KB-VISION: A Tool for Graphical Manipulation and Visualization of Domain Models*

Thorsten Liebig and Dieter Finkenzeller and Marko Luther
University of Ulm, Department of Artificial Intelligence,
Faculty of Computer Science, 89075 Ulm, Germany
{liebig, finken, luther}@informatik.uni-ulm.de

Abstract. Even if a problem solving method and a domain ontology has been identified, there still remains the problem of adding sufficient and consistent domain knowledge to a knowledge processing system. Our KB-VISION system supports this knowledge acquisition process by making use of a 3D graphical user interface in which domain objects can be easily created and composed to a domain model. Using object sensitive manipulation options and by evaluating each graphical action in the underlying knowledge representation formalism, the system ensures a consistent domain model. It can also be used as a knowledge-based graphical simulation environment for various reasoning components (e.g. planners, path generators).

1 Introduction

Representation of knowledge is a premise for intelligent behavior in every domain. Even the Turing Test implicitly presumed the ability to represent knowledge about the content of an interrogation. Smith [13] and Newell [10] conclude some years later, that any intelligent system has to incorporate knowledge in some way. This is in accordance with Goldstein and Paperts [6] interpretation of intelligence in AI research. They argue that understanding intelligence is not a question of "identification of a few powerful techniques, but rather the question of how to represent large amounts of knowledge in a fashion that permits their effective use and interaction".

In general, a knowledge-based reasoning system consists of two parts. One part is the represented knowledge —typically called the knowledge base—, and the other part are the reasoning mechanisms. Usually, the knowledge can be split into more general, conceptual knowledge and specific, individual knowledge. The first contains concepts and relations about the application context and is called *concept model* in the following. The latter is called *domain model* and consists of instantiated concepts and relations and represents a concrete and individual situation. Whereas general knowledge is static, the second type is highly dynamical because it has to be frequently adapted to changes in the focussed situation. Even in an environment where the knowledge-based system is able to explore and assert this kind of dynamic knowledge by it self we identified a strong demand for an appropriate system for building and interacting with those kinds of domain models. Therefore, the graphical KB-VISION system was developed [5]. This system can be used without having knowledge about the concrete syntax

*This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG).

of the underlying knowledge representation system or its representation peculiarities. While composing the objects graphically the system tracks the analogous symbolic relationships between the manipulated objects in the corresponding domain model. In order to keep the domain model consistent, the users graphical manipulation options are restricted to those actions the knowledge base has permitted for the currently focused objects. The system can also be used to visualize changes in the domain model initiated by other knowledge manipulating components. In summary, the KB-VISION system can play the role of an intelligent real world simulation environment.

The structure of this paper is as follows. In the next section we illustrate our motivation to build such a tool by shortly introduce our research background. This is followed by a more detailed description of the system components. We will end with related work, a short conclusion and with an outlook about potential application areas of our approach.

2 Application Background

Our work is concerned with the modeling and representation of symbolic knowledge for an autonomous mobile robot acting in an inhomogeneous office environment [12]. Within this framework we distinguish between different kinds of models, namely the concept model and the domain model. The formalism we use to represent those models is a description logic.

The concept model is basically static and represents those objects, relations and axioms on different levels of abstraction which are potentially relevant within the focused robot environment. The domain model consists of instantiated objects and relations of the concept model and represents a concrete environment scenery. This model is dynamic with respect to the task the robot has to perform within our office environment. The domain model changes because of external human or other system activities or because of actions performed by his own. In order to get an overview over the current domain model of the robot without triggering all representation terms and succeeding changes in the model by hand, we use the KB-VISION system to visualize the robots "imagination" of his environment. In this sense KB-VISION is a debugging and verification tool for our symbolic and subsymbolic knowledge processing mechanisms. Whenever a sensing component reports a new or changed object or relation to the knowledge base, this update will appear instantly in the graphical scene. Beyond that, we often have to build up different domain models for varying tasks which work as initial models for the robot. This process is much less time consuming with KB-VISION than before. Now, existing models can be reused by rearranging, deleting or creating additional objects. A new obstacle can be created simply by closing a door or placing a table in the middle of a hall with a few mouse movements, for example.

3 System Architecture

Our mobile robot is designed as a client–server system with the knowledge base serving multiple clients. Clients are knowledge manipulation, input or output components or the formerly mentioned visualization. The architecture of the KB-VISION system, as a fraction of this design, is shown in Figure 1.

The architecture shows three main components: the graphical component, the knowledge base and the network connection. These components are discussed in more detail in the following subsections.

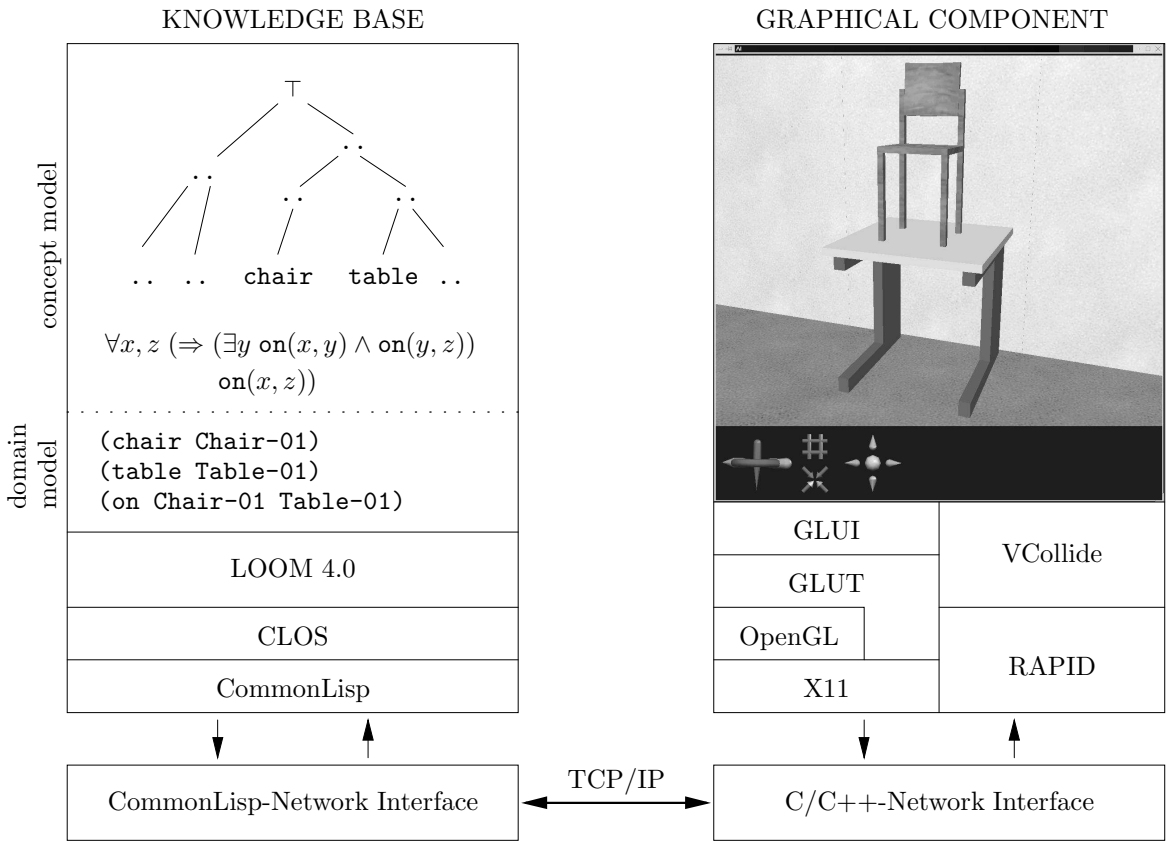

Figure 1: Architecture of the KB-VISION system. The network component connects the knowledge base with the graphical component via TCP/IP.

3.1 Knowledge Base

The knowledge base acts as the central data pool for the robot system as well as the KB-VISION system. As mentioned before, the knowledge base contains more general, basically static knowledge in a model called *concept model* as well as dynamical, situation dependent knowledge in the *domain model*.

The concept model contains all concepts relevant in the broader context of the focussed environment together with common-sense knowledge in the sense of universal rules, heuristics and general physical laws. This includes taxonomical knowledge about different types of objects, their properties and potential relationships among them.

Tables, chairs or doors as well as books, pencils, rooms or halls are objects in a typical office environment. On more abstract levels in the concept hierarchy we distinguish between movable and not movable objects or stackable and not stackable objects, for example. A further specialization differentiates between active stackable and passive stackable objects. Passive stackable objects are those who cannot be stacked on any other object but on which others can be stacked (e.g. floor). An object which can be put on other objects is an active stackable object (e.g. chair, book). A fraction of this object taxonomy is shown in Figure 2.

In the context of an office environment, relevant properties of objects are their position, color etc. as well as their 3D shape model and texture, which are needed by the graphical component for visualization.

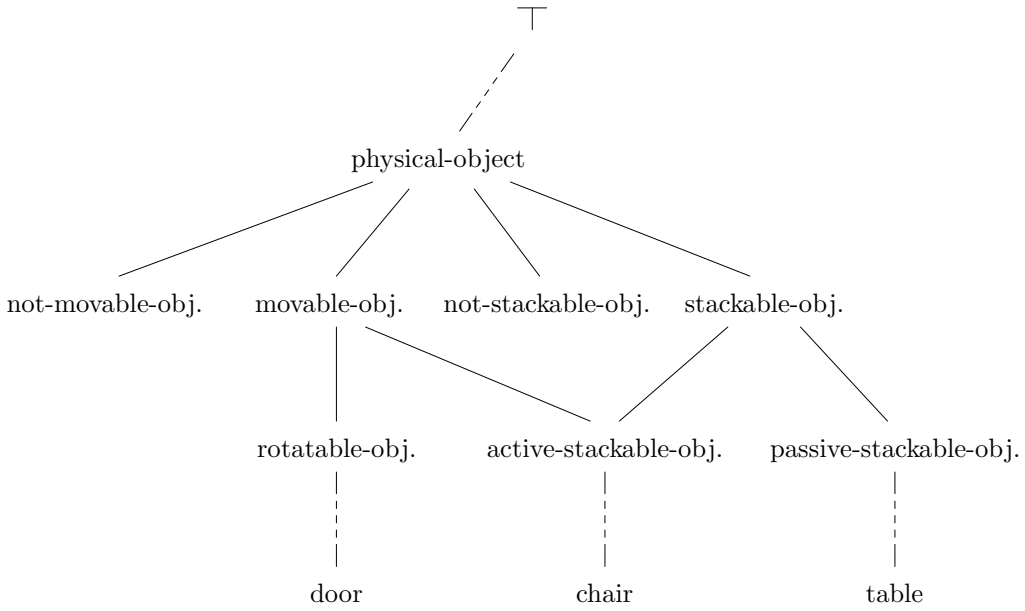

Figure 2: Fraction of the concept model concerning physical objects. (The dashed line indicates missing concept levels.)

Potential relationships between objects cover qualitative spatial aspects as well as facets of mereology (being the theory of the part-whole relationship) and connectivity. Important spatial aspects include topology, orientation and distance. Topology and orientation are qualitative properties with respect to an underlying frame of reference [2]. Adopting gravitation as the naturally given frame of reference in 3D space imposes an external, immutable topology and orientation at least with respect to the y-axis. In other words, the proposition that an object a is resting upright on top of another object b is invariant to the viewers position. Instead, left, right or clockwise are relative relationships with respect to the viewers position in x-z-space.

The domain model is a time–triggered, and therefore dynamical, representation of the current environment. It consists of instances of objects and relationships from the concept model. Such an individual incorporates information about its specific properties and relationships to other individuals.

The underlying knowledge representation formalism is a description logic. Concretely, we are using the LOOM system [9] which is based on CommonLisp using the CommonLisp Object System (CLOS) extension. LOOM combines a description language with a rule language and uses an expressive inference engine extended for reasoning with time, units and dimensions. According to the terminology of description logic formalisms the concept model is represented in the TBox whereas the domain model is represented in the ABox.

3.2 Graphical Component

The interface of the graphical component is shown in Figure 3. This interface is split into a *visualization area* and a *menu bar*.

With the menu bar the user is able to rotate, move, shift and zoom the whole scenery as well as a selected object.

The visualization area shows a 3D view of the domain model as it is currently

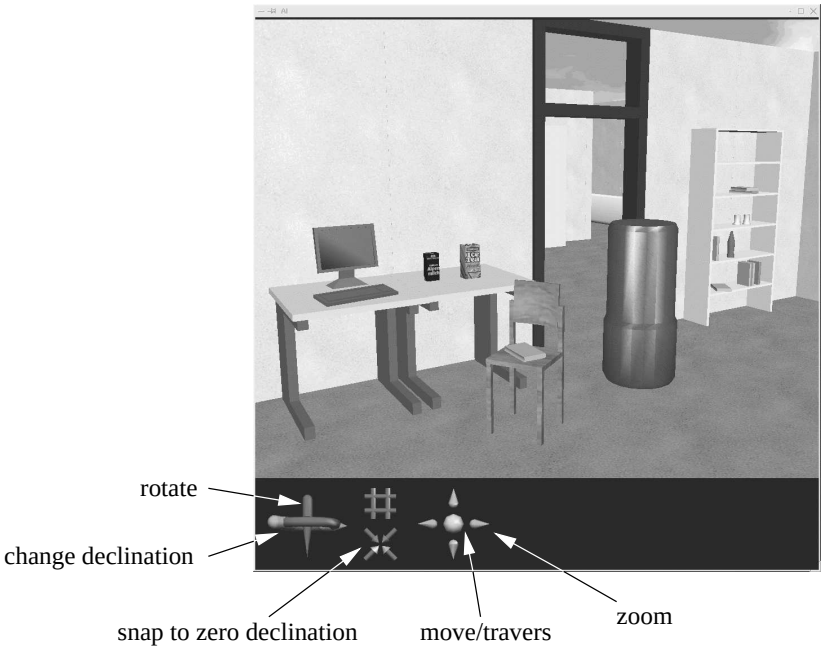

Figure 3: The KB-VISION interface: domain model visualization on top; menu bar on bottom.

represented in the underlying knowledge representation system. The view is shaded and textured in real-time. The user can select and deselect objects in this view. When selecting an object the graphical component retrieves information (type, relationship to other objects, etc.) about that object from the domain model. Depending on the type of the object the user is allowed to perform different actions. Chairs are modeled as move- and stackable objects in the concept model (see Figure 2). Therefore, an individual chair can be moved around and put on a table. Now, moving the table causes the chair (which is on the table) to move simultaneously. The fact, that the chair has to be moved whenever the table moves is deduced automatically by the knowledge base because of a referring rule about moving stacks in the concept model. This example illustrates how every graphical action directly interacts with the corresponding knowledge base. Tables cannot be stacked on other objects, because they are modeled as passive stackable objects in the concept model. Analogous, doors can only be moved around their centre of rotation, because they are rotatable objects. When deselecting an object the changed situation is checked to be consistent with respect to both concept and domain model. Consistent changes are asserted into the domain model, all others are rejected in the knowledge base as well as in the corresponding visualization. A new individual can be created simply by using a palette window containing all available object types defined in the concept model.

In order to approximate the real world as close as possible, objects should not geometrically overlap or float in space (at least in our domain). Those rules are represented as physical axioms in the concept model. However, the concept model does not reason geometrically on the base of volume models. Those kind of reasoning is done by the graphical component using collision detection mechanisms. This ensures that no object stored in the knowledge base overlaps geometrically with another one.

The graphical component is implemented in C and C++ and based on the well-known OpenGL standard, which is available for a large variety of operating systems and graphical window interfaces. Additional subcomponents are used in order to handle

user interactions (GLUT, GLUI) or to enable features which require collision detection (RAPID, VCollide [7]).

3.3 Network Connection

The network connection was build with the goal to ensure a network transparent connection between distributed components. Usually, server and clients do not physically run on the same cpu because of a resource or conceptual driven distribution of the clients. Currently we are using a simple TCP/IP connection to broadcast messages between the two components. Messages consists of strings containing LOOM assertions or LOOM query answers. In order to be able to relate each graphical object with its corresponding individual in the domain model the system uses a unique naming convention for communication. When selecting an object the graphical component queries all information (type and assertions) about the selected individual form the knowledge base. Concerning the stack shown in the visualization area of Figure 1 the knowledge base will answer among others with the assertion `(on Chair-01 Table-01)` for example. Even in a scenery with many hundred instances a single instance is typically related to only a few other instances. After each user action the graphical component sends the new positions/orientations of the changed instances via the network connection to the knowledge base. After evaluation the knowledge base accepts or rejects those changes and sends its answer back. Using this architecture, network communication is rare and message contents are short. This allows to run the knowledge base on a real mobile robot and the graphical component on an external graphical workstation using a wireless low-bandwidth network connection.

The interface is implemented in C and linked via foreign function calls into the CommonLisp framework. In order to gain more flexibility we plan to switch to the universal OKBC protocol [1] (Open Knowledge Base Connectivity) in the near future. The OKBC protocol allows to replace or to extend the knowledge based component of the KB-VISION system very easily with other knowledge processing systems providing different or additional reasoning services.

4 Performance and Related Work

The current implementation status of the KB-VISION system is that of a research prototype. Until now the main objective was to show the applicability of the approach with the premise of integrating as much functionality as possible, rather than optimize ergonomics or graphical performance. Nevertheless the implementation is currently in use in our research group with satisfying performance on Sparc and Intel processors running under SunOS respectively Linux. Currently, we are using one of recently developed (and still moderately optimized) OpenGL device driver for the 3D hardware acceleration of a nVidia TNT2 graphics card under Linux (XFree 3.3.5) with a Pentium III 450 MHz. Within this configuration KB-VISIONs graphical component achieves more than 10 frames per second with 140 objects out of 4500 triangles. Delays because of network communication with the knowledge base are not noticeable in this setting.

Our work covers at least two different research areas, namely knowledge representation and virtual reality (VR). Concretely, the KB-VISION system connects symbolic representations with pictorial expressions and vice versa. Other related work is mostly focused on either knowledge representation or VR.

Graphical modeling front ends for knowledge representation systems (e.g. GKB-Editor [11], Ontology Server [4], WebOnto [3]) differ from our approach in many ways. They are designed to give a structural view on the concept definitions while displaying the knowledge base as a semantical net. Instances are displayed as sets of assertions which mostly contain hyperlinks pointing to referenced instances or concept definitions.

The main objective in VR systems is first of all to produce realistic environments using 3D display techniques, head tracing, etc. Our approach is different from ordinary VR systems by using a highly and easy reusable, maintainable and understandable representation of the underlying knowledge and feedback mechanisms. Realistic user interaction and simulation within virtual worlds is a topic which is just now emerging in a research field called "Intelligent Virtual Environments" (see the special issue No. 1, Vol. 14 of the Applied Artificial Intelligence Journal, January 2000 for example).

A related knowledge-based system within this context is CODY a Virtual Constructor [8]. CODY enables an interactive assembly of 3D visualized mechanical parts to complex aggregates. However, this system is strongly tailored to the context of assembly simulation and offers therefore only a very limited number of different manipulation options.

Other, at least partial related systems are the various level construction kits for interactive computer games. Usually, those construction kits contain a graphical level editor for composing the 3D scenery. Functionality for different kinds of objects has to be coded in proprietary script languages. However, those approaches are not very flexible for further enhancements. Every new feature has to be taken into account in any existing script, for example.

5 Conclusion and Portability of the Approach

Many applications that implicitly have to reason about a fraction of the real world have to cope with a significant amount of domain knowledge. In order to build a domain model of a real environment, knowledge engineering experts often have to translate relevant information into specific syntax of the chosen representation system. This is a time-consuming and error-prone process. The KB-VISION system is a tool supporting the creation and manipulation of domain models graphically. However, in comparison with conventional modeling, additional effort has to be spend for building geometrical models of potential domain objects. KB-VISIONs import filters ease this task by allowing to import standard VRML models. All user actions are consistent with respect to the axioms stored in the knowledge base. In this sense, the KB-VISION system can be seen as a graphical feedback interface for the underlying reasoning mechanisms. This interactive nature qualifies KB-VISION also as an 'intelligent virtual environment' for humans as well as agents, planners, etc.

Our approach is suitable whenever an application demands to capture a huge amount of frequently changing domain knowledge (e.g. autonomous systems). Emphasizing the interactive aspect, the system could play an important role for intelligent, knowledge intensive simulators with human interaction (flight or car simulators), artificial agents (robot simulator) or tactical education (testbed for evaluation of tactical decisions, e.g. in military environments). The presented architecture is even conceivable as a realistic feedback component for 3D reality games.

Obvious future enhancements of the graphical component concern multimodal input and output options (speech, sound, data-glove, etc.) and speed-up techniques using detail-reduction of distant objects and areas. When using the system as an simulation

environment for virtual agents or planners, additional sensor modules have to be implemented in order to simulate supersonic or laser scanner input data from the graphical representation.

Currently, we are extending our concept model in order to be able to reason about aggregated complex objects (e.g. a table as an aggregate of a table top and its tablelegs).

References

[1] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 600 – 607, July, 1998.

[2] Antony G. Cohn. Qualitative Spatial Representations. In *Proceedings of the IJCAI-99 Workshop on Adaptive Spatial Representations of Dynamic Environments (ROB-2)*, 1999.

[3] John Dominque and Milton Keynes. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In *Proceedings of the 11th Knowledge Acquisition for Knowledge–Based Systems Workshop (KAW'98)*, April 1998.

[4] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. Technical Report KSL-96-26, Knowledge Systems Laboratory, 1996.

[5] Dieter Finkenzeller. Graphische Darstellung und Manipulation von Wissensbasen. Master's thesis, University of Ulm, Germany, 1999.

[6] Ira Goldstein and Seymour Papert. Artificial Intelligence, Language, and the Study of Knowledge. *Cognitive Science*, 1(1):84 – 123, 1977.

[7] T. Hudson, M. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: Accelerated Collision Detection for VRML. In *Proceedings of VRML'97*, 1997.

[8] Bernhard Jung, Martin Hoffhenke, and Ipke Wachsmuth. Virtual Assembly with Construction Kits. In *Proceedings of the 1998 ASME Design for Engineering arTechnical Conferences (DECT-DFM '98)*, 1998.

[9] Robert M. MacGregor. Inside the LOOM Description Classifier. *SIGART Bulletin*, 2(3):88 – 92, June 1991.

[10] Allan Newell. The Knowledge Level. *Artificial Intelligence*, 18:87 – 127, 1982.

[11] Suzanne M. Paley, John D. Lowrance, and Peter D. Karp. A Generic Knowledge-Base Browser and Editor. In *Proceedings of the Fourteenth National Conference on Innovative Applications of Artificial Intelligence*, pages 1045 – 1051, 1997.

[12] Günther Palm and Gerhard Kraetzschmar. SFB 527: Integration symbolischer und subsymbolischer Informationsverabeitung in adaptiven sensomotorischen Systemen. In *Informatik '97 - Informatik als Innovationsmotor. Proceedings der 27. Jahrestagung der Gesellschaft für Informatik*. Springer–Verlag, 1997.

[13] B. C. Smith. *Readings in Knowledge Representation*, chapter 3 Prologue to "Reflection and semantics in a procedural language", pages 31 – 40. R. J. Brachman and H. J. Levesque, Morgan Kaufmann, 1985.