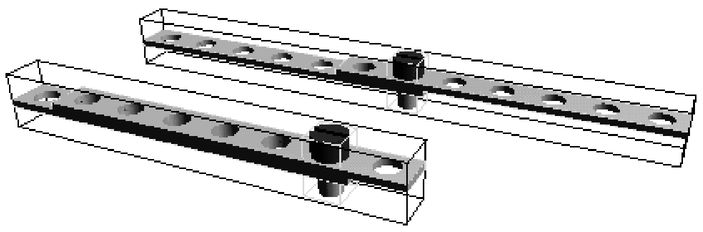
Figure 8. Geometric aggregate representations shown with their bounding boxes

As an example, Figure 8 illustrates how spatial knowledge can be extracted from the geometry scene. It shows two geometric aggregate representations, visualized by their bounding boxes, which were created after the existence of corresponding short-term concepts was inferred. By accessing these geometric representations, it can be inferred that one of the aggregates is in front of the other one and that they are in parallel orientation. By measuring their bounding boxes, it can be furthermore inferred that one of the two aggregates, which both consist of the same parts arranged differently, is larger than the other one.

As short-term concepts are grounded representations of objects and aggregates in the task environment, changes in the environment will be reflected in the short-term concept structure. For example, dynamic conceptualization involves the creation of new short-term concepts, when aggregates are assembled, or role-type instantiation of short-term concepts when objects are used as functional parts in the target assembly. These issues are further described in the next section.

4. Dynamics of Short-term Concepts

As complex aggregates are assembled, or are taken apart again, the constructor's internal view of the environment must be updated accordingly. This is at the heart of dynamic conceptualization. Inferences on short-term concepts concern the assertion or retraction of representations for constructed aggregates in a dynamic knowledge base, and the computation of role changes for individual objects combined herewith. Updates of short-term concepts, usually, will cause updates of other, related short-term concepts as well. For example, a connection relation introduced between two object representations implies the existence of some aggregate representation that has the former object representations as parts. Or becoming part of a structured assembly group determines an object's role-type classification and, hence, a reclassification should be triggered. To capture such dependencies, seven *coherence conditions* over the states of related short-term concepts are formulated below. Altogether these coherence conditions, whose validity is ensured by appropriate inference rules, define the framework for dynamic conceptualization in the assembly domain.

4.1. Role-Type Instantiation

In most cases, objects will assume different roles in the course of assemblage. The possible role types according to which a short-term concept can be classified are restricted by the role-of relations between role types and objects types in long-term memory:

(1) Restriction of role-type classification through role-of relation
The role-type classification of a short-term concept with object type classification O is always one of the role types R_1, ..., R_n, or a generalization thereof, where each of the R_i is related to O, or a generalization of O, via the role-of construct.

For example, in our reference situation a short-term concept representing a bolt will at all times be classified w.r.t. the role-type hierarchy as PROPELLERHUB or AXLE, or (general) as AIRPLANEPART. But it will never have the role type PROPELLERBLADE since only bars can assume this role.

In particular, a short-term concept will be assigned a role type when it is used as part of an assembly group. The classification of a short-term concept w.r.t. the role type hierarchy is tied to its aggregate context by the second coherence condition:

(2) Role-type classification according to aggregate context
If some short-term concept a is the part *partname* of a short-term concept b where b is an instance of the long-term concept B, and if the definition of B requires its part *partname* to be filled with an instance of A then a must be an instance of A.

For example, if a short-term concept representing a bolt is the part "has-propellerhub" of a propeller, then it must be an instance of the concept PROPELLERHUB (cf. the definition of PROPELLER given in Figure 5). Role-type instantiation, in general, will fill the instance-of slot of a short-term concept with a certain long-term concept and, by inheritance from that long-term concept, will also add further attributes to the short-term concept.

4.2. Assertion and Retraction of Connection Relations

In an assembly environment, the connection relation is of central importance. The mechanical objects considered in our scenario have connection ports that allow them to be fitted together in various ways. If an instruction states that two objects are to be connected, then appropriate connection ports are determined which are the put in the connection relation. If two entities are connected to each other and if both of these entities are part of composite objects, then these composite objects themselves are connected as well. Thus, the third coherence condition grounds connections between composites on connections between their parts:

(3) Assertion and retraction of connection relations
Connection relations between composites can only be asserted if there is a connection relation asserted between some of their parts.

For example, a connection relation between the short-term concepts for a propeller and a fuselage can be asserted when there is already a connection relation asserted between the representations for the propeller hub (which is a part of the propeller) and the motor block (which is a part of the fuselage). Ultimately, all connection relations between short-term concepts are based on connections between representations of the objects' connection ports. Whether a connection relation should be asserted between port representations is tested in the geometry scene. Once a connection relation is asserted between two port representations, this information is propagated in the short-term concepts' part-of structure where dependent connection relations are asserted. Similarly, the retraction of a connection relation between port representations will cause the retraction of induced (propagated) connection relations. That is, the connection structure of the short-term concepts, and therefore their aggregate and role-type structure, are grounded in the geometry scene.

4.3. Creation and Deletion of Aggregate Representations

When two or more mechanical objects are assembled, they jointly form an aggregate which is to be represented by some short-term concept. Such short-term concepts are dynamically created and deleted during assemblage. The creation of new short-term concepts for assembled aggregates, or the deletion of aggregates representations whose parts are disconnected, respectively, is ensured by the following coherence conditions:

(4) Creation of aggregate representations
If a connection relation is established between two short-term concepts then there also exists some aggregate representation that has the former two short-term concepts as direct or indirect parts.

(5) Deletion of aggregate representations
An aggregate representation can only exist in short-term memory as long as all its parts are directly or indirectly connected to each other.

4.4. Creation and Deletion of Assembly-Group Representations

Assemblies built from the mechanical objects might form, or contain as subassemblies, structured (named) assembly groups of the target aggregate. When such an assembly group has been constructed in the geometry scene, a corresponding short-term concept is to be created. In order to decide whether or not some assembly group has been constructed, a relation *composes* between a short-term concept a representing some unstructured aggregate and a long-term concept B of a structured assembly group is defined: *a composes B*, if a has all the parts required in the definition of B, and further, all the required relations in the definition of B, as specified by the pp-constraints of B, hold between the parts of a.

The following coherence condition ensures that a maximal number of short-term concepts representing structured assembly groups are created:

(6) Creation of assembly group representations
There is no unstructured aggregate representation in short-term memory that *composes* some long-term concept of a structured assembly group.

Instances of assembly groups must be deleted when the necessary conditions for their existence no longer hold:

(7) Deletion of assembly group representations
An instance of a long-term concept A describing a structured assembly group can only exist in short-term memory as long as also all the parts required in the definition of A exist and are in the required relationship to each other, as specified by the pp-constraints of A.

The technique we employ to infer the existence of assembly-group instances is a variant of compositional inferencing (Padgham & Lambrix, 1994). Our methods differ from those introduced in (Padgham & Lambrix, 1994) in several ways. Most importantly, the creation of multi-level part-of hierarchies in short-term memory is possible, i.e. short-term concepts representing assembly groups may themselves have representations of subassemblies as parts. Furthermore, our framework allows to have changing role-type classifications of short-term concepts as their aggregate context changes.

4.5. Example

The following example illustrates the updates of short-term concepts in response to changes in the assembly environment. These updates are brought about by inference rules that ensure the validity of the above coherence conditions. On the left of the Figures 9-11, the current state of an assembly is presented as a visualization of the geometry scene. On the right, the short-term concept structure representing the geometry scene is shown in a graphical notation. In the graphical notation, short-term concepts are presented as boxes, part relations as arrows, and connection relations as curved lines. For presentation purposes, pairs of short-term concepts representing the same object are displayed in one box. Port representations are not shown.

In the first situation (cf. Figure 9), four building blocks are present on the virtual assembly workbench. Three of these, the two bars and the bolt, are already connected. According the above coherence condition (4), there is an aggregate representation that has these three objects as parts. The aggregate cannot be interpreted as a structured assembly group of the model airplane and is therefore unstructured. Also, none of the role types of the four objects is determined at this stage.

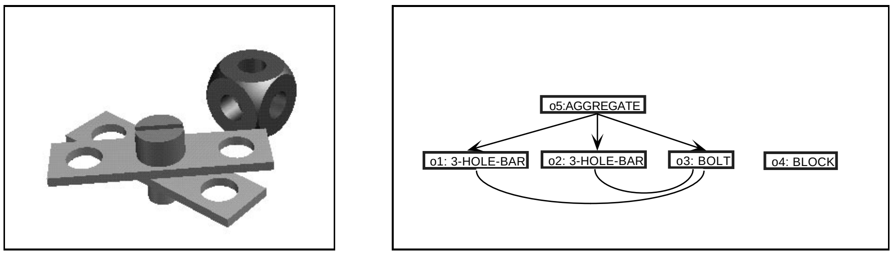

Figure 9. Assembly of a propeller (first situation)

In the following assembly step (cf. Figure 10), the bolt is fitted into the block, e.g. by the instruction: *"attach the bolt to the block"*. As after each assembly step, the system checks in the geometry scene if new connections between port representations have resulted. Assembly instructions cannot be translated directly into connection relations because, due to "physical" constraints, assembly operations can be subject to failures and side effects (e.g., more than one connection might result from one assembly step). In this situation, the system detects one new connection relation between the opening of the block's front hole and the shaft of the bolt and, accordingly, a new connection relation is asserted between the respective port representations (not shown in the graphs). This relation, according to the coherence condition (3), is then propagated in the short-term concepts' part-of structure and, eventually, a connection relation is asserted between the short-term concepts o4 and o3, representing the block and the bolt, respectively. According to coherence condition (4), the block and the bolt must both be part of the same aggregate, so the short-term concept o4 is added as a new part to the aggregate o5. It is then checked, according to coherence condition (6), if the aggregate now contains an assembly group. Since it does not, the short-term concepts are no further restructured.

In the next situation (cf. Figure 11), the two bars are brought into orthogonal orientation, e.g. by the instruction: *"rotate the front bar crosswise to the other bar"*. Rotation operations do not cause new connections nor do they detach previous connections. Yet they can induce

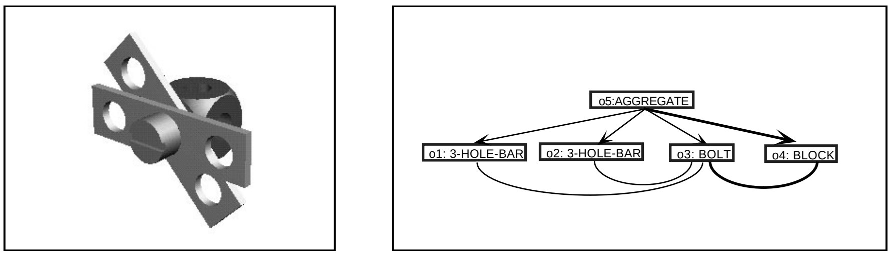

Figure 10. Assembly of a propeller (second situation)

new assembly groups, or destroy existing assembly groups. In the resulting situation, the system has to check, according to coherence condition (6), if the unstructured aggregate o5 composes some long-term concept of a structured assembly group. It does, in fact, compose the PROPELLER since it contains all of its parts in the required relationship. Therefore, a new short-term concept o6 is created as instance of the PROPELLER concept and inserted into the part hierarchy as part of o5, and with o1, o2, and o3 as parts. Then, according to coherence condition (3), the connection relation between the block o4 and the bolt o3 is propagated and a new connection relation between o4 and o6 is asserted. Finally, according to coherence condition (2), the role types of the propeller's parts are assigned: o1 and o2 are instantiated as PROPELLERBLADE, and o3 is instantiated as PROPELLERHUB.

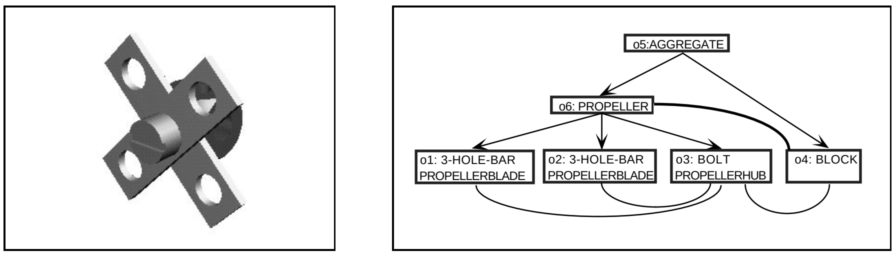

Figure 11. Assembly of a propeller (final situation)

Similarly, the above coherence conditions ensure updates of short-term concepts in all other situations where changes were made in the geometry scene. After any assembly step, the following operations are carried out:

– new connection relations between ports representations are extracted from the geometry scene and propagated in the part-of structure

– new unstructured aggregate representations are created and matched against the long-term concepts of assembly groups, and

– role-type classifications of short-term concepts are assigned according to their aggregate context.

After disassembly operations, connection relations are retracted and aggregate representations are deleted, whereby new aggregate representations could be created in case an aggregate is split. In response to rotation operations, assembly-group representations could either be created or deleted.

5. Conclusions and Future Work

In this paper, we introduced the notion of dynamic conceptualization and stressed its relevance for situated artificial communicators. Dynamic conceptualization is concerned with the fact that an agent's internal representation of the current situation must be adapted as its task environment changes. We have developed the knowledge representation language COAR which is tailored for the modeling of objects, assemblies, and roles in mechanical assembly. The language provides means to describe concepts by their attributes and to relate different concepts to one another by a small set of structural relations. COAR representations integrate taxonomic, partonomic, and functional information of objects and aggregates. Object types, used to model mechanical objects, and role types, used to model the specific functional aspects of objects in larger assemblies, are linked through a special semantic relation "role-of". Furthermore, geometric constraints can be integrated in concept definitions.

We also presented a framework for reasoning with COAR representations that further specifies dynamic conceptualization. Inferencing includes the creation and deletion of assembly representations and the reclassification of representations w.r.t. the role-type hierarchy depending on their changing aggregate context. Short-term concepts, serving as temporal conceptualizations of individual entities (objects and aggregates) in the current task environment, account for the classification of entities with respect to object type and role type. Whereas object-type classification persists, role-type classification is dynamic, i.e. it may change and evoke changing feature attribution. Being grounded in the task environment by way of the geometric scene description, short-term concepts can access the spatial knowledge contained therein which thus needs not be represented explicitly.

Our approach has so far been evaluated in the CODY Virtual Constructor, a knowledge-based system for the interactive assembly of simple building blocks in a virtual environment. In the Virtual Constructor COAR representations are coupled with the geometric scene description such that the processes of dynamic conceptualization are triggered by changes in the virtual environment. We anticipate several interesting applications for integrated knowledge-based graphics systems of this kind, e.g., in virtual prototyping and assembly planning.

Within SFB 360, the CODY system is currently being integrated in a demonstration system that comprises subsystems for image and speech understanding. We successfully coupled COAR representations with the geometric scene description produced by the vision component. As new input arrives from the vision component, dynamic conceptualization provides additional information about the objects' assembly structure and functional roles which can then be used in instruction processing. Furthermore, first experiments show that the knowledge about the legal connection structure of mechanical objects, defined by our system, can be used to correct physically impossible object configurations that are sometimes produced by the vision component due to camera tolerances (cf. Figure 12).

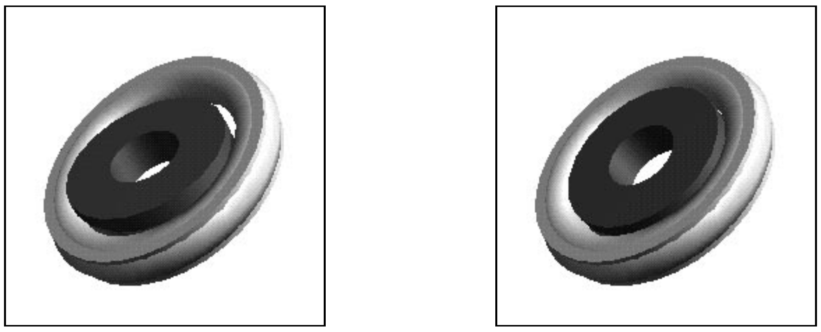
Figure 12. Raw reconstruction of a wheel as produced by the vision system (left) and its model-driven elaboration (right)

As a future extension of our knowledge representation formalism, we plan to add another, imaginal kind of representation to the structured representations we developed so far. These *imaginal prototypes*, that we already briefly described in Section 3 (cf. Figure 6), are parametric spatial representations of objects and aggregates whose realization will build on constructive solid geometry. They are useful, for example, to describe the generic shape of artifacts. Also, other space-related features, like intrinsic orientations, can be defined in imaginal prototypes. Like the structural descriptions of long-term concepts described in this paper, imaginal prototypes will be organized in a specialization hierarchy such that generic information can be dynamically inherited to all geometric instances in the mechanical-object assembly environment.

Acknowledgements

The CODY Project is part C1 of the Collaborative Research Centre *"Situated Artificial Communicators"* (see site description by Rickheit & Wachsmuth, this volume). Research in the CODY Project is partly supported by the German National Science Foundation (DFG). The authors are indebted to Britta Lenzmann, Martin Hoffhenke, Christian Pütter, and Stefan Kopp for various contributions to the research.

Notes

[1] A running demonstration of the virtual assembly workbench can be seen in our contribution to the IJCAI-95 Videotape Program, cf. (Cao et al., 1995).

[2] Some readers may still want to refer to this "false" propeller as a propeller. This is not the point in our current work. Our point is to provide means for recognizing a propeller that is correctly built according to its definition. Should the right-hand-side assemblage in Fig. 4 be recognizable as a propeller, we could allow for an according "slack" in the routine which evaluates orthogonality. Yet another point is how reference could be established to the right-hand-side assemblage by a description like "the propeller on the right". This topic concerns future work.

References

Y. Cao, B. Jung, and I. Wachsmuth: 1995. Situated verbal interaction in virtual design and assembly, IJCAI-95 Videotape Program. Abstract in *Proc. 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, pages 2061–2062.

G.A. Fink, F. Kummert, and G. Sagerer: 1994. A close high-level interaction scheme for recognition and interpretation of speech. In *International Conference on Spoken Language Processing*, volume 4, Yokohama, Japan, pages 2183–2186.

Thomas Fuhr, Gudrun Socher, Christian Scheering, and Gerhard Sagerer: 1995. A three-dimensional spatial model for the interpretation of image data. In *IJCAI-95 Workshop on Representation and Processing of Spatial Expressions*, 14th Int. Joint Conf. on Art. Int. (IJCAI-95), Montréal, Canada, pages 93–102.

M. Genesereth and N. Nilsson: 1987. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, Los Altos, CA.

Th. Gruber and G. Olsen: 1994. An ontology for engineering mathematics. In *Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, San Francisco, CA, pages 258–269.

N. Guarino, M. Carrara, and P. Giaretta: 1994. An ontology of meta-level categories. In *Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, San Francisco, CA, pages 270–280.

B. Jung, B. Lenzmann, and I. Wachsmuth: 1995. Interaktive Montage-Simulation mit wissensbasierter Grafik. SFB 360 Report 95/6, Universität Bielefeld.

B. Jung and I. Wachsmuth: 1994. Dynamische Konzeptualisierung. SFB 360 Report 94/9, Universität Bielefeld.

B. Jung and I. Wachsmuth: 1995. Integrating spatial and conceptual knowledge in virtual assembly. In L. Dreschler-Fischer and S. Pribbenow, editors, *KI-95 Activities: Workshops, Posters, Demos.* GI, pages 103–104.

R. Moratz, H.J. Eikmeyer, B. Hildebrandt, F. Kummert, G. Rickheit, and G. Sagerer: 1995. Integrating speech and selective visual perception using a semantic network. In *1995 AAAI Fall Symposium on Computational Models for Integrating Language and Vision*, Cambridge, MA, (to appear).

H. Niemann, G. Sagerer, S. Schröder, and F. Kummert: 1990. ERNEST: A semantic network for pattern understanding. *IEEE Transactions on Pattern Analysis*, 12(9), pages 883–903.

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

G. Rickheit and H. Strohner: 1994. Kognitive Grundlagen situierter künstlicher Kommunikatoren. In H.-J. Kornadt, J. Grabowski, and R. Mangold-Allwin, editors, *Sprache und Kognition: Perspektiven moderner Sprachpsychologie.* Spektrum Akademischer Verlag, Heidelberg, pages 73–92.

G. Rickheit and I. Wachsmuth: 1996. Collaborative Research Centre "Situated Artificial Communicators" at the University of Bielefeld. *This volume 10(3-4).*

G. Sagerer: 1990. *Automatisches Verstehen gesprochener Sprache.* Bibliographisches Institut, Mannheim.

J. Sowa: 1988. Using a lexicon of canonical graphs in a semantic interpreter. In M. W. Evens, editor, *Relational Models of the Lexicon.* Cambridge University Press, Cambridge, UK, pages 112–137.

E. C. Way: 1991. *Knowledge Representation and Metaphor.* Kluwer, Dordrecht.

Author's Vitae

I. Wachsmuth
Ipke Wachsmuth has held the chair of Knowledge-Based Systems/Artificial Intelligence in the Faculty of Technology at the University of Bielefeld, Germany for the past seven years. His academic background is in mathematics and computer science. He holds a masters degree in mathematics (Dipl.-Math.), obtained in 1975 from the Technical University of Hanover, Germany, where he also obtained his Ph.D. (Dr.rer.nat) in 1980 for research in cellular automata synchronization, and a Habilitation degree in Computer Science which he obtained from the University of Osnabrück, Germany, in 1989, for research on knowledge base organization. Before coming to Bielefeld, he held faculty and project leader positions in the Department of Mathematics/Computer Science and the Linguistics Department at the University of Osnabrück, Germany. He was also assistant professor in the Department of Mathematical

Sciences at Northern Illinois University in 1981-83 and research fellow in the LILOG group at IBM Germany, Stuttgart in 1986-88. Prof. Wachsmuth has a strong multidisciplinary commitment and has published in the fields of cellular automata, cognitive learning research, intelligent tutoring systems, natural language understanding, technology assessment of AI, expert systems, and large knowledge bases. A sabbatical leave from the University of Bielefeld in 1992/93, spent with the Scientific Visualization Group at the German National Research Center for Information Technology (GMD) at Bonn-Sankt Augustin, led him into computer graphics and virtual reality. His current research activities cover virtual environments, dynamic knowledge representations and multi-agent techniques in the context of human-machine communication. Among many other professional services, Ipke Wachsmuth chaired the 19th Annual German Conference on Artificial Intelligence in 1995 and was recently re-elected to the executive board of the Collaborative Research Centre SFB 360 *"Situated Artificial Communicators"*.

B. Jung
Bernhard Jung received his Diplom (MSc) in Computer Science in 1992 from his hometown University of Stuttgart, in southern Germany. His academic background further includes a minor in computational linguistics and, at the University of Missouri, St. Louis, USA, the general study of philosophy. At present, he is employed with the Collaborative Research Centre SFB 360 *"Situated Artificial Communicators"* at the University of Bielefeld where he works for the CODY project and pursues his doctorate (Ph.D.) in Artificial Intelligence. His current research generally concerns knowledge processing in situated agents and specifically addresses the integration of structural and spatial knowledge representation and reasoning. Prior research activities lie in machine learning (University of Stuttgart), natural language processing (LILOG Project, IBM Germany), and model-based configuration and failure analysis (FAW Research Institute for Applied Knowledge Processing, Ulm, Germany). Bernhard Jung's other interests include multi-agent systems, virtual reality, multimedia, and cognitive systems.

Address for correspondence: {ipke,jung}@techfak.uni-bielefeld.de