Integration of Geometric and Conceptual Reasoning for Interacting with Virtual Environments

Bernhard Jung and **Ipke Wachsmuth**
Faculty of Technology, Knowledge-Based Systems Group
University of Bielefeld
PO Box 10 01 31
D-33501 Bielefeld, Germany
Email: {jung,ipke}@techfak.uni-bielefeld.de

Abstract

This paper describes the knowledge processing in the CODY Virtual Constructor, an operational system enabling the interactive assembly of complex aggregates in a virtual environment. Two forms of reasoners are used: a geometric reasoner that infers spatial properties of scene objects and a conceptual reasoner that keeps track of the evolving aggregate's assembly structure. The combination of the two reasoners enables the system both to simulate assembly processes in the virtual environment and to understand natural language instructions. By maximizing the mutual exchange of information between the reasoners, additional knowledge can be inferred that not only improves understanding of language instructions but also increases efficiency of inferencing.

Virtual Assembly by Natural Language Instruction

The *CODY Virtual Constructor* is a knowledge-based system for the interactive assembly of complex aggregates on a virtual assembly bench (Cao, Jung, & Wachsmuth 1995). In our testbed scenario, the user can assemble a toy airplane and similar constructs from parts of the *Baufix* construction kit, such as bolts, blocks, and bars (see Figure 1).

One way to assemble or disassemble parts is by direct manipulation using the mouse or similar input devices: The user selects an object and moves it in close proximity of another object; a knowledge-based snapping process will then complete the fitting. Alternatively, the user may instruct the system in natural language, e.g., *"Insert the long bolt into the top of the airplane"* (see Figure 1). As this example shows, instructions may build, on the one hand, on spatial properties of objects (*"long", "the top of"*) in the assembly environment and, on the other hand, on the concept-based understanding of grouped structures (*"airplane"*) in the developing construct. Accordingly, two forms of reasoners have been developed: a *geometric reasoner* that infers spatial properties of scene objects and a *conceptual reasoner* that keeps track of the assembly structure of the evolving aggregate.

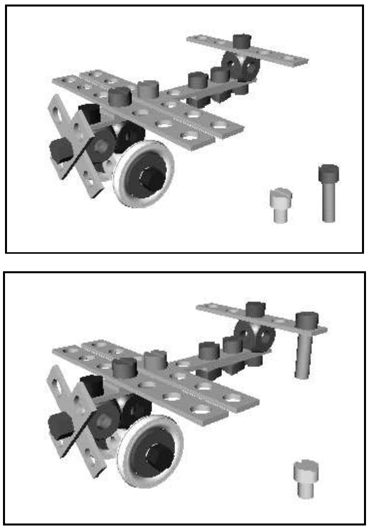

Figure 1: Both geometric and conceptual reasoning is required to process the instruction *"Insert the long bolt into the top of the airplane."*

In the remainder of this paper, we first present the geometric and the conceptual reasoner, in separate, focusing on the kind of knowledge each system is particularly suited to represent. We then describe what information is exchanged between the two reasoners, when this is done (when-changed vs. when-needed) and the additional knowledge gained by doing so. Later we point out that only the combination of geometric and conceptual reasoning but not a single reasoning mode alone achieves the simulation of assembly operations in the virtual environment. Also, experimental results are presented on how a maximized communication with the geometric reasoner can speed up inferencing in the conceptual reasoner. Finally, we conclude and discuss our results.

Geometric Reasoner

The geometric reasoner maintains knowledge about the scene objects' spatial properties including primitive features such as position, orientation, shape and size as well as several spatial relations derivable from these primitive features. Geometric reasoning is used for assembly simulation, processing of natural language instructions, and, as described below, to support inferencing in the conceptual reasoner.

Generic knowledge about scene objects includes their wire frame models (exact shape), bounding boxes (shape approximation), center of gravity, prototypical orientation, as well as positions and orientations of the objects' connection ports (e.g. shafts and holes).

Current knowledge about scene objects and a camera model representing the viewpoint of the user is maintained in a data-structure called "geometric scene description". By only explicitly storing the current position and orientation of objects (and aggregates, see below) and camera, the geometric reasoner can infer the following additional spatial properties: position and orientation relative to the user, and relative to each other; several qualitative spatial relations between scene objects, such as $\mathsf{parallel}_x$, $\mathsf{parallel}_y$, $\mathsf{parallel}_z$, $\mathsf{orthogonal}_x$, $\mathsf{orthogonal}_y$, $\mathsf{orthogonal}_z$, near, touches, and connection; further size and approximate shape. These spatial properties are only computed on demand but not explicitly stored; thus, computational costs for updating them are avoided.

The geometric reasoner also performs *collision avoidance* to exclude physically impossible interpenetrations of scene objects.

Conceptual Reasoner

A main purpose of the conceptual reasoner is to keep track of the evolving aggregate's assembly group structure and changing functional roles of parts, in short: to *dynamically conceptualize* the changing situation in the assembly environment (Wachsmuth & Jung 1996). Like geometric reasoning, conceptual reasoning is used to support both assembly simulation and natural language processing.

A frame-based representation language – COAR ("Concepts for Objects, Assemblies, and Roles") – has been developed as basis for dynamic conceptualization of the virtual environment (Jung 1997), (Wachsmuth & Jung 1996). COAR's language constructs for assembly modeling resemble the semantic network language ERNEST (Kummert *et al.* 1993), and the terminological language for part-of hierarchies introduced in (Padgham & Lambrix 1994). Based on a similar distinction in CONCEPTUAL GRAPHS (Sowa 1988), COAR further differentiates concepts into *object types* that are used to model multi-functional objects and assemblies and *role types* that model specific functional aspects of objects as components of larger assemblies. In the Baufix airplane, for example, screws can assume the role of an axle and blocks can assume the role of an undercarriageblock. The idea behind this separation is that object representations can change their role type (and are assigned changing attribute sets) according to their aggregate context.

```
Long-Term Concept: UNDERCARRIAGE
  is-a: ASSEMBLYGROUP
  part has-left-halfaxlesys #1: HALFAXLESYSTEM
  part has-right-halfaxlesys #1: HALFAXLESYSTEM
  part has-block #1: UNDERCARRIAGEBLOCK
  pp-constraint connection ⟨has-block⟩ ⟨has-left-halfaxlesys⟩
  pp-constraint connection ⟨has-block⟩ ⟨has-right-halfaxlesys⟩
  pp-constraint parallel_x⟨has-left-halfaxlesys⟩
                          ⟨has-right-halfaxlesys⟩
```

Figure 2: COAR-definition of Baufix airplane's UNDERCARRIAGE. Three parts are required, two of which with object type HALFAXLESYSTEM (an assembly group consisting of five parts), and one with role type UNDERCARRIAGEBLOCK (instances of object type BLOCK are reclassified as such when used as component of an undercarriage). Further, the parts are required to be pairwise connected. The geometric constraint $\mathsf{parallel}_x$ requires parallelism of halfaxlesystems.

In COAR, concepts for assembly groups are defined by their parts and part-part-(pp-)constraints describing necessary relations between them (negative constraints decribe prohibited relations). Constraints may either be connection-constraints which require, when tested, corresponding relations to be asserted between COAR-representations of individual objects, or, *geometric* constraints, e.g., $\mathsf{parallel}_x$ that trigger tests in the geometry models directly on an when-needed basis.

Inferences over COAR representations include *aggregation* by which connected objectes are grouped to unstructured aggregates; *aggregate conceptualization*, by which unstructured aggregates are recognized as structured subassemblies of the target aggregate; and *role assignment*, by which parts are reclassified w.r.t. the underlying role type hierarchy according to their use in larger assemblies.

Information Exchange between Geometric and Conceptual Reasoner

In the Virtual Constructor, information exchange between reasoner is bi-directional: The conceptual reasoner may access information from the geometric rea-

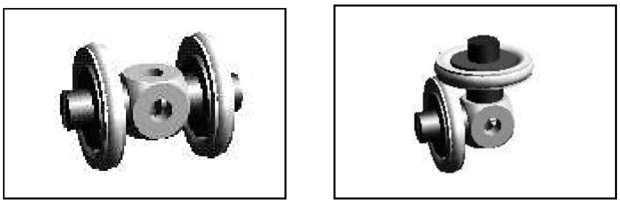

Figure 3: Aggregate conceptualization is an inference that recognizes the aggregate to the left as instance of the concept UNDERCARRIAGE (see Figure 2). The aggregate to the right is no instance, since the geometric constraint $\mathsf{parallel}_x$ is violated.

soner and vice versa. Exchange of knowledge between the reasoners occurs for two reasons: (1) Many aspects of the enviroment are represented in one reasoner only. Even if some of these aspects could, in principle, be represented in both reasoners they are not so as to avoid the adminstration overhead associated with maintaining multiple representations. If such aspects can be used to guide inferencing in the other reasoner, they are exchanged on a when-needed basis. (2) Other aspects of the environment are represented in both reasoners. If they change, measures must be taken to keep the representations synchronized; in this case, the two reasoners inform each other on a when-changed basis.

Information flow from geometric to conceptual reasoner. When the geometric reasoner detects new connection relations between objects (due to assembly operations in the virtual environment), or detects the invalidity of previously valid connection relations (due to dissassembly operations), or other changes of object geometry that may affect assembly-group representations in the conceptual level (e.g. rotation operations may invalidate $\mathsf{parallel}_x$ or similar constraints required in the definition of assembly groups), this information is supplied to the conceptual reasoner on a when-changed basis. Similarly, after assembly or disassembly operations, the new amount of consumed capacities of the objects' connection ports are calculated in the geometry scene and stored in the conceptual representations immediately. When-needed knowledge is supplied to the conceptual reasoner when aggregate conceptualization requires testing of geometric constraints.

Information flow from conceptual to geometric reasoner. Existence or non-existence of aggregates is inferred by the conceptual reasoner. Knowledge about the assembly structure is propagated back to the geometric reasoner on a when changed-basis.

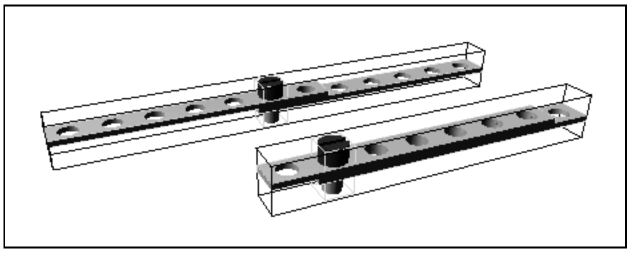

Figure 4: By creating geometric aggregate representations (indicated by bounding boxes), spatial properties of aggregates can be inferred when needed. Here, the system can infer, e.g., that the aggregate *in front* is *smaller* than the other.

By creating aggregate representation in the geometric reasoner, spatial knowledge (such as position, orientation, and size) about aggregates can be easily inferred that would be difficult to compute in the conceptual reasoner and, thus the representational power of the combined system is improved. For example, Figure 4 shows two aggregates that consist of identical but differently arranged parts (one is "folded in", the other "folded out"). An adequate calculation of the relative size of these aggregates is not possible over COAR-representations and, in general, would require – if at all posssible – very complex object descriptions using qualitative relations only. However, by measuring the (diameter of the) aggregates' bounding boxes, the geometric reasoner can easily infer which aggregate is larger than the other. Geometric aggregate representations are further exploited in conceptual reasoning for testing of geometric constraints between assembly groups.

Assembly Simulation

The CODY Virtual Constructor supports the simulation of various assembly-related operations in the virtual environment, such as assembly and dissassembly of parts or rotation of subassemblies w.r.t. other parts in a larger aggregate. In the following, we give a detailed example showing step-by-step the various stages involved when simulating assembly operations. The example demonstrates, that a combination of geometric and conceptual reasoning is necessary for collision-free part mating.

The original situation of the example is shown in Figure 5. There are two aggregates in the environment and the system is instructed: *"Insert the aggregate into the bar"*. The first step of the assembly simulation involves the selection of concrete objects and their connection ports matching the natural language instruction. By using conceptual knowledge alone, the

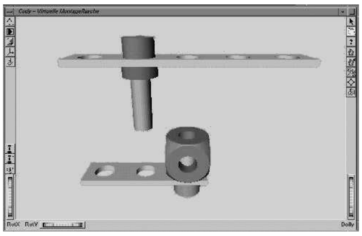

Figure 5: Original situation and first step of assembly example: The system is instructed: *"Insert the aggregate into the bar"*. The systems selects suitable objects and connection ports for the mating operation.

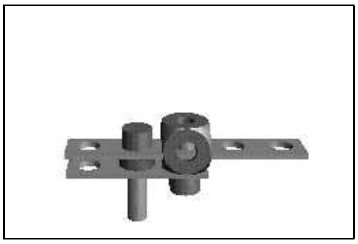

Figure 6: Second step of assembly example (not visualized by system): Parts are mated by only using information about current capacities of ports stored in conceptual representations. This results in a collision between the upper bar and the cube.

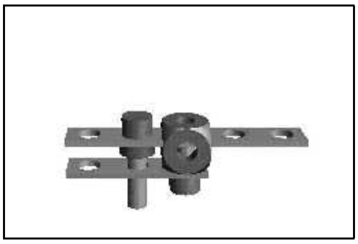

Figure 7: Third step of assembly example (not visualized by system): Collision avoidance using exact geometric knowledge. The screw is moved back out of the bar in little steps.

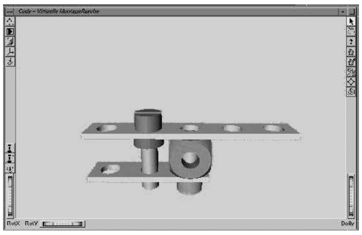

Figure 8: Fourth step of assembly example and resulting situation: The scene is now visualized and the conceptual model is updated.

system infers that the screw of the upper aggregate must be involved in the mating operation. Therefore, the expression *"bar"* must refer to the bar of the lower aggregate. There are, however, two unoccupied holes in the bar and it is also unspecified whether to insert the screw from above or from below. In this case, the system chooses to insert the screw from above into the middle hole of the bar.

The second step of the assembly example is shown in Figure 6 (this step is not visualized by the system) The parts are mated based on the currently available capacities of their connection ports, whose values are stored in the COAR-represenations of the conceptual reasoner. Also, a default assumption is made, in which orientation the upper aggregate is inserted into the bar. As result of this step, two objects interpenetrate each other in a physically impossible way.

The third step in assembly simulation is collision avoidance. In Figure 7, the screw is moved back out the bar in little steps until a collision-free state is reached. Collision avoidance operates over geometric object representations that provide exact, numeric information about object locations and shape.

In the fourth step of the assembly example, the resulting state is reached and the scene is visualized (Figure 8). The geometric reasoner then tests if and how many new connection relations between objects have resulted from the assembly operation. New connection relations are asserted in the COAR-representations of scene objects and other conceptual reasoning such as aggregate conceptualization is triggered.

Improving the efficiency of conceptual reasoning using geometric constraints

Highly interactive applications such as virtual environments demand fast system replies. Unfortunateley, aggregate conceptualization (which infers the existence of wholes based on the existence of the required parts that are in the required relationships) is NP-complete (Jung 1997). Actual running time of aggregate conceptualization, however, depends on (a) how many parts need to be considered as possible components of an assembly group and, (b) the degree of constrained-ness of the assembly group's definition. The latter property implies that running time of aggregate conceptualization can be reduced by adding (geometric) constraints to the COAR-definitions of assembly groups.

Experiments with the COAR-definitions of the Baufix airplane's (see Figure 1) assembly groups were carried out. Specifically, the number of geometric constraints in the COAR-definition of one particularly computationally expensive concept, FUSELAGE, was varied (concepts for other assembly groups did not

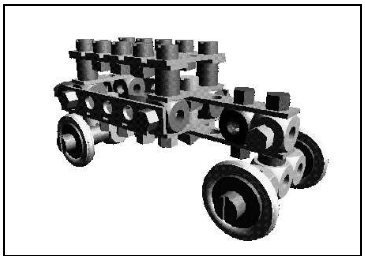

Figure 9: Baufix bus (124 parts).

cause combinatorical explosion during aggregate conceptualization). In the first knowledge base, FUSELAGE was modeled using all possible connection constraints. In a second knowledge base, thirteen more positive geometric constraints were added to the concept definition. An interesting aspect of these additional geometric constraints is that most of them are redundant in the sense that adding them to the concept description does not exclude any more Baufix assemblies from being instances of FUSELAGE. In a third knowledge base, sixteen further *negative*, again redundant constraints were added to the concept description of the second knowledge base. Finally, in a forth knowledge base, fourteen negative, also redundant connection constraints were added to the concept description.

Our experiments included the assembly of the Baufix airplane (33 parts) and a Baufix bus (124 parts; the bus contains no instance of FUSELAGE) in the virtual environment. The experiments measured the number of choose-operations of the backtracking-algorithm implementing aggregate conceptualization. The results are summarized in Table 1. They show that the use of as many as possible (most of them redundant) geometric constraints – each of which resulting in a call to the geometric reasoner during conceptual reasoning – resulted in a speed-up of approximately 90%. Using even more (again redundant) negative connection constraints, a total speed-up of up to 98% was achieved. Absolute running times[1] of aggregate conceptualization went down to a maximum of 0.84 seconds per assembly operation in case of the airplane and down to 1.94 seconds in case of the bus.

COAR-concept of FUSELAGE	Assembly of airplane		Assembly of bus	
1. minimal description using 11 positive connection constraints	9581	100%	1170872	100%
2. As 1, plus 13 positive geometric constraints	6652	69.4%	348233	29.7%
3. As 2, plus 16 negative geometric constraints	1331	13.9%	118630	10.1%
4. As 3, plus 14 negative connection constraints	1580	16.5%	25393	2.2%

Table 1: Cost of aggregate conceptualization using differently constrained descriptions of FUSELAGE: Total number of choose-operations in backtracking-algorithm and and relative costs in virtual assembly of Baufix airplane and Baufix bus.

Conclusions and Discussion

To make interaction with virtual environments more intuitive, VR systems of the future must be both knowledgeable and responsive. We have developed an operational system, the CODY Virtual Constructor, that supports the simulation of several assembly-related operations in a virtual environment, such as assembly and disassembly of parts and rotation of subaggregates. A unique feature of the CODY Virtual Constructor is the dynamic conceptualization of the evolving aggregate's assembly structure.

The knowledge processing of the Virtual Constructor comprises both a geometric and a conceptual reasoner. The hybrid approach is necessary to achieve the following system functionalities:

- Natural language processing: Verbal instructions may refer both to spatial properties of scene objects that are inferred by the geometric reasoner and to constructed assemblies and functional roles of objects that are inferred by the conceptual reasoner.
- Assembly simulation: Both abstract, qualitative knowledge describing the parts' connection possibilities and exact, numeric knowledge about object location and shape is necessary to enable collision-free part mating. This is consistent with the *poverty conjecture* of qualitative kinematics: "There is no purely qualitative, general-purpose representation of spatial properties" (Forbus, Nielsen, & Faltings 1991), (Forbus, Nielsen, & Faltings 1987).

Given the necessity of both a geometric and the conceptual reasoner, the question arises how the interaction between the reasoners is organized. Benefits of

[1] On SGI Indigo^2/R4400 platform.

careful balancing and a maximized, two-way communication between the reasoners include:

- Increased representational power: By making geometric knowledge available to the conceptual reasoner, spatial relations can be included into conceptual descriptions of assembly groups. Further, by making available knowledge about constructed aggregates (maintained by the conceptual reasoner) to the geometric reasoner, additional knowledge about the aggregates' spatial properties such as location and size can be inferred.
- Increased efficiency of conceptual reasoning: Computational costs of aggregate conceptualization can be significantly reduced if generic assembly group descriptions include a large number of geometric constraints. Efficiency of reasoning is particularly important considering the high interactivity demands of virtual reality applications.
- Low cost of model maintenance: The geometric reasoner can infer several kinds of spatial relations between scene objects that are of potential interest for conceptual reasoning. However, the amount of spatial relations between object pairs grows over-exponentially with the number of parts and, moreover, due to the dynamic nature of the virtual environment, spatial relations are subject to frequent change. Thus, an explicit assertion of spatial relations in the conceptual representations is not an option and, instead, they are only inferred from the geometry scene when needed. This design choice is in agreement with well-known insights about the advantage of analog/direct over symbolic representations w.r.t. model updating and the related frame problem, e.g. (Barr & Feigenbaum 1981).

In current work, we explore alternatives/ complements to COAR for conceptual reasoning about assemblies. In COAR, assembly group representations are described in terms of their mechanical components and the necessary relations between them. Alternatively, assemblies might be described in terms of their shape only. For example, in the ACRONYM vision system (Brooks 1981), complex shapes are composed of primitive, parametric shapes represented as generalized cylinders. We currently work on similar representations called *imaginal prototypes*. The goal is the development of more general assembly group representations that are independent of any particular construction kit.

Assembly simulation in the CODY Virtual Constructor has so far concentrated on the Baufix construction kit. Besides Baufix, we have also imported and assembled a set of industrial CAD-based parts. In current work, we aim at extending virtual assembly to a variety of other construction kits.

Acknowledgment. The research described here is partly supported by the German National Science Foundation (DFG).

References

Barr, A., and Feigenbaum, E. 1981. *The Handbook of Artificial Intelligence*, volume 1. Los Altos, CA: William Kaufmann.

Brooks, R. 1981. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence* 17:285–348.

Cao, Y.; Jung, B.; and Wachsmuth, I. 1995. Situated verbal interaction in virtual design and assembly, IJCAI-95 Videotape Program. In *Proc. International Joint Conference on Artificial Intelligence*, volume 2, 2061–2062. IJCAI, Morgan Kaufman.

Forbus, K.; Nielsen, P.; and Faltings, B. 1987. Qualitative kinematics: A framework. In *Proc. of the International Joint Conference on Artificial Intelligence*, 430–435. San Francisco: Morgan Kaufmann.

Forbus, K.; Nielsen, P.; and Faltings, B. 1991. Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence* 51(1-3):417–471.

Jung, B. 1997. *Wissensverarbeitung für Montageaufgaben in virtuellen und realen Umgebungen*, volume 157 of *Dissertationen zur Künstlichen Intelligenz.* Sankt Augustin, Germany: infix.

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

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

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

Wachsmuth, I., and Jung, B. 1996. Dynamic conceptualization in a mechanical-object assembly environment. *Artificial Intelligence Review* 10(3-4):345–368.