MULTINET/WR: A Knowledge Engineering Toolkit for Natural Language Information

Carsten Gnörlich

Department of Applied Computer Science VII / Artificial Intelligence
University of Hagen, Germany
EMail: carsten.gnoerlich@fernuni-hagen.de

Abstract

In the field of artificial intelligence a lot of work is spent on developing knowledge representations, but few research is undertaken in finding the inherent characteristics of the knowledge engineering task and to develop comprehensive tools for it.
The paper investigates into the characteristics and requirements of the knowledge engineering process in a complex knowledge representation paradigm. A model of the knowledge engineering process is obtained by evaluating manual methods of assembling a knowledge base and is then adapted for a tool using a state-of-the-art graphical user interface. The resulting knowledge engineering tool is discussed and presented together with two case studies of actual knowledge engineering applications.

1 Introduction

A central field within the research of artificial intelligence (AI) is the development of knowledge representation formalisms. Knowledge representations are needed when computer systems must work with information which is not conveniently accessible or usable in its "natural" form (e.g., geographical data or text given in natural language). In general, a formal abstraction of the domain being modeled is created which is simple enough to be processed on a computer, but still produces an adequate model of the original information. By evaluating the shortcomings of early AI systems it became clear that in addition to be formally sound and complete, a knowledge representation has also to be intuitively understandable by a human user in order to facilitate typical knowledge engineering tasks like proof-reading and editing of the knowledge base.
Most current AI system fulfill these needs partially by warranting readability for the human user through methods for transforming the content of their knowledge bases into a verbose, well-structured format, or even by generating meaningful graphical representations of the stored knowledge.

However, only few AI systems have so far incorporated mechanisms or tools for improving the "human write-ability" (the entering and editing) of their knowledge bases. The difference can be pointed out best by looking at a related discipline, word processing. Written text is undoubtfully a knowledge representation, so word processing can be seen as the most basic form of knowledge engineering. Therefore comparing the status quo of "text engineering" and knowledge engineering, at least from a users perspective, seems reasonable.
Modern word processors are built using the "What you see is what you get" (WYSIWYG) paradigm, which proposes a common user interface for all integrated tools and more importantly, demands a unified representation in which all text processing tasks are carried out. Users view and edit documents in their final (usually, the print-out) form, and changes to the document can be reviewed immediately. As a result, "Text engineers" are alleviated from the "howtos" of text processing (like memorizing the printer code for bold face text and invoking the document previewer), and focus their energies on creating content instead.

Applying the same approach to more complicated knowledge engineering tasks in AI systems seems to be self-evident, however most AI systems still appear stuck in the 'pre-WYSIWYG' era. Well-known knowledge representations like KL-ONE[1] and SNePS[8] have been augmented with a large number of separate tools dedicated to specific knowledge engineering tasks, but it appears that not much effort has yet been undertaken in terms of combining these tools into a comprehensive knowledge engineering system.
Knowledge acquisition is typically carried out in several "compile-and-edit" steps: A knowledge editor provides some means of creating and manipulating the knowledge base in a formal textual representation. The resulting files are processed with an inference engine, and another tool provides a graphical representation of either the files from the knowledge editor or the results of the inference process. Therefore, a common user interface for all required tools is missing as well as there is no unified knowledge representation since knowledge is entered in text mode while it is browsed using graphical representations.

In this paper it is demonstrated that creating WYSIWYG tools for knowledge engineering tasks is possible, and even more important, that the additional effort in developing the user

interface yields similar improvements for the knowledge engineering process as in the word processing example. The knowledge acquisition tool MULTINET/WR will be presented which supports the MULTINET paradigm, a complex knowledge representation used at the University of Hagen [3].

The outline of this paper is as follows: In the next section, an example oriented discussion gives the reader an understanding of the concepts of the MULTINET paradigm. The development of an advanced knowledge engineering tool is motivated by the high affinity of the MULTINET paradigm to a graphical representation. Although the core of MULTINET is formally a directed graph, the complete MULTINET paradigm consists of a far greater complexity. Therefore, the task of editing and viewing MULTINET knowledge bases can not be carried out by using or adapting standard graph visualization methods like those provided by the Graphlet framework [6] or similar libraries. As a consequence, the MULTINET/WR system has been specifically built for use with the MULTINET paradigm.

The main body of this report, section 3, discusses the characteristics and requirements of knowledge engineering tasks in MULTINET and describes possible solutions using state-of-the-art user interface methods.

A MULTINET knowledge base is fed by different knowledge sources; by the human knowledge engineer as well as by automatic computerlinguistic tools, which leads to certain implications for the overall system architecture. In order to provide an easy to use and intuitive user interface, the general approach of a knowledge engineer creating MULTINET expressions by means of pen and paper is analyzed and taken as a model for the computer user interface. The section concludes with an in-depth presentation of the MULTINET/WR system and the available tools contained in MULTINET/WR.

Section 4 contains two case studies of actual applications of the MULTINET/WR system. The first application resembles a traditional knowledge engineering task of manually creating MULTINET expressions and is therefore concentrating on MULTINET editing and viewing tasks. In the second example, the MULTINET/WR system hosts a prototype module of a natural language interface. Here, the MULTINET/WR system does not only serve as a knowledge editor, but also acts as a development environment for building knowledge-based systems.

2 An introduction to the MULTINET paradigm

The MULTINET paradigm (Multi-layered semantic networks)[3], which has been developed as a knowledge representation paradigm at the University of Hagen, is a semantical representation for information given in natural languages like German[1].
MULTINET is rooted within the family of semantic networks which have historically been a valued formalism for expressing the meaning structure of natural language. Since the evolution of semantic networks was greatly influenced by results from cognitive psychology, semantic network formalisms like MULTINET represent a promising attempt at modeling the human knowledge representation. See [3] for a detailed discussion of the historical context and a cognitive motivation of the MULTINET paradigm.
Within this paper, MULTINET will be discussed from a technical perspective. Basically, the specification of MULTINET can be broken down into the "core" semantic network and an additional set of attributes which partition the semantic network into layers of different forms of knowledge. The following sections provide some basic notions from the MULTINET formalism.

2.1 Basic MULTINET concepts – The semantic network

The core of the MULTINET representation is the semantic network which is formally a directed labeled graph. Nodes in the graph represent certain entities in the discourse area while edges express semantic relations between the nodes. In principle every possible concept from the real world is eligible as a net node, while there is only a fixed set of about 100 relations allowed for labeling the edges, with each relation having its own pre-defined meaning.
The following table presents an excerpt from the MULTINET relation repertoire:

Relation	Description
AGT(e,a)	a is the agent in an event e
DIRCL(e,l)	an event e is directed towards a location l
OBJ(e,o)	o is an object being involved in the event e
ORIGM(o,m)	o consists of material m
PARS(p,w)	p is a part of w
PROP(o,p)	object o has a property p
SUB(x,y)	concept x is subordinated to superconcept y
SUBS(e,a)	a is an action or a state which is subordinated to concept e

Table 1: An excerpt from the possible MULTINET relations

Using the relations from table 1, the meaning of the facts "*Bob drives the yellow car to Berlin. Cars are made of steel. A car has a motor.*" can be expressed as a first approximation in a semantic network as shown in fig. 1 below.
Inner nodes (c1, c2, ...) represent complex concepts which result from combining the semantic relations between the c_i node in question and its neighboring nodes. For example,

[1]Currently, our research concentrates on representing German language text, but in general, the MULTINET paradigm can be applied for other natural languages also.

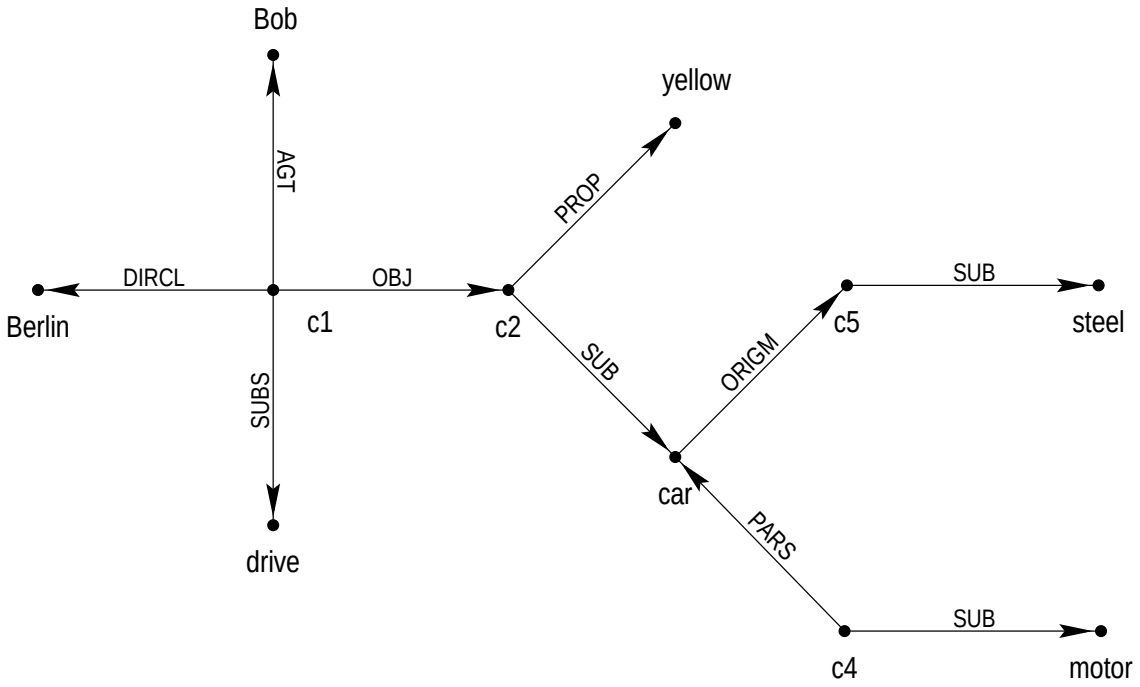

Figure 1: Representing "Bob's car world" in MULTINET

c1 describes the event that Bob is driving to Berlin using the object represented by *c2*, and *c2* in turn represents the yellow car. "SUB" edges differentiate between instances (or subclasses) and classes of objects, allowing for objects to inherit properties from their classes: Bob is driving a certain instance of a car (*c2*), which inherits all information from the *car* node. Therefore it can be inferred that Bob's car is made of steel and that it has a motor. The interpretation of the other edges follows from the descriptions of table 1.

The basic semantic networks discussed so far are only suitable for a very coarse knowledge representation. Therefore, the expressive power of MULTINET is increased by augmenting it with additional types of knowledge described in the next section.
The need for further knowledge types is motivated by discussing semantic restrictions which may be employed to perform consistency checks within the knowledge base. In the following two examples an important conceptual difference between the concepts "motor" and "steel" is revealed which can not yet be described by means of the basic semantic network:

The car saves fuel because it contains 10% less steel.
The car saves fuel because it contains 10% less of a motor.

The first sentence is acceptable because steel is a substance which can be divided nearly infinitely without losing its main characteristics: 90% of a given portion of steel is still a valid concept of steel. On the other hand, the second sentence does not make sense since motors are discrete objects which can not be divided any further. 90% of a motor is not a valid concept and if such utterances are actually observed, they have a metaphoric meaning and must be treated differently.
In order to model these kinds of background knowledge, and ultimately, to allow for correct formal inferences, the semantic network is augmented with additional types of knowledge

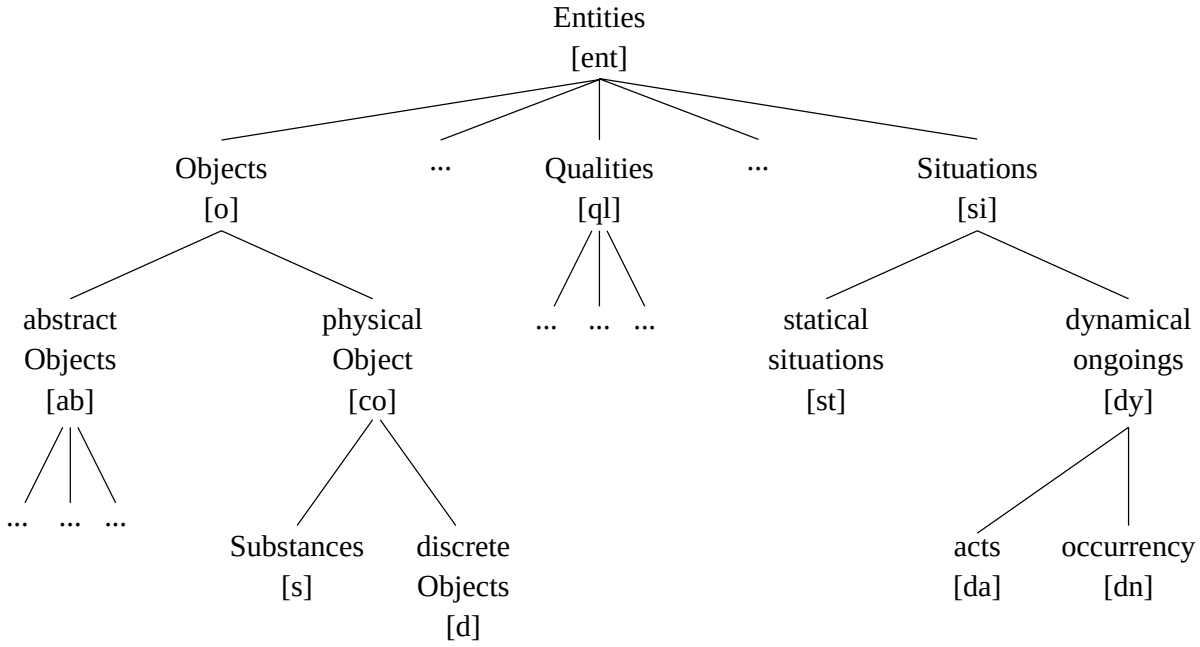

Figure 2: Excerpt from the ontology of concepts (the sorts) in MULTINET

described in the next section.

2.2 More sophisticated MULTINET concepts

The transition from simple semantic networks (as described in the previous subsection) to multilayered semantic networks is carried out by adding several new attributes to the nodes and edges of the semantic network, and by applying higher-order expressions (axioms and encapsulation of sub networks) to the basic semantic network elements.

Sorts and Layers

Nodes are augmented with additional attributes called “sorts” and “layers”. While nodes are representatives of concepts, sorts are used to partition these representatives into a conceptual ontology. By doing so, one important conceptual difference between steel and the motor of the car (from the example at the end of the previous subsection) can be properly formalized. Figure 2 shows a small excerpt from the sort ontology of MULTINET. In the semantic network shown in fig. 3, sort information has been added to the nodes. Note how “steel” and “motor” have been assigned with different sorts, according to the aforementioned observations.
Also, the action “Bob drives to Berlin” represented by node *c1* has been tagged with the sort for “acts” (da) in contrast to the sort for “occurrence” (dn). The difference between the two is that the former requires some sort of agent (“Bob drives”), while the latter doesn’t have an identifiable agent (“It rains”). This kind of information is important from an inference stand point, since searching for the agent is a typical subgoal in answering questions over a given knowledge base. The sort associated with “yellow” (tq) is a subsort of “Qualities” (ql); see

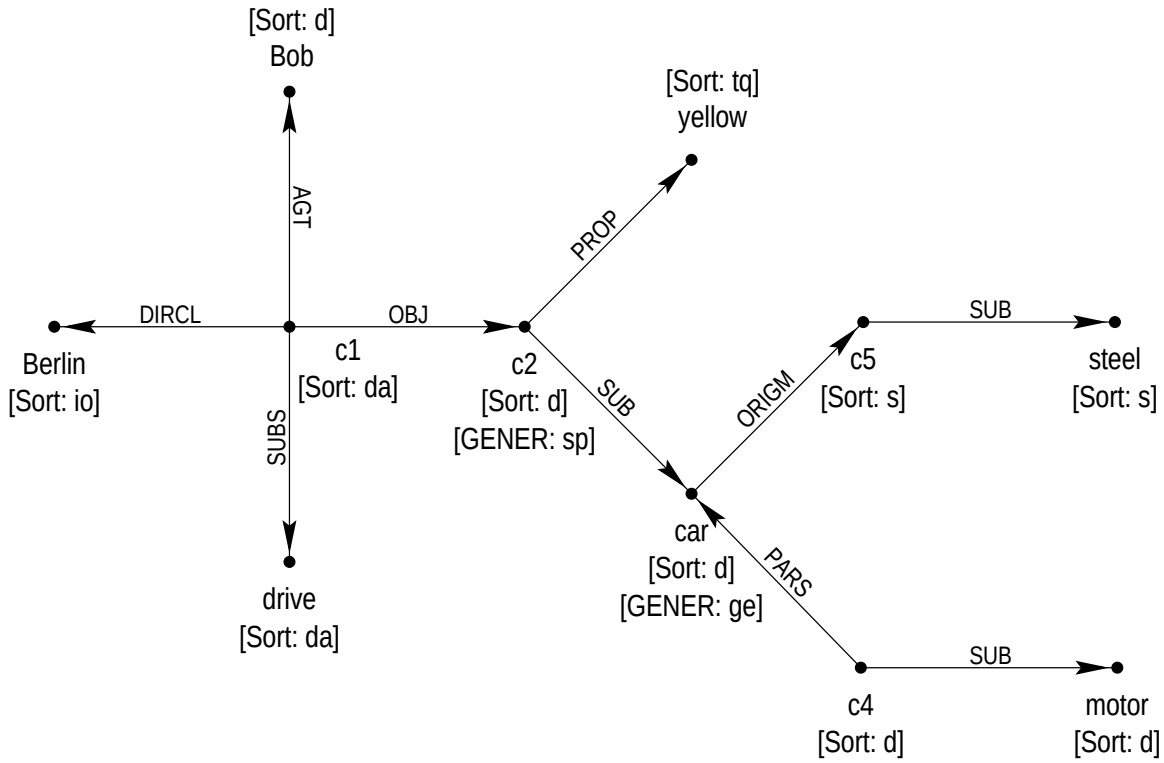

Figure 3: Adding sorts and layers to "Bob's car world"

[3] for more detailed information on the complete sort ontology of MULTINET.
It is important to note that this kind of sort information can not be deduced from the textual information given earlier in this section. Instead, sort information is typically obtained by looking up concepts like "car", "steel", "yellow" etc. in the computer lexicon, which contains the sort information among other things.
Additional information is attached to each node by means of seven so-called "layer attributes". These attributes stem from comprehensive linguistic and cognitive considerations; explaining them in detail would exceed the scope of this paper. Just to give an example, note that the nodes *c2* and *car* have the additional layer attribute "GENER" (for generality) shown in fig. 3, meaning that *c2* represents a special car, while *car* represents the general concept "car". See [3] for more information on layer attributes.

k-types

Edges represent semantic relations between entities of the discourse area. As is turns out, these semantic relations may either be relevant to a concept only within the scope of a specific situation, or they may define the concept with an universal meaning. In order to describe these different types of semantic relations, edges are associated with a "k-type" attribute for each starting and ending node, which may have among others the following values:

- **situa** – this type of information is tied to the actual situation.
  Example: (c1 OBJ c2) – $situa$ for node $c2$,
  Car *c2* is the object in the current situation "Bob drives to Berlin", but outside of this

situation, the knowledge represented by the "OBJ" edge is not really meaningful for the concept *c2*.

- **proto** – default knowledge.
  Example: (car ORIGM c5) – $proto$ for "car",
  because it is reasonable to assume that cars are made of steel. Some cars may be made of aluminum though, therefore this type of knowledge can be overridden in subordinated concepts if more specific information becomes available.

- **categ** – knowledge viewed as to be universally holding.
  Example: (c4 PARS car) – $categ$ for "car".
  At least by the writer's opinion, each car by definition contains a motor.

Sub networks as complex situations

Usually, MULTINET edges refer to situations constituted by a single MULTINET node and its context within the remaining network; all examples from the previous sections have been built this way.
However, in some cases edges need to refer to a complex situation which is itself described by a semantic network. In order to fit these cases into the graph-oriented semantic network formalism, MULTINET contains a higher-order mechanism for tagging a set of nodes and edges as a sub network. The resulting sub network then serves as a "virtual node", which can be used just like ordinary nodes in conjunction with MULTINET relations.
Demonstrating the need for sub networks requires relatively sophisticated example constructions which would exceed the scope of this overview.

Axioms

Axioms are employed to describe inferential links between MULTINET relations (R-Axioms), or between represented concepts (B-Axioms).

Consider the following example of a R-Axiom:

`(a PARS b)` $\wedge$ `(b ORIGM c)` $\rightarrow$ `(a ORIGM c)`

Basically, this axiom states that knowledge obtained via an "ORIGM" edge will spread transitively over "PARS" edges. When applied to the semantic network from fig. 1, it can be inferred that since a car is made of steel, and a motor is a part of it, the motor must also be made of steel. Obviously, R-Axioms provide a means of default reasoning, since there may be counter-examples for which they do not apply (substitute "tyres" for "motor" in the above example).

R-Axioms describe inferential connections between MULTINET relations and are therefore domain-independent. Within the discourse area, B-Axioms are used to express relations between concrete concepts. The following example is a bit farfetched, by might still be of use in the car domain from fig. 1:

`(s AGT a)` $\wedge$ `(s SUBS drive)` $\wedge$ `(s OBJ b)` $\rightarrow$ `(b SUB vehicle)`

The axiom describes that if there is an actor a who drives an object b, then b must be some form of an vehicle (e.g. it would infer from fig. 1 that (*c2* SUB vehicle) holds. This axiom would be helpful in answering paraphrased questions on a knowledge base ("What is the color of the $vehicle$ Bob drove to Berlin"). [3] gives about 150 examples of R- and B-Axioms for the MULTINET paradigm.

2.3 Applications of the MULTINET paradigm

The MULTINET paradigm can be applied whenever the need arises to process information or knowledge which was originally given in natural language.

Natural language interfaces and background knowledge. Using the MULTINET representation in a natural language interface (NLI) is straight-forward. The aim of a NLI is to conduct a data retrieval task by letting the user formulate queries in his mother language (e.g., German) as opposed to using formal retrieval languages like SQL or form-based entry fields.
However, natural language expressions are extremely unwieldy for direct processing on computer systems due to their non-formal and redundant nature. Therefore, the NLI could first translate the natural language query into a MULTINET representation which still carries the semantic meaning of the original query, but provides it in terms of a formal language. For the German language, a suitable parser is already available [5]. Building on the MULTINET representation, further processing like generating (and ultimately, conducting) a database query in a formal language can be done. In section 4.2 of this paper, a rule-based approach to transform natural language queries into SQL expressions will be outlined.

Knowledge representation. Since MULTINET is capable of representing the meaning of natural language expressions, it is also a powerful knowledge representation by itself. The previously mentioned NLI contains several types of background knowledge modeling among other things the peculiarities of the data base scheme and a general dialogue model. Expressing the background knowledge in MULTINET is not only convenient because of its expressive power, but also results in representing the user query and the background knowledge in the same formalism making inference processes (e.g., rule application) very elegant.

The representation of textual meaning. The use of the MULTINET representation as a knowledge representation can be even taken a step further by automatically processing whole textual corpora and storing them in MULTINET format. A typical application would be an advanced library information system which contains the content of technical abstracts as background knowledge in MULTINET. Such a system would allow for content related queries on technical reports going well beyond the abilities of current key-word based systems.

3 The MULTINET/WR knowledge engineering toolkit

Knowledge acquisition in MULTINET means constructing and maintaining a multilayered network structure which may either represent the meaning of a given natural language text, or consist of "stand-alone" background knowledge like in the NLI application.
Because the MULTINET representations are backed by means of computerlinguistic tools, MULTINET knowledge engineering tasks usually do not have to start from scratch. In order to build parts of the knowledge base, the knowledge engineer has the choice to enter the information manually, or to obtain and integrate results from existing tools into the knowledge base. For the purpose of the system design, it is helpful to model the above choices as different knowledge sources. By counting the knowledge engineer as one knowledge source, and differentiating between tools with are built into MULTINET/WR and those which are integrated by means of an external system interface, three major sources of MULTINET expressions can be identified in a typical MULTINET knowledge engineering task:

1. **The knowledge engineer**, who inspects, modifies and augments knowledge provided by the above mentioned tools. The knowledge is accessed and entered by means of a graphical user interface.

2. **Built-in tools.** Examples are the assimilation tool, which helps integrating new MULTINET information in an already existing body of knowledge. A layout component calculates topology information in order to arrange MULTINET structures in a visually pleasing way[2].

3. **External tools.** A word-class functional analysis WCFA [5] generates MULTINET expressions from natural language phrases and sentences. Additionally, a computer lexicon HaGenLex [7] exists in the MULTINET framework whose task is primarily to provide linguistic information to the WCFA, but it can also be used to augment the structure of net nodes with lexico-semantic information. This is particularly important for nodes which have been manually added to the knowledge base.

During the construction of a complex MULTINET knowledge base the weight between the three major knowledge sources may shift. Therefore, the MULTINET/WR system must be prepared for fully manual input of MULTINET expressions as well as for nearly automatic generation of semantic networks. Especially, it must ensure that knowledge from all different sources can be used interchangeably, and that subsequent revisions of the knowledge base do not block out certain knowledge sources. In order to fulfill these needs, the MULTINET/WR system is designed in the "cloverleaf" architecture depicted in fig. 3.

In the center of the system lies the knowledge base which integrates the three main knowledge sources and mediates the information flow between them. Access to the knowledge base is supported by an application programmers interface (API) which encapsulates all features of the MULTINET paradigm and the respective knowledge base functionality, and is in form and content based upon an abstract datatype developed for the MULTINET paradigm.

[2] In a strict sense, topology information is not a part of the MULTINET paradigm, but it is necessary for viewing purposes within the MULTINET/WR system and treated as a local extension of the semantic network structures.

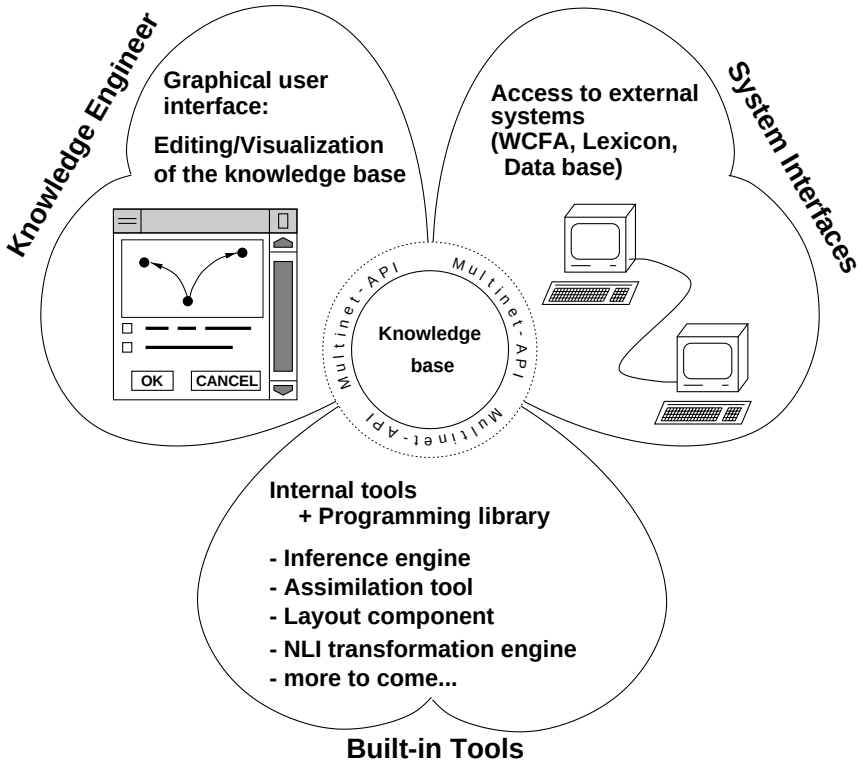

Figure 4: The "cloverleaf" structure of the MULTINET/WR tool

The provided implementation of the API wraps the knowledge base into a "black box" of well-defined interface routines which can not be by-passed from the three leaves. On the user interface side, graphical viewing and editing facilities exist for every information which can be retrieved and changed through the API, giving the knowledge engineer access to all knowledge base contents without having to recourse to the programming language level.
By forcing all "clients" to access the knowledge base through the API, full interoperability between the three leaves with their connected tools is guaranteed in a transparent way.

In the following subsections, the contents of the three leaves will be discussed in more detail.

3.1 The graphical user interface

The task of the graphical user interface (left leaf) is to give the user full access to the knowledge base, and to support the knowledge engineering process in a comfortable and natural way following the so-called WYSIWYG[3] principle.
Since the WYSIWYG paradigm requires that viewing and editing of the semantic network take place in the same visual representation, finding a suitable human-readable description for MULTINET is very important. Observations from our lectures and discussions on the MULTINET paradigm revealed that the graphical semantic net notation is the easiest to understand for the human reader (the other alternatives being a formal language to represent MULTINET structures in a textual format and a logical notation). Therefore, a full graph-

[3] What you see is what you get

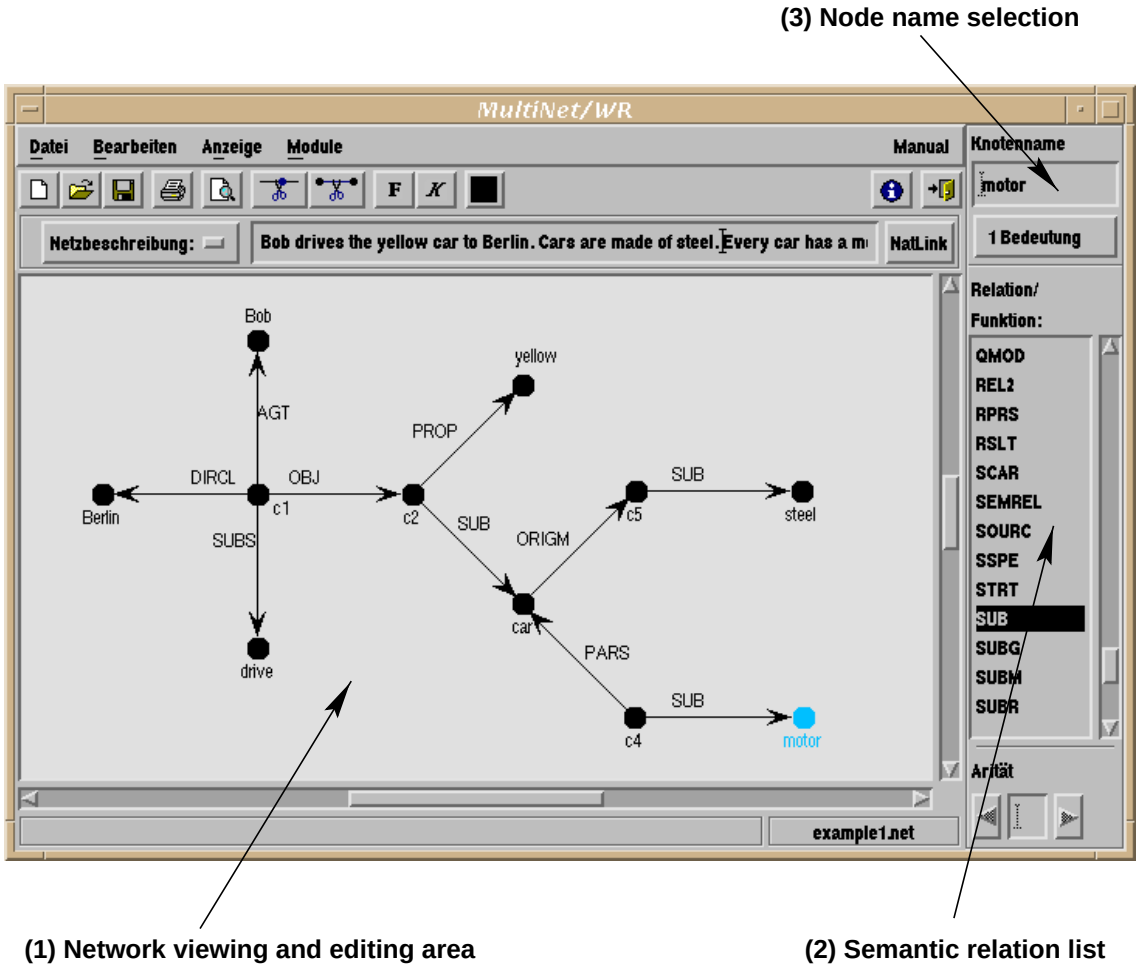

Figure 5: Basic editing functions of the MULTINET/WR tool

ical representation (with drawn nodes and edges) is chosen as the common representation through the whole system.
It will turn out in the following discussion that this representation is not only optimal for viewing purposes, but is also a natural choice for editing semantic networks.

3.1.1 Basic editing tasks

The basic editing functions of MULTINET/WR comprise the entering and editing of semantic networks. Since MULTINET is a very complex formalism, it would be confusing to view and enter all possible MULTINET elements (cf. section 2) in an unstructured manner. To obtain a convenient structure for the user interface and to model the stages of the editing process, the typical approach of a knowledge engineer creating MULTINET structures using pen and paper had been analyzed and then adapted for the MULTINET/WR system.
It turns out that knowledge engineers do not use all features of the MULTINET paradigm simultaneously, but rather create semantic networks by carrying out several steps in which different MULTINET elements are added or refined:

1. First, the knowledge engineer starts with drawing the semantic network nodes and semantic relations. This step comprises the largest share of effort and consumed time of the overall work.

2. When a preliminary model of the core semantic network has been finished, additional attributes (sorts, layers, and the other elements from section 2.2) are added.

3. Usually, some fine-tuning of the results from the previous steps takes place.

- Unexperienced users have the possibility to look up the definition of MULTINET elements they are unfamiliar with during steps 1 and 2 in the documentation.

During these steps, entering and accessing of the semantic network (e.g., nodes and edges) plays the most important role, since it is the central task performed in step 1 and all subsequent activities start out from a certain net node or edge. Therefore, all functions related to the semantic network are assigned a permanent and exposed place in the user interface. Figure 5 contains a screen shot of the MULTINET/WR system and identifies the user interface elements involved in step 1.

The largest portion of the MULTINET/WR window is taken up by the semantic network viewing and editing area (item 1 in fig. 5). In case of a semantic network exceeding the physical size of the screen (in terms of a two-dimensional layout), the editing area uses the common approach of showing a rectangular subsection of the network which can be moved (scrolled) over the larger network. Besides from hardware constraints, the system imposes no limits on the size of networks, so virtually "infinite" networks can be processed.
Editing actions in the editing area are completely mouse-driven with different functions assigned to each mouse button. The left mouse button is used for selecting and "grabbing" network elements. Nodes and labels can be moved along the screen in real-time and without flickering due to an efficient graphics engine. New network components are inserted using the middle[4] mouse button. Clicking over a blank space of the canvas inserts a new node while pressing the button over an existing node and then dragging the mouse and releasing the button over another node will create a new edge. Finally, the right mouse button brings up a context-sensitive pop-up menu which lets the user access additional actions (like editing attributes and bringing up help information) for the underlying MULTINET concept.

The decision to distribute the editing functionality over different mouse buttons - as opposed to context switches between different editing modes like "inserting", "deleting" and "shifting elements" - has been taken in order to warrant the least disruption in the editing process. Considering the network editing area as the user's "paper" and the mouse as the "pen", the objective is to avoid situations where the user has to move outside the "paper" or to put down the "pen" to indicate a function. Following these premises, switching editing modes is not acceptable since it would require moving the mouse outside the editing area into some sort of menu or toggle button row.
Analogously, switches between mouse and keyboard input should be minimized also. Thus, MULTINET relations are selected by a mouse click from the list shown as item 2 in fig.5,

[4]Unlike Windows-based systems, Unix hardware provides three-button mice as a standard.

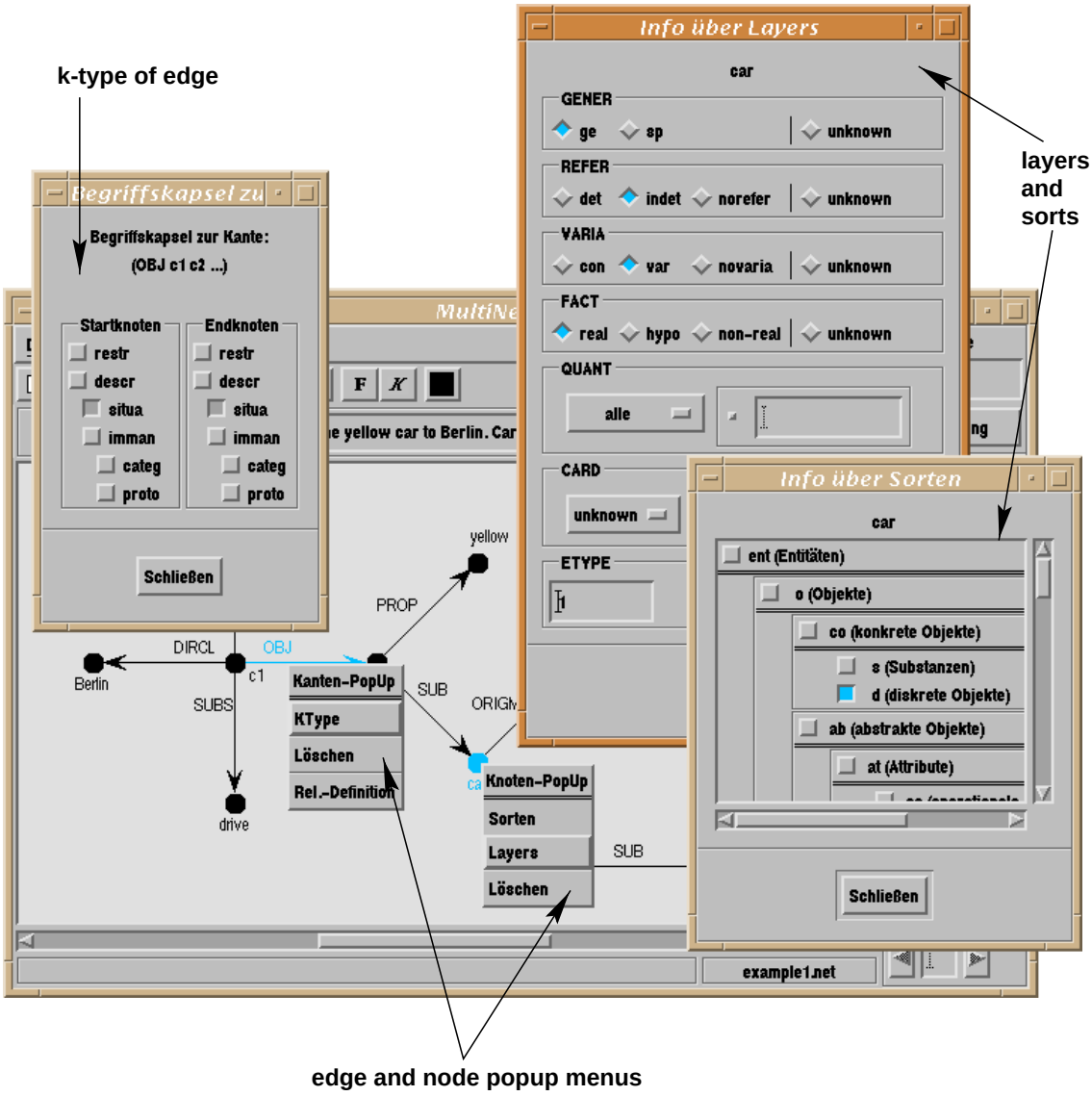

Figure 6: Popups for additional MULTINET elements

and keyboard input is only required for entering concept names in field 3[5]. Changes made in this field will always be applied to the most recently added or selected edge or node in the semantic network.

Further attributes of the MULTINET concepts are reachable through popup menus associated with the network nodes and edges. Therefore, the transition from step 1 to step 2 takes place by invoking the popup menus and filling in the additional information for the nodes and arcs of the network entered and prepared in step 1.
Figure 6 shows the pop-up menus for nodes and edges[6] and the resulting dialogues for k-

[5]The concept name selection field also provides a connection with the computer lexicon in order to select between different meanings of a concept; see section 3.3 for more details.

[6]For demonstration purposes, the figure shows pop-ups for edges and nodes at the same time. In the real application, only one pop-up menu can be active at one time.

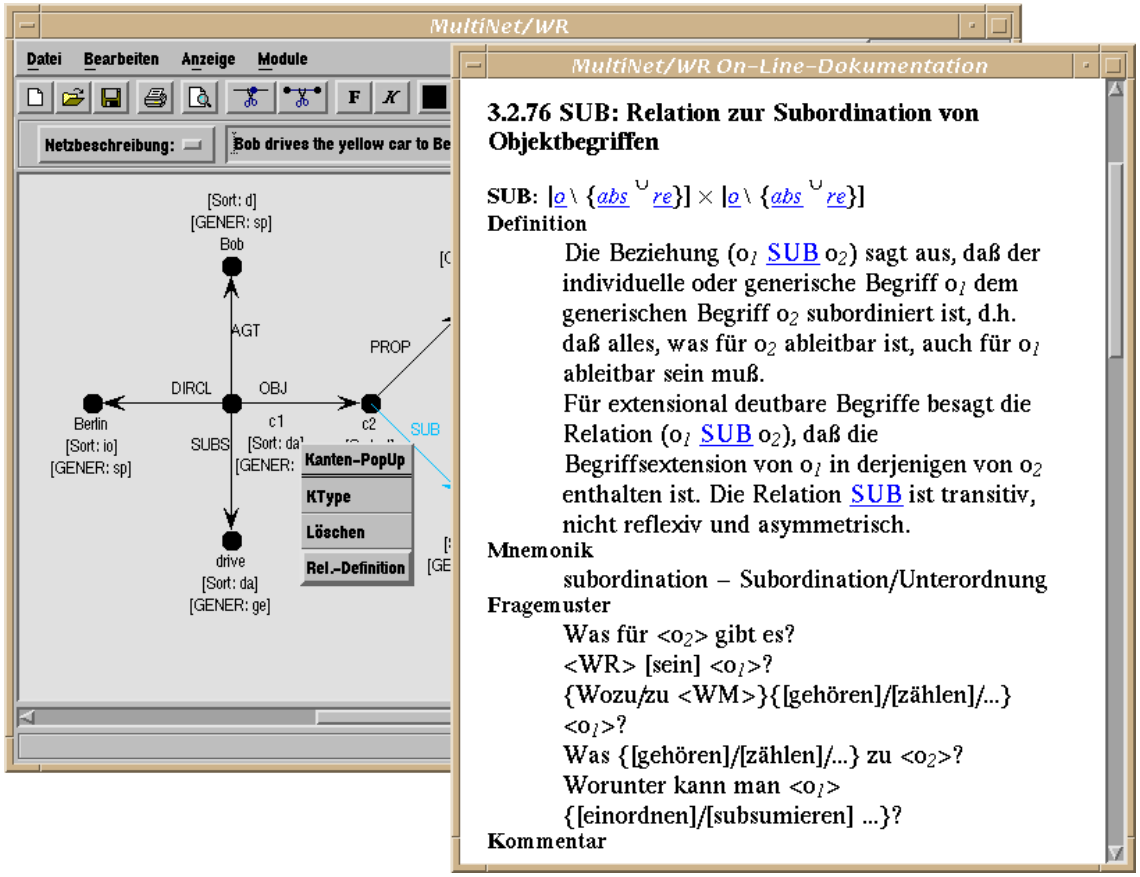

Figure 7: Displaying additional attributes and help information

types, sorts and layers. The shown dialogues are optimized for editing MULTINET attributes in a node-by-node (or edge-by-edge) approach, which is the preferred way in creating semantic networks. For browsing purposes, the MULTINET/WR system can be configured to display an arbitrary combination of additional attributes along with the semantic network in the viewing area. Viewing the network with different attribute combinations is usually done during in the final refinement step 3 of the previous discussion. Aside from these viewing requirements, no new user interface elements are needed for the refinement phase.

Figure 7 contains an example with a different display configuration. Furthermore, the figure includes an excerpt of the MULTINET online documentation invoked from a pop-up menu of a “SUB” edge. MULTINET is a very complex formalism where the documentation of the semantic primitive relations, functions, sorts and layer-attributes alone comprises more than about 150 printed pages. Since unexperienced users tend to look up representational means quite frequently during their work with MULTINET, a HTML version of the MULTINET documentation is included into the MULTINET/WR system. For most elementary MULTINET constructs, context sensitive help menus or help buttons exist which directly lead the user to the relevant part of the online documentation. In a similar manner, online help exists with information on the MULTINET/WR system itself.
Providing online help is an example of actively supporting the user during the knowledge engineering process. The next subsection discusses further methods of assisting the user

beyond the basic editing and viewing tasks.

3.1.2 Additional knowledge engineering support

With MULTINET being a formalism rich of different features, there is also the potential for human error when entering MULTINET structures manually. Luckily, MULTINET is well structured, so there is a maximum chance to detect ill-formed MULTINET expressions automatically. The MULTINET/WR user interface performs various consistency checks while the user is entering or modifying information, with either warnings or error dialogues issued depending on the severity of the problem.

MULTINET relations are defined to take nodes with a certain subset of sorts as their arguments; for example, the "ORIGM" relation requires its arguments to have the sort "co" (physical object). MULTINET/WR checks matchings between relations and their signature while the user is entering new network elements and highlights mismatched nodes and edges in a special error-condition color. It is then up to the user to either correct these parts immediately or to keep the network unchanged for a later review (users might prefer to concentrate on sorts after entering the whole network so detecting a relation/sort inconsistency should not block the editing process).

Conditions which would lead to non-recoverable inconsistencies in the knowledge base evoke an error dialog box which can not be left without rectifying the problem. For example, multiple nodes representing the same concept are not allowed in the MULTINET paradigm because of its object-centered nature. If the user tries to create a node for an already existing concept, the system will bring up an error dialog giving the user several options to solve the problem. In case of a network exceeding the size of the on-screen viewing area, the user might just have overlooked a node already existing for the concept he entered, so the system provides help for locating that other node and for reviewing its context within the semantic network. Then, the user can decide either to undo entering the double node or to combine the old and new node. The latter case typically arises when an existing node is renamed and equals a concept already present in the network. In that case, the combined node inherits the edges of both predecessors.

The support functions discussed so far check the user input immediately and return an appropriate action. These functions are directly implemented in the editing and viewing parts of the MULTINET/WR system (left cloverleaf of fig. 3). More complex tools are implemented as MULTINET/WR-plug-ins, which are described in the next subsection.

3.2 Built-in tools

While the basic functionality comprising the knowledge base and the graphical user interface are hard-wired into the MULTINET/WR core, a lot of system extensions are implemented using a plug-in mechanism. System extensions based upon the plug-in mechanism can be developed without recompiling the MULTINET/WR system core, so the development process of new system components is simplified and largely independent from an understanding of the inner workings of the system. Therefore, it is possible that writing a MULTINET/WR extension may be part of an exercise in future lectures or perhaps part of an diploma thesis work by interested students.

The existing built-in tools and modules, which are represented by the lower leaf of fig.3, can be classified into three categories:

1. Additional tools for the knowledge engineer.

2. Modules contributing functions to a high-level programming library built upon the MULTINET API. These modules represent generic functions which are considered useful in different applications.

3. Complex modules built for special applications.

The first category consists of tools the knowledge engineer can call any time during his work with the knowledge base. While the support functions discussed in the previous section concentrated on single network components like nodes or edges, these tools affect either the whole network or at least a significant portion of it.
A layout tool supports the visualization of semantic networks originating from external tools like the WCFA parser. Since the parser does not provide any (x,y)-coordinates or general network topology information for its MULTINET structures, the layout component is needed to heuristically spread the network over the screen surface while trying to minimize the number of edge intersections. As an add-on, the layout component is also capable of re-arranging the net around a picked reference node, making it easier to inspect the relations between a certain concept node and the rest of the knowledge base.
An assimilation tool comes into use when constructing large semantic networks from multi-sentence texts. While the WCFA is already very good at translating a single natural language phrase or sentence into a MULTINET expression, merging a whole paragraph of natural language text into a large semantic network currently lies outside its scope. In that case, the WCFA translates one sentence at a time from the given text, and the assimilation tool is used to assemble them into a common semantic network. It should be noted though, that the assimilation process is still far from being fully automated especially in those cases where logical inferences are required. Therefore, a certain amount of manual work and experience from the knowledge engineer is still needed to resolve all references between the existing network and the incoming information from the new sentence; however, work is under progress to understand and support the assimilation task better.

The second category of MULTINET/WR modules comprises system components which are usually not visible from the user interface, but are intended to be the building blocks of higher-order MULTINET-based functionalities. These modules include a programming library with functions built on top of the MULTINET API (e.g. providing graph traversal routines on MULTINET structures) and a rule-based inference engine. Section 4.2 outlines how the inference engine is used to build a natural language interface application to a bibliographical data base.

In the third category, a MULTINET/WR module like the before-mentioned natural language interface can support a special-purpose MULTINET application. In that case, the MULTINET/WR system serves as a development environment and test bed for the corresponding application.

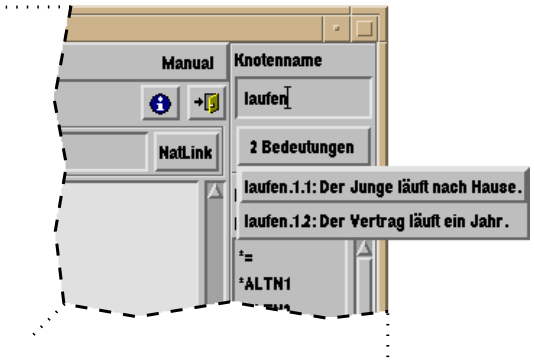

Figure 8: Selecting between different meanings of the verb “laufen” (engl. “to run”)

3.3 The external system interfaces

While the main goal of MULTINET/WR is to support the collection of knowledge represented in the MULTINET paradigm, another field of research on our faculty lies in automatically deriving MULTINET expressions from natural language text.
Within these research areas, the word-class controlled functional analysis (WCFA) [5] and a computerlinguistic lexicon HaGenLex [2] have been developed. Services provided from the respective tools are highly relevant for the knowledge acquisition task, and are therefore transparently embedded into the MULTINET/WR system as indicated by the right leaf of fig. 3.

The WCFA translates a natural language phrase or sentence into MULTINET expressions. Natural language text can be sent to the WCFA either through the MULTINET/WR user interface or using the programming interface. The resulting MULTINET expression will be automatically pre-processed by the layout component and is then available for editing and processing within the MULTINET/WR system like any other content of the knowledge base.

The primary task of HaGenLex is to provide computerlinguistic information to the WCFA, but it does also contain useful information for the MULTINET/WR user. When the knowledge engineer enters a new node like “car” into the knowledge base, MULTINET/WR automatically initiates a look-up for “car” in HaGenLex and augments the sort attribute of the node if appropriate information is available in the lexicon. Another important issue are different meanings of certain words. In order to give concept nodes a unique name, word labels have an extension “$.n.m$” [7] in the lexicon to indicate which meaning should be actually used. For compatibility with the lexicon and other MULTINET-based applications, it is important to specify the correct meaning when creating semantic networks. Whenever the user assigns a new concept name to a node, MULTINET/WR will look up all meanings in the lexicon and notify the user if more than one meaning is known for the first part of the concept name; the respective information is shown just below the node name entry field depicted in fig. 8. The user can then browse the different meanings in a pop-down menu and select the entry matching his intentions. The example from fig. 8 contains two meanings for the German

[7] n indicates the number in a list of homographs and m the number of a special meaning belonging to a certain word. For the sake of brevity, extensions have been left out in the examples of this paper.

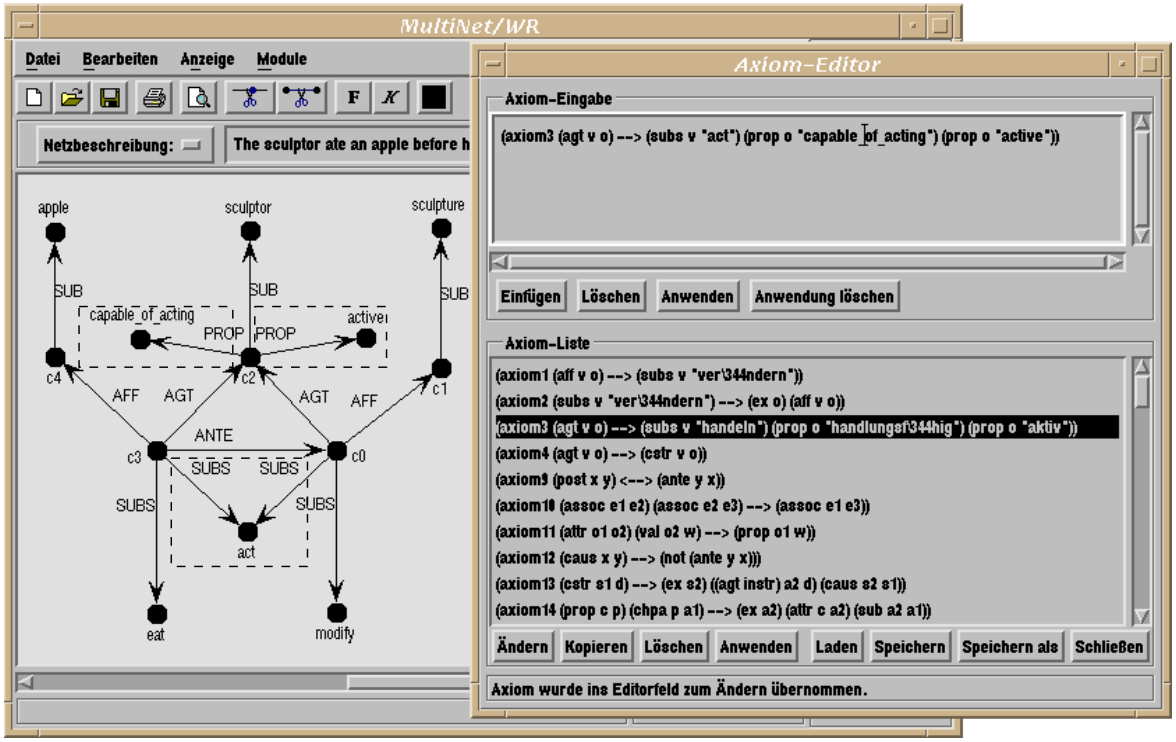

Figure 9: Trying out a MULTINET axiom

verb “laufen” (engl. “to run”), which are “run.1.1: *The boy runs home*” and “run.1.2: *The contract runs a year*”.

The WCFA and HaGenLex are developed independently of MULTINET/WR and reside on different machines on the local network. On the MULTINET/WR side, the external interface is actually just a set of stub procedures which handle the communication with the external tools over Unix sockets and translate their output into the MULTINET API. However from the user’s point of view it appears as if these tools were integrated parts of the MULTINET/WR system, and he is fully shielded from the peculiarities of operating and accessing the external components.

3.4 Axioms

The maintenance of the MULTINET axiom base is different from the knowledge acquisition tasks discussed before because it represents a form of “meta-knowledge engineering” on the knowledge representation itself. Changes to the axiom base are typically not carried out while entering semantic networks, and only the knowledge engineer maintaining MULTINET itself is involved in the updating of the axiom base.

The axiom base is gradually augmented as the work on the MULTINET paradigm continues, so axioms are the most frequently altered part of the MULTINET paradigm itself. Other elements of the MULTINET paradigm like the repertoire of available relations, the layers and the sort hierarchy are mostly static and are provided to the MULTINET/WR tool by means of read-only configuration files. While the static parts of the MULTINET paradigm can not be

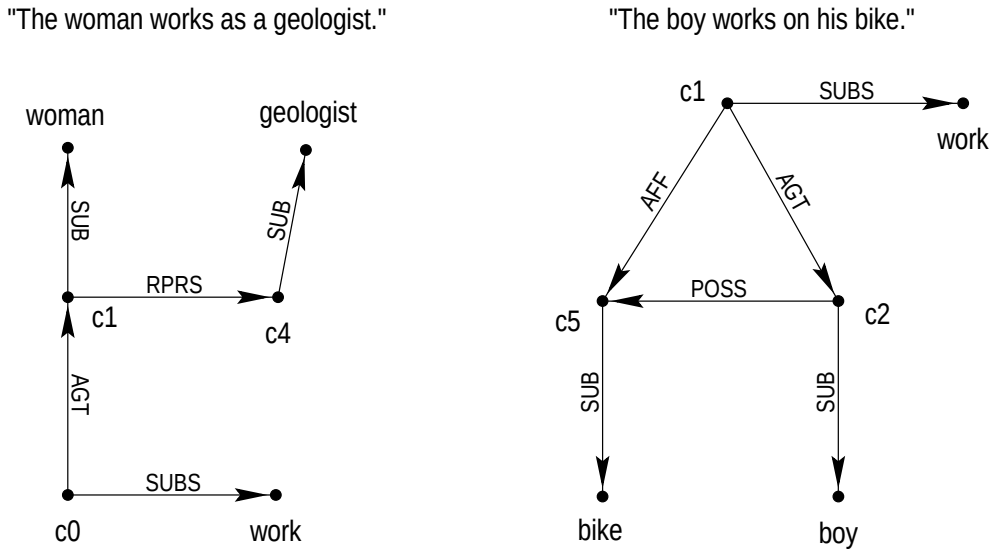

Figure 10: Example networks for two different meanings of the verb “to work”

changed using the functionalities of the tool, MULTINET/WR provides help in maintaining the axiom base as shown in fig.9.

Axioms are accessed and edited in a special window (called the axiom editor) shown in the right side of figure 9. The lower portion of the axiom editor shows a list of all available axioms while the upper area is reserved for the editing of axioms. This area allows for updating of already existing axioms as well as for creating new axioms. However the most important feature of the axiom editor is its direct connection with the knowledge base and the inference engine. Axioms can be selected from the list and then be applied to the semantic network in another MULTINET/WR window. Newly deduced network nodes and edges will be shown in a special highlight color in order to give the knowledge enterer a visual feedback about the axiom’s implications. Furthermore we found out that this mechanism encourages the creation of example networks for the axioms, which in turn is helpful for verifying the axiom’s semantics. In figure 9, the screen shot has been edited in a few places for didactic reasons. For better readability, the axiom shown in the editing field has been represented using English terms; the original version of the axiom is highlighted in the axiom list. In the semantic network, dashed boxes have been added to identify the network elements deduced from the axiom. Normally these elements are indicated by color changes in the MULTINET/WR system. The meaning of the semantic network translates to “The sculptor ate an apple before he modified the sculpture”.

4 Example applications

4.1 Description of German verbs in MULTINET

The first application of the MULTINET/WR tool is the creation of semantic network patterns describing the semantic structure of hundreds of special German verbs. The resulting semantic networks are part of the verb entries in the lexicon HaGenLex [7] and serve as guidance

for the natural language parser NatLink [5][8]. Figure 10 includes two examples showing different meanings of the verb "to work", which have been translated for convenience. The *RPRS(a,b)* relation from the left example expresses that a appears in the form of b. In the right example, *POSS(x,o)* denotes that x possesses the object o, while *AFF(p,o)* means that o is the object which is affected by the process p. Not shown in the figure is an abstraction step which replaces exemplary concepts like "woman", "boy", etc. with variables.

The work was carried out by a student worker with no technical background beyond having basic computer skills. After a brief introduction into the MULTINET paradigm and the basic functions of the MULTINET/WR program, the student was already capable of entering semantic networks into the system. The program was found easy to use and provided valuable guidance in "learning MULTINET by doing" through its various consistency checks and context-sensitive documentation features. During the 6-month part-time work period a comprehensive list of about 800 verbs was processed and described by means of the MULTINET paradigm.
It was the students and our experience that the availability of a WYSIWYG-style knowledge engineering system did largely contribute to the practicability of the MULTINET paradigm. Since both the MULTINET paradigm and the MULTINET/WR tool are intuitively understandable, the learning period before becoming "productive" (e.g., creating the first semantic network) is considerably shortened as if an approach of using formal language expressions and command-line driven tools had been chosen. The latter point is especially important in the field of natural language research since specialists with a linguistic background are normally not interested in dealing with technical issues.

4.2 The natural language interface application

A natural language interface (NLI) to bibliographical data bases [4] has been also developed on the base of MULTINET. In the NLI, the user enters queries in natural language, as for instance *"Which books about artificial intelligence have been written by Shortliffe?"*. We believe that natural language queries provide an alternative approach superior to standard form-based query methods. They better cover the needs of "naive" users better, i.e. of people not being familiar with computers.

The MULTINET/WR knowledge engineering tool played an important role in the development of this NLI by supporting the *transformation module* which translates the MULTINET representation of the user query into SQL commands for the target data base. The background knowledge needed for the transformation process is expressed by means of the MULTINET paradigm and maintained by the MULTINET/WR system. In addition, the MULTINET/WR system hosts a prototype of the transformation module. This makes it possible to utilize the MULTINET/WR "infrastructure" in the transformation module. Since the knowledge base, the inference services and the interface to the external WCFA are also used in the transformation module of the NLI, a significant portion of development time is saved. Even more important, the standard functionality of the MULTINET/WR user in-

[8]The verbs in question had been those which govern obligatory prepositional phrases, where the prepositional phrase can't be dropped without changing the sense of the verb. These verbs are especially difficult to describe semantically. For this reason, they are almost entirely neglected in semantic investigations.

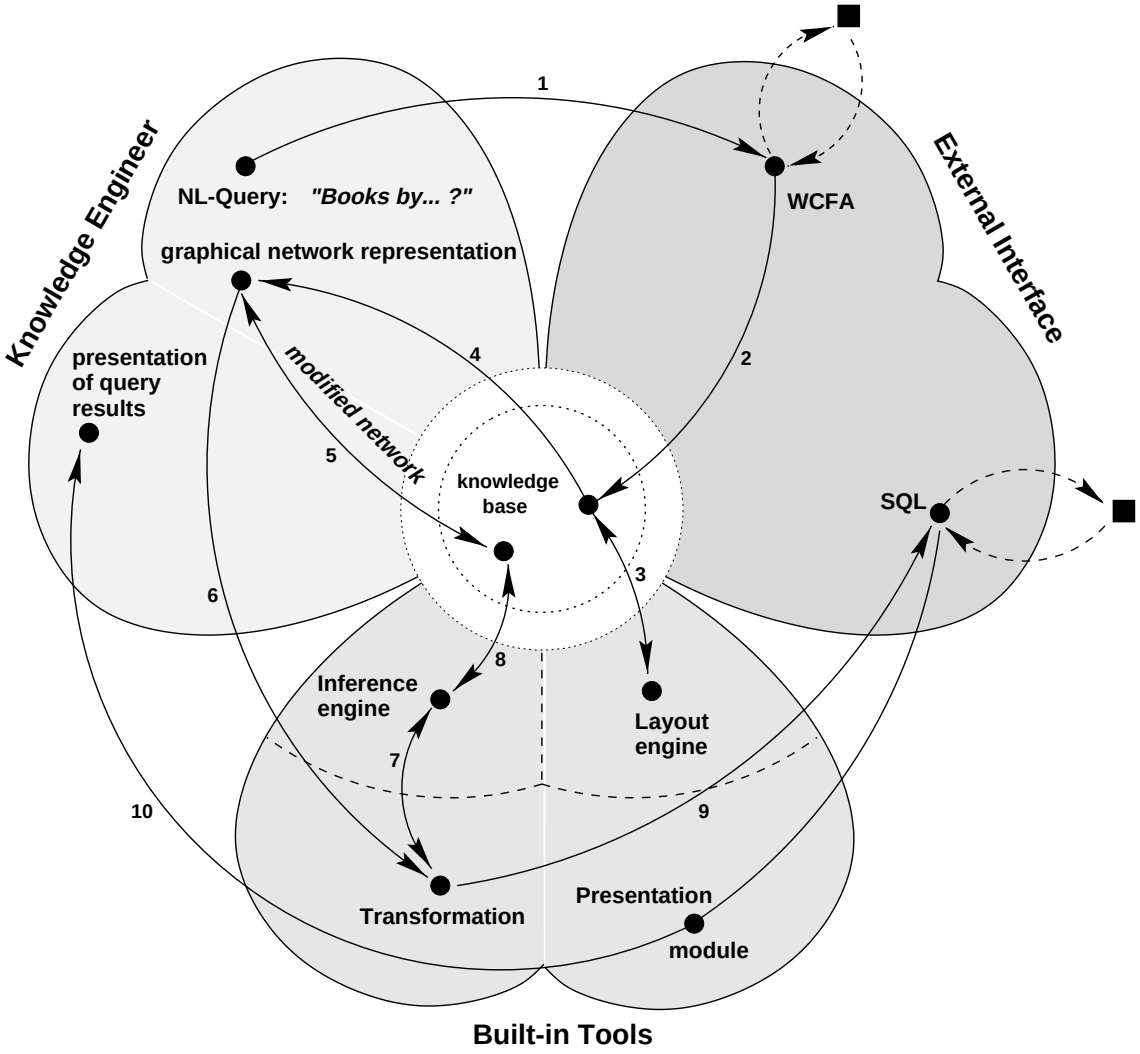

Figure 11: data flow in the MULTINET/WR tool during the transformation process

terface is available for inspecting and fine-tuning of the transformation process during the development of the NLI.

The following subsection gives an overview of the transformation component. Figure 11 summarizes the data flow between the MULTINET/WR modules involved in the transformation process. A screenshot of the MULTINET/WR system while processing a data base query follows at the end of this section in fig.12.

4.2.1 An overview of the transformation process

Processing a natural language query to data bases is basically a two-staged process. The first stage translates the natural language query into a MULTINET expression. This step is carried out by the standard WCFA interface of MULTINET/WR. The second stage transforms the content of the query expressed in MULTINET into a SQL-expression.
In this paper we will concentrate on the role of MULTINET/WR during the development of the NLI. The knowledge engineer types the query phrase into the NatLink text field which

is then sent to the WCFA server (step 1 in fig. 11). The received MULTINET representation of the query will be stored in the knowledge base (step 2), graphically layouted (step 3), and finally displayed in the network editing area (step 4). At this point, the knowledge engineer can apply manual changes to the query representation in order to correct errors or to generate special test cases for the transformation component (optional step 5).
Once the MULTINET representation of the query has been approved by the knowledge engineer (step 6), the transformation module performs the second stage of the query translation. The task of the transformation module is to find the relevant information in the natural language expression for each data base attribute; with typical data base attributes being related to information about the author, the title or the publisher of a book, and descriptor fields containing keywords on the book contents. Since we are looking to support existing data bases rather than data bases build specifically (or artificially) for our NLI, we often find data base schemes structured according to commonly used form-based access systems. Following this view, the transformation module can be thought of an assistant freeing the user from filling in the keywords into traditional form-based query sheets by automatically evaluating the user's natural language input.
The transformation follows a rule-based approach to recognize certain sub structures within the semantic network, which will eventually be projected into the suitable places of the formal data base query. Transformation rules are basically Horn formulas, with literals being either MULTINET relations like *SUB(a,b)* or user-defined predicates with an arbitrary predicate name. Typically, the left side of a transformation rule contains a partial MULTINET construct which is related to a corresponding attribute-value pair of the data base query language in the right side of the rule. In this particular example, the data base language is SQL, but the same approach can be applied for other data base languages, too.
Since transformation rules are formally a subset of the rules supported by the built-in inference engine of MULTINET/WR, the transformation component heavily relies on the inference engine for evaluating the transformation rules (step 7). In the transformation application, the knowledge base contains the MULTINET representation of the user query and additional background knowledge about the bibliographic library (see also fig 12 and the accompanying explanations on the background knowledge).
Rules are processed by the inference engine using a PROLOG-style backtracking algorithm. When trying to prove a literal, the system first attempts to match the literal against a MULTINET relation in the knowledge base (step 8). If no such matching exists, or all possible matchings have been exhausted in previous backtracking steps, the system continues looking for another rule containing the literal in its conclusion, and tries to prove that rule accordingly.
Finally, the transformation component assembles an SQL query from the results of the rule application, sends the query to the data base (step 9) and displays the results to the user (step 10). In order to carry out these steps, the MULTINET/WR system has been enhanced with additional modules for communicating with a local Oracle data base and a dialogue field for presenting the query results.

Figure 12 shows the MULTINET/WR user interface to the transformation component. Window (1) contains the background knowledge about the bibliographic library. The respective semantic network is manually created by means of MULTINET/WR's editing facilities. Typ-

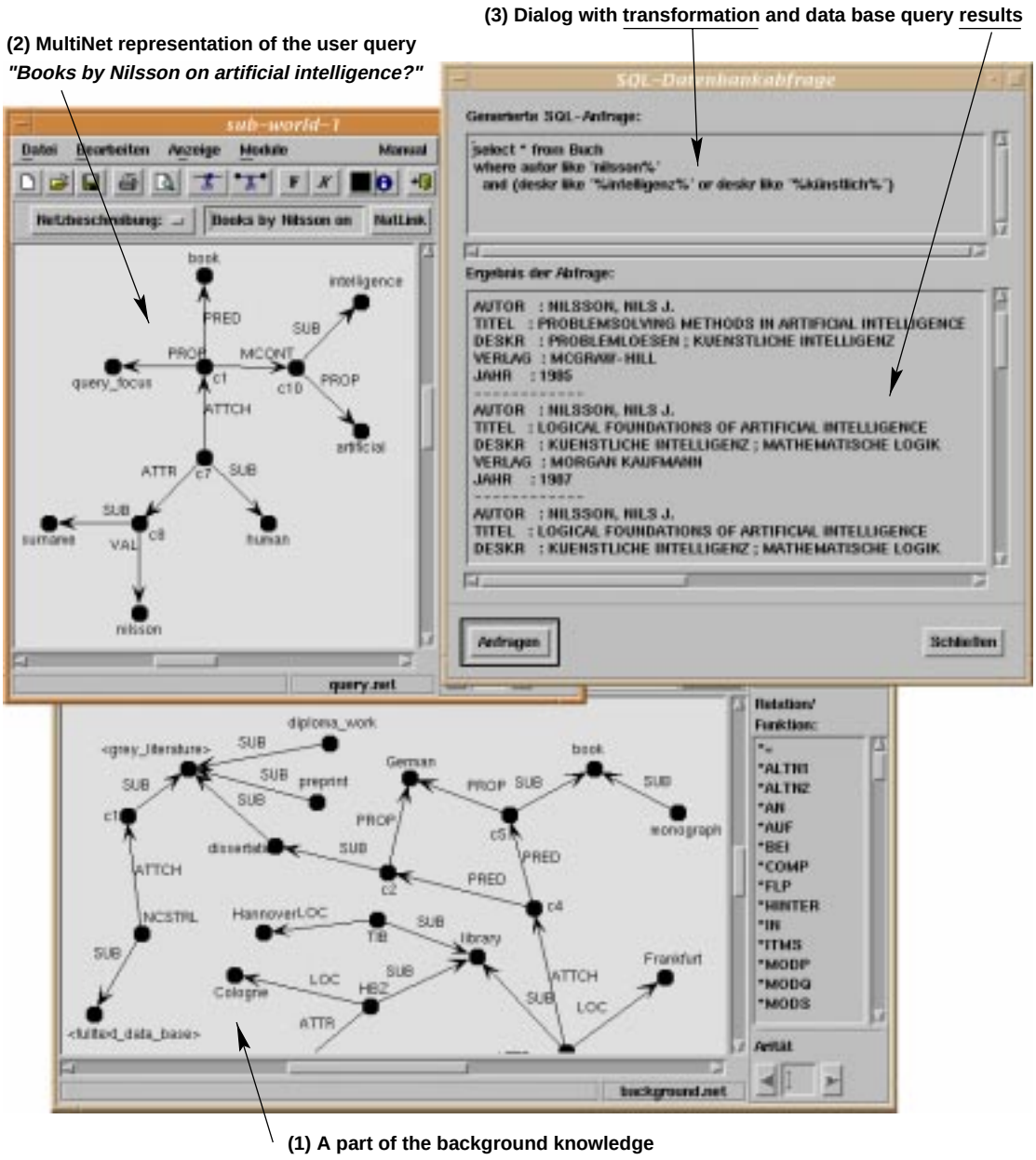

Figure 12: MULTINET/WR hosting the transformation component

ical contents of the background knowledge are a classification of several types of literature (e.g. that a monograph is a book) and information describing the addressable libraries themselves such as technical protocols for sending queries. The latter information is not exploited in the transformation prototype, but will be used in a more sophisticated NLI which provides access to more than one pre-selected library. See [4] for a more detailed discussion of the NLI and [3] for a definition of all MULTINET relations used throughout this example.

Window (2) contains the MULTINET representation of the user query which has been automatically created by sending the typed-in query phrase to the WCFA parser. If necessary, manual changes can be applied to the query representation. The transformation process is

invoked from the plug-in menu (not shown in the picture). It takes the semantic network from windows (1) and (2) as input and produces the SQL query shown in the upper region of window (3). The SQL query can be edited or rejected by the knowledge engineer in order to avoid sending non-correct queries[9] to the data base during development of the transformation prototype. Finally, the results of the data base query are presented in the lower part of the dialog box. Please note that the underlying library data base is actually about German literature which leads to some unavoidable inconsistencies in the example from fig.12. The background knowledge (window 1) and the MULTINET representation of the user query (window 2) have been translated for convenience, but appear in German language in the actual system. For the SQL query and the data base query results in window (3) no translations were possible because of technical reasons. The German term "künstliche Intelligenz" stands for "artificial intelligence" in the English version of the query.

5 Conclusions and further work

We observed as a starting point that WYSIWYG office systems shield the user from the technical side of a task and enable him to get started on his work immediately. A hypothesis was stated that the WYSIWYG paradigm can be applied also to a knowledge engineering problem in an appropriate manner and that similar improvements should result.
In this paper, the MULTINET/WR system was presented which successfully realizes a WYSIWYG user interface for the MULTINET knowledge representation paradigm. From our experience in several knowledge acquisition tasks, the abovementioned hypothesis has been verified. By looking at a cost/gain ratio it appears that the extra effort spent for developing a graphical user interface quickly pays off because of the easier access to the MULTINET paradigm.
As a side effect, the MULTINET/WR system prove to be a valuable help as an environment for developing the transformation module of the NLI application. Since the user interface of the MULTINET/WR system has already reached a mature state, ongoing work will concentrate on expanding the library of generic functionalities and methods in MULTINET/WR in order to support the development of further applications based on the MULTINET paradigm.

Encouraged by the users acceptance of the system (among them many students), we plan to include a special version of the MULTINET/WR system into an AI laboratory accompanying the artificial intelligence lectures at the University of Hagen. On the knowledge engineering side our aim is to extend the methods being currently available in the system towards assimilating larger parts of natural language text in MULTINET. If, for instance, the abstracts of literature sources were available in a MULTINET representation, the natural language interface could answer content-related queries with a much better precision in comparison with conventional descriptor-based methods.
Other interesting applications of processing large text bodies represented in MULTINET include document classification and summarizing systems, and finally implementing a system capable of answering questions about a full text stored in the knowledge base. Thus, the work done in connection with MULTINET/WR will be a good base for developing full-fledged Question-Answering systems.

[9] to prevent ill-formed queries from wasting a lot of CPU time on the local database

References

[1] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[2] S. Hartrumpf and M. Schulz. Reducing lexical redundancy by augmenting conceptual knowledge. In G. Brewka, C. Habel, and B. Nebel, editors, *Proceedings of the 21st Annual German Conference on Artificial Intelligence (KI-97)*, number 1303 in Lecture Notes in Computer Science, pages 393–396, Berlin, 1997. Springer.

[3] H. Helbig. *Die semantische Struktur natürlicher Sprache: Wissensrepräsentation mit MultiNet*. Springer, Berlin, 2001.

[4] H. Helbig, C. Gnörlich, and D. Menke. Realization of a user-friendly access to networked information retrieval systems. In *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, pages 62–71, Stanford, California, 1997.

[5] H. Helbig and S. Hartrumpf. Word class functions for syntactic-semantic analysis. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP'97)*, pages 312–317, Tzigov Chark, Bulgaria, Sept. 1997.

[6] M. Himsolt. The graphlet system. In *Symposium of Graph Drawing, GD '96*, number 1190 in Lecture Notes in Computer Science, pages 233–240, Berlin, 1996. Springer.

[7] M. Schulz. *Eine Werkbank zur interaktiven Erstellung semantikbasierter Computerlexika*. PhD thesis, FernUniversität Hagen, Fachbereich Informatik, Hagen, Germany, Aug. 1998.

[8] S. C. Shapiro. SnePS: A logic for natural language understanding and commonsense reasoning. In L. Iwanska and S. Shapiro, editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge und Knowledge for Language*. MIT Press, Cambridge, Massachusetts, 1999.