In T. Pequeno, F. Carvalho (Eds.), Proceedings of the XI Brazilian Symposium on Artificial Intelligence, Fortaleza, pp. 493–506, 1994.

MANTRA: A Multi-Level Hybrid Knowledge Representation System

G. Bittencourt[†], J. Calmet[‡], K. Homann[‡], A. Lulay[‡]

[‡] Universität Karlsruhe
Institut für Algorithmen und Kognitive Systeme
Am Fasanengarten 5
76128 Karlsruhe · Germany
{calmet,homann,lulay}@ira.uka.de

[†] Universidade Federal de Santa Catarina
Laboratório de Controle e Microinformática
88049-900 Florianópolis · SC · Brazil
gb@lcmi.ufsc.br

Abstract

The intelligent behavior of a system is based upon its represented knowledge and inference capabilities. In this paper we report on a knowledge representation and reasoning system, developed at the University of Karlsruhe, called Mantra. The system provides four different knowledge representation methods – first-order logic, terminological language, semantic networks, and production rules – distributed into a three levels architecture. The first three methods form the lowest level of the architecture, the epistemological level. The supported hybrid inferences as well as the management of knowledge bases form the second level, called logical level. Finally, the third level, the heuristic level, provides representation of procedural knowledge of a domain, and the introduction of ad hoc rules. This knowledge is represented in terms of production rules which are processed by a Ops5-like rule interpreter. This paper mainly describes the introduction of this level into the hybrid system. The semantics of the knowledge representation methods of the epistemological level is defined according to a four-valued logic approach. This definition insures that all inference algorithms are sound, complete and decidable. The system has been implemented in Common Lisp using the object-oriented extension CLOS, and the graphical user interface was implemented in C with XToolkit.

1 Introduction

One of the characteristics of research in the field of reasoning about knowledge is the lack of a unified theory. Therefore, some philosophical controversies in this domain have arisen, e.g. proceduralists versus declaratives or symbolics versus connectionists.

A central problem, when specifying an "intelligent" system, is to determine which knowledge representation formalism is more adequate to the intended problems. The implementation of a given formalism usually implies the compromise between expressivity and efficiency. The analysis of how efficiency is affected, when the expressivity increases, is an active research field.

Two solutions are available if one needs a large choice of knowledge representation formalisms: (i) to dispose of a large number of system building tools, or (ii) to use a hybrid system (e.g. [4]). We report on MANTRA[1], a system that supports hybrid knowledge representation which has been designed at the University of Karlsruhe during the last years and is still under development.

The system was implemented according to the following design principles: (i) **several cooperating formalisms** are better than a unique representation formalism, (ii) a **clear semantics** explaining the meaning of the knowledge representation language is fundamental and (iii) all algorithms involved must be **decidable** and reasonably fast. From a knowledge engineering point of view MANTRA could also be regarded as a general-purpose shell for building large knowledge-based systems.

The system provides four different representation formalisms which can interact through hybrid inference algorithms. The motivation is that several cooperating formalisms ought to enhance the expressiveness and inference power of the system. We adopt a knowledge representation approach consisting of a representational theory, explaining which knowledge is to be represented by which formalisms, and of a common semantics to define the relationship between expressions of different formalisms in a semantically sound manner. The decidability of all algorithms involved is achieved by adopting a four-valued semantics based on the works of Belnap [2] Patel-Schneider [12] [13], Frisch [9] and Thomason et al. [14]. The system supports four different knowledge representation formalisms – first order logic, frames, semantic nets and production systems – with their associated inference mechanisms – assertional reasoning, terminological reasoning, inheritance with exceptions and heuristic programming. The architecture of the system is shown in figure 1.

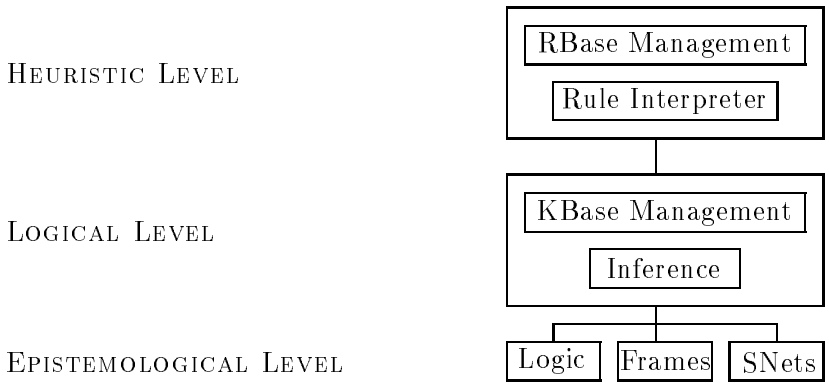


Figure 1: The Architecture of the System

The paper is organized as follows: In section 2, the theoretical background of a

[1]Modular Assertional, semantic Network and Terminological Representation Approach

decidable first-order logic, relying on the notions described in [12], is presented. This approach is used to define the semantics of the logic, frame and semantic network formalisms. In section 3, the different methods of the epistemological level are defined. In section 4, the logical level, that supports inference mechanisms and knowledge base management – creation, modification and query – is presented. In section 5, the introduction of production systems in the heuristic level is described, the rules provide the system capabilities for representing procedural knowledge and ad hoc rules for the hybrid inferences. In section 6, the implementation is commented upon and, finally in section 7, some applications and conclusions are presented.

2 The Semantics

One of the main characteristics of MANTRA is the definition of a unified semantics for knowledge representation and reasoning methods and their interaction. This semantics is based upon a four-valued logic [2] [12] [13] [9] [14].

Many knowledge representation systems support standard first-order logic due to its naturality and expressive power. However, when reasoning about some formulae, these systems may never stop due to the fact that first-order logic is undecidable. This situation is higly unsatisfatory if we intend to use this logical knowledge representation in a working system. Possible solutions are to restrict the expressive power of the logic, to change the inference methods, and to limit the length of derivations or the elapsed time. The four-valued semantics proposed by Patel-Schneider [12] [13] has been adopted because it does not restrict the expressiveness of the language and supports an inference mechanism, called t-entailment, that is decidable and semantically sound.

Initially, we briefly introduce propositional tautological entailments — a simple type of propositional relevance logic. The syntax of the logic of propositional tautological entailment is the same as that of standard propositional logic, but without an implication operator. Besides the standard two-valued assignment, formulae can also be assigned *neither* true nor false or *both* true and false. Its semantics is based on the four-valued *setups* of propositional relevance logic [2], i.e. $\mathcal{B} = \{\{T\}, \{F\}, \{T, F\}, \{\}\}$. The propositional tautological entailment is defined as follows: α entails β, written $\alpha \rightarrow \beta$, iff β is true whenever α is and α is false whenever β is. This entailment is a much weaker notion than implication as known in standard propositional logic due to the four-valued *setups* which include the set of two-valued assignments. For example, $a \wedge \neg a \not\rightarrow b$ and $a \not\rightarrow b \vee \neg b$, which mean that the classical unsatisfiability and tautologies are not defined in this semantics. In this entailment *modus ponens* is not a valid rule due to $a \wedge (\neg a \vee b) \not\rightarrow b$.

Definition 1 *A situation s consists of a triplet containing a non empty set D, the domain of the situation, a function ε_s, the environment function of s, and a function ξ_s, the extension function of s, i.e. $s = \langle D, \varepsilon_s, \xi_s \rangle$. ε_s maps each function letter, f_j^n, into a function from D^n to D and ξ_s consists of a pair of functions $\langle \xi_s^+, \xi_s^- \rangle$ associating to each predicate a* positive extension, ξ_s^+, *the tuples in the domain known to possess the property of the predicate, and a* negative extension, ξ_s^-, *the tuples known not to possess this property.*

Definition 2 *A variable map is a mapping from variables into some set. If v is a variable map into D, x is a variable, and d is an element of D, then v_d^x is a variable map into D with $v_d^x(y) = d$, if $y = x$, and $v_d^x(y) = v(y)$, otherwise. Given a situation, s, and a variable map, v, a mapping, v_s^*, from terms into domain of s can be defined as follows: $v_s^*(x) = v(x)$, if x is a variable, $v_s^*(f_j^n(t_1, \cdots, t_n)) = (\varepsilon_s(f_j^n))(v_s^*(t_1), \cdots, v_s^*(t_n))$, otherwise.*

Definition 3 *The support relationships of first-order relevance logic for atomic formulae are defined as follows: $s, v \models_t A_j^n(t_1, \cdots, t_n)$ iff $(v_s^*(t_1), \cdots, v_s^*(t_n)) \in \xi_s^+(A_j^n)$, s supports the truth of $A_j^n(t_1, \cdots, t_n)$ under v, and $s, v \models_f A_j^n(t_1, \cdots, t_n)$ iff $(v_s^*(t_1), \cdots, v_s^*(t_n)) \in \xi_s^-(A_j^n)$, s supports the falsity of $A_j^n(t_1, \cdots, t_n)$ under v.*

The relationships are extended to arbitrary first-order formulae — very similar to standard tarskian semantics — by the following rules:

1. $s, v \models_t \neg\alpha$ iff $s, v \models_f \alpha$
   $s, v \models_f \neg\alpha$ iff $s, v \models_t \alpha$

2. $s, v \models_t \alpha \vee \beta$ iff $s, v \models_t \alpha$ or $s, v \models_t \beta$
   $s, v \models_f \alpha \vee \beta$ iff $s, v \models_f \alpha$ and $s, v \models_f \beta$

3. $s, v \models_t \alpha \wedge \beta$ iff $s, v \models_t \alpha$ and $s, v \models_t \beta$
   $s, v \models_f \alpha \wedge \beta$ iff $s, v \models_f \alpha$ or $s, v \models_f \beta$

4. $s, v \models_t \forall x\alpha$ iff for all $d \in D$ $s, v_d^x \models_t \alpha$
   $s, v \models_f \forall x\alpha$ iff for some $d \in D$ $s, v_d^x \models_f \alpha$

5. $s, v \models_t \exists x\alpha$ iff for some $d \in D$ $s, v_d^x \models_t \alpha$
   $s, v \models_f \exists x\alpha$ iff for all $d \in D$ $s, v_d^x \models_f \alpha$

Definition 4 *If α and β are first-order sentences, α entails β iff for all situations, s, and all variable maps, v, if $s, v \models_t \alpha$ then $s, v \models_t \beta$ and if $s, v \models_f \beta$ then $s, v \models_f \alpha$.*

The drawback of first-order tautological entailment is that it can be used to simulate first-order implication and, thus, it is *undecidable*.

Now, we deal with a variant of relevance logic. Initially, we need to introduce the notion of compatible sets of situations. A compatible set of situations is a set of situations with the same domain and the same environment function. Given S, a compatible set of situations each with domain D, and v, a variable map into D, the two support relations for this logic, $S, v \models_t \alpha$ and $S, v \models_f \alpha$ are defined as follows.

1. $S, v \models_t \forall x\alpha$ iff for all $d \in D$ $S, v_d^x \models_t \alpha$
   $S, v \models_f \forall x\alpha$ iff for some $d \in D$ $S, v_d^x \models_f \alpha$

2. $S, v \models_t \exists x\alpha$ iff for some $d \in D$ $S, v_d^x \models_t \alpha$
   $S, v \models_f \exists x\alpha$ iff for all $d \in D$ $S, v_d^x \models_f \alpha$

3. $S, v \models_t \alpha$ iff for all $s \in S \quad s, v \models_t \alpha$
   $S, v \models_f \alpha$ iff for all $s \in S \quad s, v \models_f \alpha$

The interpretation of the formula $\exists x Px$ would be: There exists a known individual for which the P is true, i.e. for some domain element x Px is true in each situation.

There are three different versions of entailment of $\alpha \rightarrow \beta$: (i) β must be true whenever α is, t-entailment (written $\rightarrow_t$), (ii) α must be false whenever β is, f-entailment (written $\rightarrow_f$) and (iii) Both conditions must be fulfilled, tf-entailment (written $\rightarrow$). The entailments for quantifiers can be expressed as follows:

$\forall x Px \rightarrow Pa$	$Pa \rightarrow \exists x Px$
$\forall x Px \rightarrow_t Pa \wedge Pb$	$Pa \vee Pb \not\rightarrow_t \exists x Px$
$\forall x Px \not\rightarrow_f Pa \wedge Pb$	$Pa \vee Pb \rightarrow_f \exists x Px$
$\forall x Px \not\rightarrow Pa \wedge Pb$	$Pa \vee Pb \not\rightarrow \exists x Px$

Thus, the t-entailment is best-suited for knowledge representation since a universal t-entails the conjunction of any number of instantiations whereas a disjunction of instantiations does not t-entails an existential [12].

Finally, using the following theorem we are able to devise a decidable algorithm to compute t-entailment as described above.

Theorem 1 *If α and β are sentences in skolemized prenex conjunctive normal form, i.e. $\alpha = \forall \vec{z} \bigwedge \alpha_j$ and $\beta = \exists \vec{x} \bigwedge \beta_i$ where $\vec{z}$ is some ordering of the universally quantified variables in α and $\vec{x}$ is some ordering of the existential quantified variables in β, then $\alpha \rightarrow_t \beta$ iff there exists θ, a substitution for $\vec{x}$, such that for each β_i there exist some α_j and ψ, a substitution for $\vec{z}$, such that $\alpha_j \psi \subseteq \beta_i \theta$ where α_j and $\beta_i \theta$ are treated as sets of literals.*

The semantics of for first-order logic, frames and semantic network definitions and inferences, as well as the hybrid inference algorithms are given in [3].

3 The Epistemological Level

The lowest level of the architecture contains three representation methods. The knowledge is represented as formulae, concepts and relations, as well as hierarchies and classes, which can be specified through a lisp-like syntax.

Logic

This method is intended to be used to represent logic knowledge about a particular domain. It is adequate to domains where the knowledge is largely unstructured and consists of a collection of independent facts. The main drawback of the logic formalism

is the inherent inefficiency of the inference method: automatic deduction. Most of the existing hybrid systems [4] contain a first-order logic method, because of its advantages: (i) naturality of the representation, (ii) modularity and flexibility of the represented knowledge which can easily be modified and extended, and additionaly, (iii) classical logic has a formal semantics due to Tarski.

Adopting the t-entailment as defined above, a four-valued semantics for the first-order logic of this method can be introduced. However, the syntax of the formulae remains the same as in the classical case.

To give an idea of the results of the entailment calculation we sketch the algorithm performing this task: Given a set of asserted facts, F_i, and a question $Q = \bigwedge_i Q_i$, the algorithm searches for the set of all substitutions such that, for each Q_i in the query, there is at least one F_i which implies, according to the classical semantics, this Q_i when one of the substitutions is applied. Once this set is calculated the algorithm tries to find a compatible subset, i.e. where the same variables are substituted by the same terms. If this subset is not empty then we say that F_i entails Q.

Frames

This method is intended to be used to represent a terminology by means of concepts, the categories of objects, and relations, the properties of objects. The formalism is adequate to taxonomically structured domains where the inheritance mechanism can be efficiently explored. The characteristics of technical knowledge - such as machine descriptions, process descriptions, technical terms, troubleshooting strategies, etc - make the frame formalism a preferential choice when building expert system knowledge bases for those domains [8].

The notion of relations is an extension of the notion of roles, usually used in terminological languages. Roles are binary relations and relations are arbitrary n-place relations. The main idea of extending roles is that it provides a better integration of this method with the logic method: The correspondence of n-place relations to n-ary predicates. The principal operation in this method is the subsumption relation which verifies whether a concept or relation subsumes another concept or relation.

The terminological language embedded into the system has some additional characteristics usually not possessed by the terminological languages or hybrid systems: (i) It possesses a rich set of primitives, including disjunction and negation of both concepts and relations, (ii) It provides special symbols for the universal concept and for the bottom concept as well as for the universal relation and for the bottom relation and (iii) It includes tests for subsumption and for equality between concepts and between relations.

Semantic Nets

A semantic network consists of a set of nodes connected by a set of links. Its main inference mechanism is inheritance through the network. The flexibility of the formalism

makes it a good choice for expert system knowledge representation, but the user must follow some discipline in order not to misuse the formalism capacities. The method manipulates the notions of classes and hierarchies. The hierarchies can be explicitly created by defining links among classes. Two types of links are provided: *Default* links and *Exception* links. The hierarchies are used as inheritance paths between classes. The main inference procedure of this method calculates the *Subclasses* relation taking into account the explicit exception.

4 The Logical Level

One of the main features of MANTRA is to allow the definition of powerful hybrid inference algorithms which enlarge the deductibility power of any of the single formalisms available in the epistemological level. The two main functions are Tell and Ask.

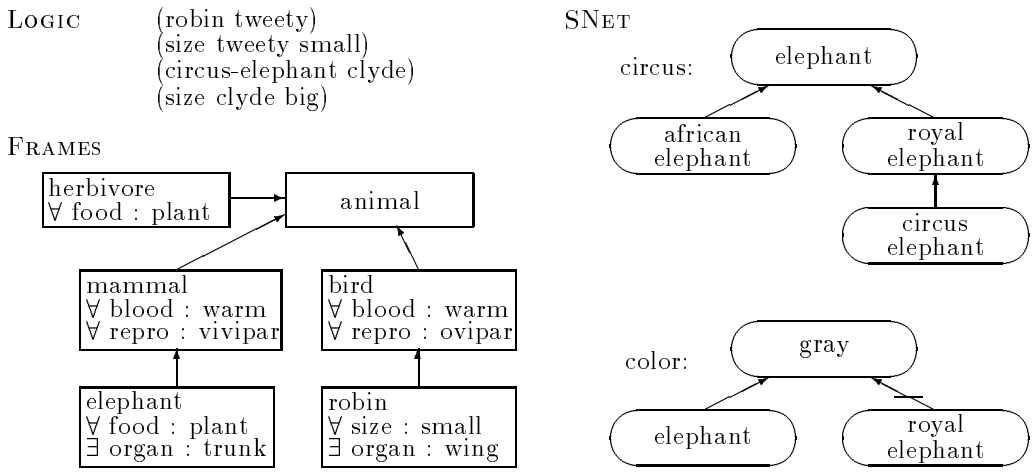
The queries to the knowledge bases can be formed either by using a specific method or by using one of three combinations of the methods currently available: logic+frame, logic+snet and frame+snet. The idea of the interaction between the three methods is that the functionalities of one method can be used in order to increase the inference power of another method. For example, to bypass the invalidity of modus ponens in the logic method one can use the frame or the semantic networks method to represent the chaining of a predicate in an appropriate way. In this way, the user is given a possibility not only to represent a specific domain by means of several knowledge representation methods but the user can also make use of the hybrid reasoning in order to get a semantically motivated answer from a specific knowledge base.

Using the logic method and the terminological method one can make use of the hybrid reasoning possessed by the system. The idea of the interaction algorithm is to determine all the frame entities subsumed by the predicates appearing in a logical question and to use this subsumed entities as they were predicates t-entailed by the original predicates.

The interaction algorithm for the logic and semantic network methods is very similar to the frame case, but in the present case a hierarchy is used to represent an explicit entailment between first-order logic predicates according to the subclass relation represented in the hierarchy.

The next hybrid reasoning is the interaction between frame and semantic network methods. The idea of this algorithm is to explicitly construct the subsumption graph of the frame hierarchy, and to use the union of this graph and of the given hierarchy graph to calculate subsumptions of primitive concepts during the subsumption calculation. The next example presents this hybrid reasoning.

As mentioned previously, the three inference problems, t-entailment, subsumption and subclass, and their combinations are decidable. The following simple examples illustrates some of the capabilities.



The content of a knowledge base is defined at the epistemological level. These knowledge bases can be managed, manipulated and interrogated by using the inference algorithms of the logical level. Given the previous example, one can ask the following questions:

```
(Ask 'kbase '(Exist x (And (size x small) (robin x))))
--> yes   (x . tweety)

(Ask 'kbase '(Subsume-concept herbivore elephant))
--> yes

(Ask 'kbase '(Sub-class (Union circus color) circus-elephant gray))
--> no

(Ask 'kbase '(Union circus color)
            '(Exist x (And (size x big) (Not (gray x)))))
--> yes  (x . clyde)

(Ask 'kbase 'circus '(Subsume-concept animal african-elephant))
--> yes
```

5 The Heuristic Level

To combine the advantages of both declarative and procedural knowledge we introduce another paradigm into the system: production rules. They allow the definition of procedural knowledge of a domain using the represented declarative knowledge [10].

At this level, the primitives that allow the definition of production systems for the automatic manipulation of knowledge bases are defined. The syntax of the language at this level is given below. A rule of a rule base is made up of the following parts: (i) rule identifier, (ii) a list of variables, (iii) condition part and (iv) action part. The condition and the action parts mainly rely on the **Tell** and **Ask** primitives as defined at the logical level. We allow the user to encapsulate a set of rules in a context, i.e. the rules are valid or can fire if the context is active. Activating, or deactivating, a

context can be performed by an appropriate primitive embedded in the action part. The major goal of introducing such contexts is to exclude "redundant" rules while the rule interpreter is selecting rules to be executed, i.e. to minimize the set of conflict rules.

The interpretation of rules is performed by invoking the primitive **Execute**. Presently, the interpretation is performed by means of forward chaining. The conflict resolution strategy can explicitly be given by the user. Three kinds of strategies, which are embodied by the following three filters, are currently available and given in figure 2.

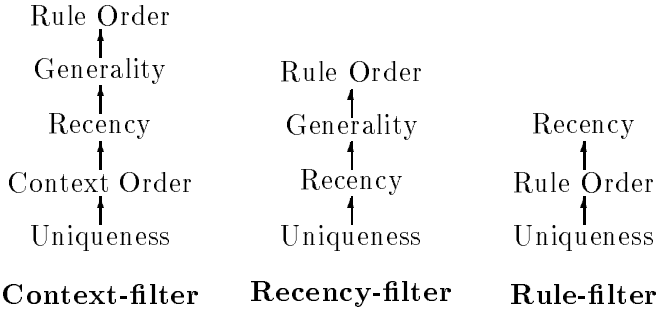


Figure 2: Conflict Resolution Strategies

The meaning of each strategy, e.g. recency or generality, is defined as usual [1].

Moreover, the user can explicitly determine the flow strategy which is either **all-rules** (breadth-first search) or **first-rule** (depth-first search).

The lisp-like syntax of the language at this level including the explanation facilities is the following:

heuristic-level ::= *rbase-declaration* | *rbase-statement*
rbase-query |
interpret |
explanation

rbase-declaration ::= (Decl-rbase *identifier* $\cdots$ *identifier*)

rbase-statement ::= (Tell-rbase *identifier context* $\cdots$ *context*) |
(Tell-rbase *identifier identifier*) |
(Remove-rbase *identifier* $\cdots$ *identifier*) |
(Remove-context *identifier identifier* $\cdots$ *identifier*) |
(Remove-rule *identifier identifier* $\cdots$ *identifier*) |
(Rule-order *identifier identifier* $\cdots$ *identifier*) |
(Context-order *identifier identifier* $\cdots$ *identifier*)

rbase-query ::= (Ask-rbase *identifier*) |
(Ask-context *identifier identifier*) |
(Ask-rule *identifier identifier*)

interpret ::= (Execute *identifier identifier*) |

(Execute *identifier identifier goal*) |
(Execute *identifier identifier flow-strategy goal*) |
(Execute *identifier identifier goal flow-strategy conflict-strategy*)

flow-strategy ::= all-rules |
first-rule

conflict-strategy ::= context-filter |
recency-filter |
rule-filter

context ::= (*identifier rule* $\cdots$ *rule*) |
(*rule* $\cdots$ *rule*)

goal ::= (*condition-part*)

rule ::= (*identifier variables condition-part action-part*) |
identifier

variables ::= (*identifier* $\cdots$ *identifier*) | ()

condition-part ::= *condition* $\cdots$ *condition*

condition ::= *identifier* : *kbase-rule-quest* |
kbase-rule-quest

action-part ::= *action* $\cdots$ *action*

action ::= *identifier* : *kbase-definition restriction* |
identifier : *kbase-definition* |
kbase-definition restriction |
kbase-definition |
(to-lisp *lisp-expression*) |
(activate *identifier* $\cdots$ *identifier*) |
(deactivate *identifier* $\cdots$ *identifier*)

restriction ::= (to-context *identifier* $\cdots$ *identifier*)

explanation ::= (Explain how *identifier*) |
(Explain when *identifier*) |
(Explain why *identifier*) |
(Explain history)

kbase-rule-quest and *kbase-definition* coincide with the language of **Ask** and **Tell** primitives, respectively.

However, in opposite to usual rule-based systems the database consists of powerful

inference algorithms of the logical level as shown in figure 3.

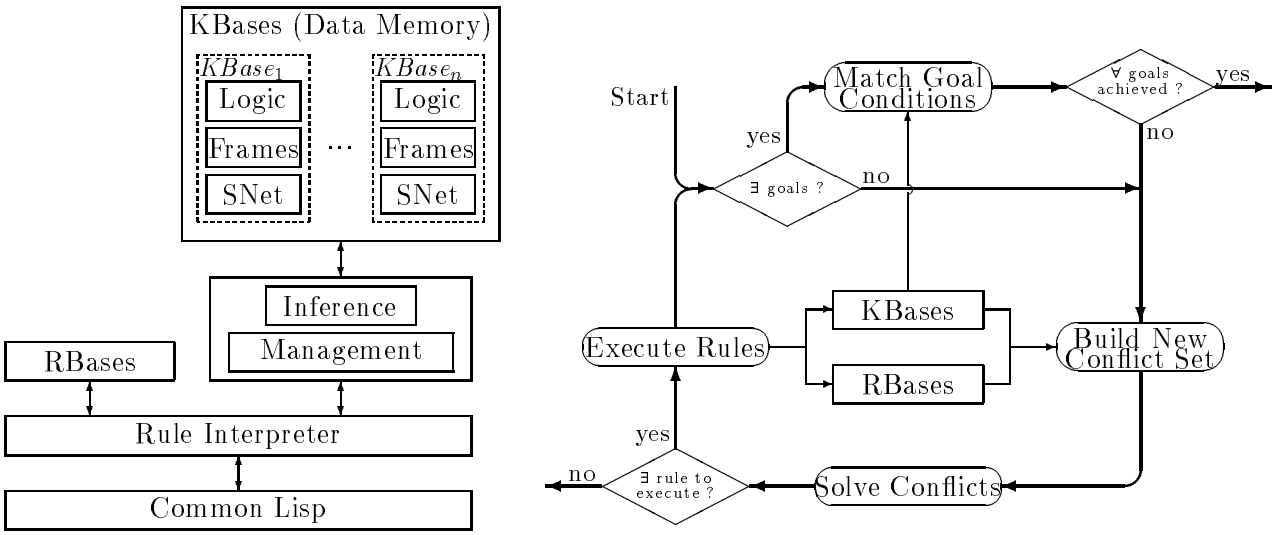


Figure 3: Rule based systems with MANTRA & MANTRA's recognize-act cylce

Furthermore, ad hoc rules for the utilisation of the primitives of the logical level can be introduced. The rule interpreter consists of the recognize-act-cycle.

An important question concerns the efficiency to set up the matching rules to resolve conflicts. To avoid to match every condition of every rule in each cycle, three methods are available. First, rules may be grouped into contexts which may be activated and deactivated. This reduces the number of applicable rules. Secondly, common conditions are detected and tested only once. This reduces the number of applicable conditions. Finally, the execution of rules changes only the stored knowledge incrementally. For this reason, only those conditions have to be matched, whose underlying predicates, concepts, hierarchies ... have changed. This reduces the amount of possible new conditions drastically.

As for the knowledge bases, rules are stored as rule bases which can be managed, manipulated and executed using the heuristic level primitives. Additionaly, explanations about the history of the applications of rules are provided.

6 The Implementation

Due to the interconnections among the different epistemological methods a single data abstraction consisting of directed graphs has been selected. The system has been implemented in Common Lisp and uses the object-oriented extension CLOS to define and manipulate the knowledge bases. This increases the modularity of the system and makes it easy to modify.

In the earlier version of this system, the interface had been developed using KYACC-KLEX, an interface between KCL and the compiler generator YACC and LEX. Due to the portability difficulty of YACC and LEX we have replaced this part by an LALR

syntax analyzer that we have implemented directly in KCL. The use of an object-oriented programming paradigm increases the modularity of the system and makes is easy to modify. To facilitate the interconnection between the different methods a single data abstraction has been adopted. This data abstraction consists of a set of *Directed Graphs*. Directed graphs subsumes several of the most commonly used data structures and is also suitable to be used in an interactive system due to their inherent graphical character. The system Grasp, a graph manipulation package, has been adopted as the programming tool implementing this data abstraction.

The efficient unification algorithm of Martelli and Montanari [11] is at the core of each inference procedure.

A graphical user's interface allowing the visualization and definition of knowledge is provided. It has been implemented in C with XToolkit, is very fast, and interacts with MANTRA through a C$\leftrightarrow$Lisp interface. A graph editor allows to browse concepts, frames and hierarchies, and a syntax-driven mask editor allows easily the correct specification of new knowledge.

7 Conclusion

The main characteristics of MANTRA are the interactions between different representation formalisms, and the embedded semantics which is sound and complete. Additionaly, the introduction of production rules allows the construction of expert systems using the knowledge through powerful inference algorithms.

One of many important enhancements to be achieved concerns the user interface. We are implementing a cooperative graphical user interface for MANTRA based on X-Windows. A graph editor which can be used to visualize, for instance, hierarchies or terminologies would aid the user for representing expert's knowledge by means of frames or semantic networks. The other possible interactions among the methods are also being implemented. The theoretical studies of these interactions have shown that the algorithms being implemented are sound and complete according to the four-valued semantics. The rule interpreter, in the heuristic level, is also being extended to be capable of performing backward chaining.

The semantic soundness is mandatory for one of the application of MANTRA which is to design an environment for mathematical knowledge representation suitable for Computer Algebra Systems [7] [6]. Considering the fact that mathematical domains of computation are inherently modular and that there are inter-relationships among the domains computer algebra is seen , in this environment, as another sort of knowledge that is called mathematical knowledge. The proposed representation of mathematical domains of computation is based on the notion of abstract computational structures. In this way we make use of all knowledge representation methods in order to represent mathematical domains, e.g. the logic method to represent the laws of an abstract domain, the frame method to represent the terminologies of a domain and the semantic network method for representing the whole hierarchy.

This application to symbolic computation in Mathematics is an on-going project where one represents and manipulate highly non-trivial knowledge. MANTRA is proving itself to be very well suited to such an elaborated application. The shell concept on which MANTRA is based enables also to use it to develop expert systems. Another possible application lies in teaching Knowledge Representations to students since it encompasses several cooperating formalisms.

The soundness of the semantics was necessary for one of our applications which is to use the system as part of an intelligent environment, where computer algebra and theorem proving techniques can be combined. The mathematical knowledge can be specified as abstract computational structures [5] [6] [7] and is represented using the different formalisms. The development of this environment is an ongoing project.

Another application just under way is to use the decidability features of the four-valued semantics in the domain of distibuted AI.

References

[1] A. Barr, E.A. Feigenbaum, *The Handbook of Artificial Intelligence*, William Kaufmann, 1982.

[2] N.D. Belnap, *A Useful Four-Valued Logic*, in "Modern Uses of Multiple-Valued Logic", ed. G. Epstein and J.M. Dunn, Boston: Reidel, 1977.

[3] G. Bittencourt, *An Architecture for Hybrid Knowledge Representation*, Ph.D. Dissertation, University of Karlsruhe, Department of Computer Science, 1990.

[4] R.J. Brachman, V.P. Gilbert, H.J. Levesque, *An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON*, Proceeding of IJCAI 9, 1985.

[5] J. Calmet, K. Homann, I.A. Tjandra, *Unified Domains and Abstract Computational Structures*, Proceedings of Conference on Artificial Intelligence and Symbolic Mathematical Computations, LNCS 737, Karlsruhe, August 3 – 6, 1992, Springer.

[6] J. Calmet, I.A. Tjandra, *An AI Environment for Computer Algebra*, in J. Johnson (Ed.), Proceedings of International Conference on Artificial Intelligence in Mathematics, Glasgow, 1991 (to appear in 1994).

[7] J. Calmet, I.A. Tjandra, G. Bittencourt, *A Framework for Representing Algebraic Knowledge Using a Hybrid Knowledge Representation System*, Proceeding of The Fourth International Symposium on Knowledge Engineering, 1990.

[8] R.E. Fikes, T. Kehler, *The Role of Frame-Based Representation in Reasoning*, Communications of the ACM, Vol. 28, No. 9, pp. 904-920, September 1985.

[9] A.M. Frisch, *Knowledge Retrieval as Specialized Inference*, Report No. 214, University of Rochester, Department of Computer Science, 1987.

[10] A. Lulay, K. Homann, *Entwurf und Implementierung der heuristischen Ebene eines hybriden Wissensrepräsentationssystems*, Diplomarbeiten, Universität Karlsruhe, 1991.

[11] A. Martelli, U. Montanari, *An Efficient Unification Algorithm.* ACM Transactions on Programming Languages and Systems, Vol. 4, No. 2, April 1982.

[12] P.F. Patel-Schneider, *A Decidable First-Order Logic for Knowledge Representation*, Proceeding of IJCAI 9, 1985.

[13] P.F. Patel-Schneider, *A four-Valued Semantics for Frame-Based Description Languages*, Proceeding of AAAI-86, 1986.

[14] R.H. Thomason, J.F. Horty, D.S. Touretzky, *A Calculus for Inheritance in Monotonic Semantic Nets*, Technical Report CMU-CS-86-138, Carnegie-Mellon University, Department of Computer Science, 1986.