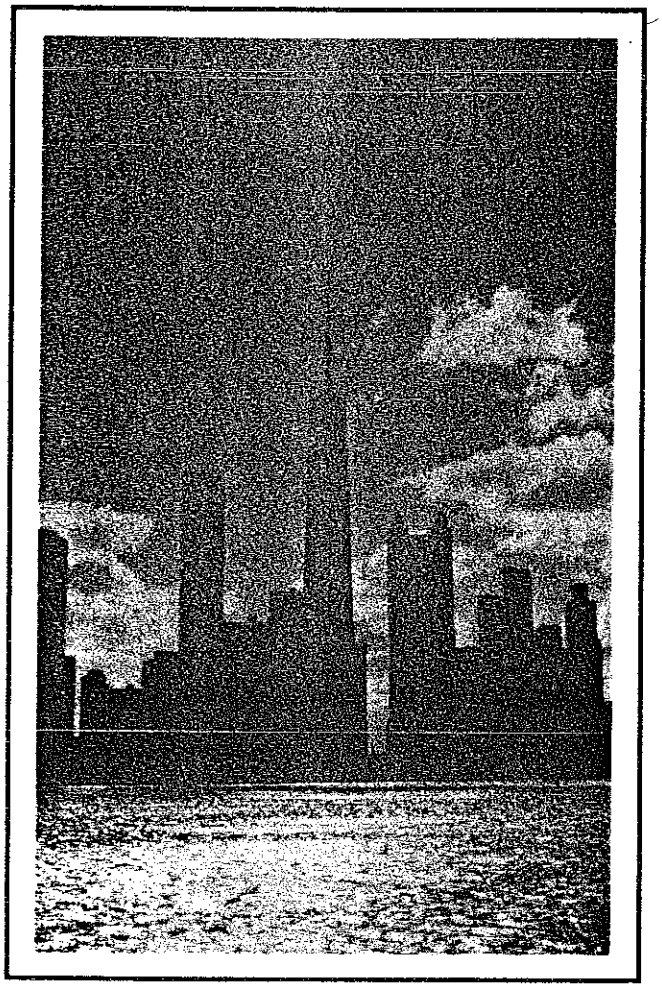
1992 IEEE International Conference on Systems, Man, and Cybernetics



Emergent Innovations in Information Transfer Processing and Decision Making

The Knickerbocker Hotel
Chicago, Illinois

October 18 - 21, 1992

Volume 1 of 2
92CH3176-5

IEEE Systems, Man and Cybernetics Society

Modularity, Uncertainty and Reflection in MILORD II

Carles Sierra, Lluís Godo
Institut d'Investigació en Intel·ligència Artificial (IIIA), CSIC
Camí de Santa Bàrbara, 17300 Blanes, Spain
e-mail: sierra@ceab.es, godo@ceab.es

Abstract **Knowledge Based Systems, when programmed in the large, require special architectures, adapted to implement complex reasoning tasks and able to combine simple tasks into more sophisticated ones in a safe way. To tackle this problem MILORD II proposes three basic programming techniques: modularisation, reflection and local uncertainty management. *Modularisation* is a technique used to map the task/subtask structure of a problem into a structured KB. Modules consist of a clean interface, a propositional object-level language and a first order meta-level language. *Uncertainty* is managed by means of finite multiple-valued local logics associated to the object-level language of each module. *Reflection* between the object-level and the meta-level languages allows to implement, inside each module, non-standard reasoning patterns such as default reasoning or hypothetical reasoning. The combination of several modules into a structured KB is the way MILORD II implements complex reasoning patterns.**

I. INTRODUCTION

Reasoning patterns that appear when modelling complex tasks cannot often be modelled only by means of a pure classical logic approach. This is due to several reasons - incompleteness of the available information, need of using and representing uncertain or imprecise knowledge, combinatorial explosion of classical theorem proving when knowledge bases become large, or a lack of methodology in building complex and large knowledge bases among them. In this paper we present a system (MILORD II) that proposes particular solutions to these problems.

Incompleteness of the available knowledge may lead to make assumptions in a deduction process, even if later on these assumptions are proved to be erroneous. To deal with non-monotonicity a meta-level approach, based on reflection techniques and equipped with a declarative backtracking mechanism is provided by the system. The use of reflection techniques is a common practice in several knowledge areas, natural language, philosophy, literature, etc. [4]. The application of reflection techniques to KBS has been widely used in the recent past as a clear separation between domain and control knowledge [6,11]; however, few systems have clear semantics, among which we can find OMEGA [6] and BMS [10].

Due to the need in AI of dealing most of the times with uncertain and/or imprecise information, our system provides at the object-level a family of representation languages based on multiple-valued logics, where sets of truth-values stand for scales of linguistic terms representing different degrees of uncertainty or belief.

Modularisation is a standard technique to manage the complexity of highly interacting systems, such as KBSs. A module can be understood as a functional abstraction, by fixing both the set of components it needs as input, and the type of results it can produce. This technique has been used in the context of functional programming, Standard ML [5], and Logic Programming [8]. In the architecture described in this paper - MILORD II - both techniques, reflection and modularisation, are mixed in order to be able to define complex reasoning patterns in the large.

In figure 1 you can see how a KB looks like in MILORD II. It consists of a set of hierarchically interconnected modules, each one containing an Object-Level Theory (OLT) and a Meta-Level Theory (MLT) interacting through a reflective mechanism.

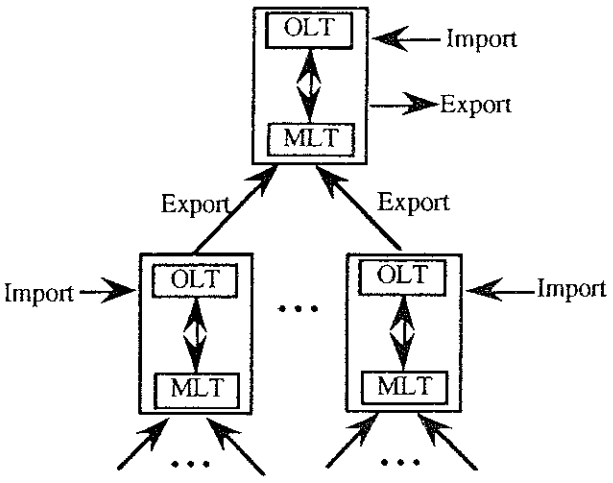


Figure 1. Milord II KB structure.

Each module has also an import/export interface. The user of a KB provides information to modules via the import in-

Manuscript received August 1, 1992. This work was supported in part by the ESPRIT BRA 3085 DRUMS.

0-7803-0720-8/92 $3.00 ©1992 IEEE

terface, and modules provide information to the user or to other modules via the export interface.

The paper is structured as follows. In the second section the object-level, based on multiple-valued logics, is presented and formalised. The third section is devoted to the meta-level language, the reification and reflection processes, and the dynamics of the execution of a module. Finally, in Section 4 a brief description of the MILORD II architecture is given.

II. UNCERTAINTY

Reasoning at the object-level in one module is generically based on the use of a multiple-valued logic. A particular logic can be specified inside a module by defining which is the algebra of truth-values, i.e. which is the ordered set of truth-values and which is the set of logical operators associated to them. The kind of algebras of truth-values used in MILORD-II are those defined as follows.

Definition: An ordered **Algebra of truth-values** is a finite algebra $\mathbf{A_{n,T}} = \langle A_n, N_n, T, I_T \rangle$ such that:

1) The ordered set of truth-values A_n is a chain of n elements:

$$0 = a_1 < a_2 < \ldots < a_n = 1$$

where 0 and 1 are the booleans *False* and *True* respectively.

2) The negation operator N_n is a unary operation defined as $N_n(a_i) = a_{n-i+1}$, the only one that fulfils the following properties:

N1: if $a < b$ then $N_n(a) > N_n(b)$, $\forall a \in A_n$, $\forall b \in A_n$

N2: $N_n^2 = Id$.

3) The conjunction operator T is any binary operation such that the following properties hold $\forall a, b, c \in A_n$:

T1: $T(a,b) = T(b,a)$

T2: $T(a,T(b,c)) = T(T(a,b),c)$

T3: $T(0,a) = 0$

T4: $T(1,a) = a$

T5: if $a \leq b$ then $T(a,c) \leq T(b,c)$ for all c

4) The implication operator I_T is defined by residuation with respect to T, i.e.

$$I_T(a,b) = \text{Max}\ \{c \in A_n \text{ such that } T(a,c) \leq b\}$$

Such an implication operator satisfies the following properties:

I1: $I_T(a,b) = 1$ if, and only if, $a \leq b$

I2: $I_T(1, a) = a$

I3: $I_T(a, I_T(b,c)) = I_T(b, I_T(a,c))$

I4: if $a \leq b$, then $I_T(a, c) \geq I_T(b,c)$ and $I_T(c,a) \leq I_T(c,b)$

I5: $I_T(T(a,b), c) = I_T(a, I_T(b,c))$

As it is easy to notice from the above definition, any of such truth-value algebras is completely determined as soon as the set of truth-values and the conjunction operator T are determined. So, varying these two characteristics we can obtain a parametric family of different multiple-valued logics. In this way, each module can have a local type of reasoning, potentially different from others. Additionally, in order to allow a flux of information between modules and submodules with different local logics, renaming mappings between sets of truth-values can also be specified inside a MILORD II module.

In classical logic we distinguish syntactically A from $\neg$A (A being a formula) because the only two possible truth-value assignments to A are *true* and *false*. Similarly, in our case, we need a syntactical representation for each truth-value assignment to a formula A. Moreover, to handle imprecision, intervals of truth-values are attached to formulas instead of single values: the more imprecision there is, the larger the intervals are. Then, the language we propose is such that sentences are pairs of type (p, V) where p is a classical-like sentence and V is an interval of truth-values, possibly empty (represented as []). This representation leads to single satisfaction and entailment relations (see §II.B), rather than having classical sentences but several satisfaction and entailment relations, as for example in [10]. Another interesting characteristic of the object-level reasoning system is that deduction is based on what we call "Specialisation Inference Rule" (SIR), a more general inference rule than Modus Ponens, introduced in [7] to improve the input/output communication behaviour of the system.

In the sequel, the syntax, the semantics and the deduction system of the object-level logic are described given a particular algebra of truth-values $\mathbf{A_{n,T}} = \langle A_n, N_n, T, I_T \rangle$.

A. *Syntax*[1]

The propositional language $\mathbf{OL_n} = (A_n, \Sigma_O, C, OS_n)$ of the object-level is defined by:

- A **Signature** Σ_O, composed of a set of atomic symbols plus *true* and *false*.
- A set of **Connectives** $C = \{\neg, \wedge, \rightarrow\}$
- A set of **Sentences** OS_n whose elements are pairs of classical-like propositional sentences and intervals of truth-values. The classical-like propositional sentences are restricted to literals and rules. That is, the sentences of the language are only of the following types:

OL-literals: (p,V)

OL-Rules: $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q, V^*)$[2],

being $p_i \neq p_j$, $p_i \neq \neg p_j$, $q \neq p_j$, $q \neq \neg p_j$ $\forall i,j$

where p, p_1, p_2, ..., p_n and q are literals (atoms or negations of atoms), V and V^* are intervals of truth-values. Intervals V^* for rules are constrained to the upper intervals, i.e. of the

[1] The syntax used in this Object Level Language description is a simplification of the actual MILORD II syntax.

[2] The corresponding rule in MILORD II syntax would be:
If p_1 **and** p_2 **and** ... **and** p_n **then conclude** q **is** V^*

form [a,1], where $a > 0$. In the sequel, and for the sake of simplicity, we will identify intervals of type *[a, a]* with the value *a*.

B. Semantics

The semantics is basically determined by the connective operators of the truth-value algebra $A_{n,T}$. Having truth-values explicit in the sentences enables us to define a classical satisfaction relation in spite of the models being multiple-valued assignments.

- **Models** $M\rho$ are defined by valuations ρ, i.e. mappings from the first components of sentences to A_n such that:

$\rho(true) = 1$ $\quad$ $\rho(false) = 0$

$\rho(\neg p) = N_n(\rho(p))$ $\quad$ $\rho(p_1 \wedge p_2) = T(\rho(p_1), \rho(p_2))$

$\rho(p \rightarrow q) = I_T(\rho(p), \rho(q))$

- The **Satisfaction Relation** between models and sentences is defined by:

$M_\rho \models_0 (p, V)$ iff $\rho(p) \in V$

C. Deduction system

The object-level deduction system is based on the following axiom schemes:

(AS-1) $(\neg\neg p \rightarrow p, 1)$ $\quad$ (AS-2) $(p, [0, 1])$

the following axioms:

(A-1) (true, 1) $\quad$ (A-2) (false, 0)

and on the following inference rules:

(RI-1) **weakening**: $(p,V_1) \vdash_0 (p,V_2)$ where $V_1 \subseteq V_2$

(RI-2) **not-introduction**: $(p,V) \vdash_0 (\neg p, N^*_n(V))$

(RI-3) **composition**: $\{(p,V_1), (p,V_2)\} \vdash_0 (p,V_1 \cap V_2)$

(RI-4) **SIR**:

$\{(p_i,V_i),(p_1 \wedge ... \wedge p_{i-1} \wedge p_i \wedge p_{i+1} \wedge ... \wedge p_n \rightarrow q,V_r)\}$

$\vdash_0 (p_1 \wedge ... \wedge p_{i-1} \wedge p_{i+1} \wedge ... \wedge p_n \rightarrow q, MP^*_T(V_i,V_r))$.

N^*_n and MP^*_T stand for the point-wise extensions[1] of N_n and MP_T respectively. MP_T is the binary function from A_n to the set $I(A_n)$ of intervals of A_n defined as:

$$MP_T(a,b) = \begin{cases} [], & \text{if } a \text{ and } b \text{ are inconsistent} \\ [a,1], & \text{if } b = 1 \\ T(a,b), & \text{otherwise} \end{cases}$$

where *a* and *b* are inconsistent if there exists no *c* such that $I(a,c) = b$. $MP_T(a, b)$ provides the set of all solutions in A_n for $\rho(q)$ in the following equation system:

$$\begin{cases} \rho(p) = a \\ \rho(p \rightarrow q) = b \end{cases}$$

which corresponds to a multiple-valued version of the classical Modus Ponens inference rule.

[1] Actually, MP^*_T is defined to give the minimal interval containing the point-wise extension.

It is easy to check that this deductive system is sound.

Theorem (Soundness). *Let A be a sentence and Γ a set of sentences. Then, $\Gamma \vdash_0 A$ implies $\Gamma \models_0 A$.*

III. META-LEVEL AND REFLECTION

A. The meta-level

The **meta-level language** is a restricted classical first order language $ML = (\Sigma_M, C, MS)$ defined by:

- A **Signature** $\Sigma_M = (\Sigma_{rel}, \Sigma_{fun}, \Sigma_{con}, \Sigma_{var})$, where:

Σ_{rel} = A set of atomic predicate identifiers plus *Ass*, *Res*, *K*, *WK* and *P* (with special semantics, see §III).

Σ_{fun} = A set of classical arithmetic function symbols.

Σ_{con} = A set of constants including the truth-values and object propositional symbols.

Σ_{var} = A set of variable symbols; it can be empty.

- The same set of **connectives** C as in the object language.
- A set of **sentences** $MS = MLL_G \cup MR$ where:
  1.- MLL_G is the set of ground literals, in a classical sense, from Σ_M.
  2.- MLR is the set $\{p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q \mid p_i$ and q are literals from $\Sigma_M\}$ of meta-rules, where every variable occurring in q must occur also in some p_i. Variables in meta-rules, if any, are considered universaly quantified.

The **semantics** of the language is a first order classical one. The meaning of the special predicates *K*, *WK* and *P* will be explained when defining the reification correspondence (§III.C), which gives sense to these predicates as a representation of object-level sentences. The meaning of *Ass* and *Res* predicates is out of the scope of this paper. They are used to implement hypothetical reasoning via a declarative backtracking mechanism. The interested reader is referred to [9].

The **deduction** system is based on only one inference rule:

$$p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q,\ p_1', p_2', ..., p_n' \vdash_{\mathcal{M}} q'$$

where $p_1'... p_n'$ are ground instances of $p_1 ... p_n$ respectively, such that there exists a unifier σ for $\{p_1 \wedge p_2 \wedge ... \wedge p_n, p_1' \wedge p_2' \wedge ... \wedge p_n'\}$, and $q' = q\sigma$ is the ground instance of q resulting from σ.

B. Dynamics of the reasoning process

A complex KB consists of a hierarchy of modules. Each module contains an Object-level Theory (OLT) and a Meta-Level Theory (MLT). The goal of a module is to compute the most precise truth intervals for the propositions contained in its export interface. Then, a module execution consists of the reasoning process necessary to compute the truth intervals for some of the propositions in the export interface, those the user is interested in.

The execution of a module can activate the execution of its submodules in the hierarchy. These executions only interact with the parent module through the export interface of the submodules, giving formulas back as result. So, submodule execution extends the OLTs of modules by adding to them the formulas returned. It is worth noticing that the interaction is made only at the object-level.

The reasoning process will be described in terms of variations of the Object-level Theories[1] and Meta-Level Theories. The initial OLT of a module consists of its set of rules, i. e. a partial KB. It is the same for the MLT, which initially consists of the set of meta-rules.

Definitions

- We call an *object-level elementary extension* the extension of the OLT by a single literal. This can be done either by inference on the OLT, or by importing a piece of data either from the user (as specified in the import interface), from a submodule OLT or from the current MLT.
- We call a *meta-level elementary extension* the extension of the current MLT by the set of ground literals resulting from the application of a single meta-rule over all the possible instantiations of its premise.

The communication between OLTs and MLTs is done through reification and reflection processes that are explained in detail in the next subsection. Here, and as a rough introduction, we can say that a reification process maps sentences of the object-level into sentences of the meta-level, whereas a reflection process maps meta-level sentences into object-level sentences, verifying reify(reflect(A)) = A.

The reasoning dynamics follows the next scheme.

STEP 1: The user selects a module to begin the system execution. The current OLT is the set of rules of the module, the current MLT is the set of meta-rules plus the instances of the user defined meta-predicates. These instances are given when defining the dictionary of a module (see §IV.A).

STEP 2: The reasoning process starts at the object-level with the current OLT. If no elementary extension of the OLT is produced then STOP, otherwise the reasoning process control is passed to the meta-level (STEP 3).

STEP 3: When the meta-level gets the control it builds the current MLT as the previous MLT plus the axioms coming from the reification of the current OLT. Then the meta-level reasoning process is activated. If no extension of the MLT can be obtained, the control is passed back to the object-level without extending the current OLT, i.e. the reflection does not modify the current OLT. On the other hand when an inference can be performed, and thus a meta-level elementary extension is made, the control is passed to the object-level extending the current OLT by adding the reflection of the computed extension of the MLT[2]. In any case the control goes to STEP 2.

This dynamic process goes on till no possible extension of the OLT can be made.

C. Reification

The reification correspondence relates a subtheory of the OLT with the set of ground literals of the meta-language MLL_G.

Definition: Given a OLT, we define the minimal literal theory OLT_I* as:

$$OLT_I* = \{ (p, W) \mid p \text{ literal}, W = \cap\{V_i \mid (p, V_i) \in OLT\}\}$$

A small set of meta-predicates are necessary to relate the OLLs with MLL_G. For each one the corresponding *reflection rules*[3] are defined. Given that the constant names used in the MLT are exactly the same as those used in the OLT as proposition names, the renaming operation is omitted.

Meta-predicate K: *K(p, V)* means that *V* is the minimal interval such that the proposition *(p, V)* belongs to the OLT. There is a close world assumption on this predicate.

$$\frac{(p, V) \in OLT_I*}{\vdash_{\mathcal{M}} K(p,V)}$$

$$\frac{(p, V) \notin OLT_I*}{\vdash_{\mathcal{M}} \neg K(p,V)}$$

Meta-predicate WK: *WK(p,V)* means that *(p, V)* is deducible in the OLT, i.e. $OLT \vdash_O (p, V)$. $\neg WK(p, V)$ means that $OLT \vdash_O (p, V')$ with $V' \neq [0, 1]$ but $V' \not\subseteq V$.

$$\frac{(p, V) \in OLT_I* \text{ and } V \subseteq V^*}{\vdash_{\mathcal{M}} WK(p,V^*)}$$

$$\frac{(p, V) \in OLT_I* \text{ and } V \not\subseteq V^*}{\vdash_{\mathcal{M}} \neg WK(p,V^*)}$$

Meta-predicate P: *P(p)* means that *(p, V)* belongs to the deductive closure of OLT being $V \neq [0, 1]$. If at the moment of the reification the computing of the deductive closure for *p*

[1] Theories are considered not closed under deduction. So, they are closer to the concept of **presentation**.

[2] In fact, the object-level will immediately pass the control to the meta-level. This is the implicit iteration construct provided in MILORD II.

[3] We use this name, according to the literature, for both the upwards and downwards inference rules.

is not finished neither $P(p)$ nor $\neg P(p)$ will be generated.

$$\frac{\text{OLT} \vdash_{\mathcal{O}} (p, V) \text{ and } V \neq [0, 1]}{\vdash_{\mathcal{M}} P(p)}$$

$$\frac{\text{OLT} \nvdash_{\mathcal{O}} (p, V) \text{ and } V \neq [0, 1]}{\vdash_{\mathcal{M}} \neg P(p)}$$

With these reflection rules we can differentiate between different sorts of partiality in the information: propositions provisionaly unknown, $\{ p \mid \vdash_{\mathcal{M}} K(p, [0, 1]) \}$, and propositions that are definitively unknown because they cannot be proven $\{ p \mid \vdash_{\mathcal{M}} \neg P(p) \}$.

D. Reflection

The reflection process maps the meta-level theories into object-level literals. Only instances of the *K* predicate are considered, because the other related meta-predicate *Ass* is transformed into several instances of the *K* predicate in different MLT extensions, see [9]. The reflection rule that relates MLT with OLT is defined as:

$$\frac{\vdash_{\mathcal{M}} K(p, V)}{\vdash_{\mathcal{O}} (p, V)}$$

IV. MODULARITY

MILORD II is a currently working KBS environment, which contains the features described up to now, and others that are out of the scope of this paper. The interested reader is referred to [1,2]. We will concentrate on the description of the modular structure of the language.

The language provides three basic mechanisms of module manipulation:

1) Composition of modules through the declaration of submodules,
2) Refinement of modules, and
3) Composition of modules through operators defined by the user via generic modules definition (out of the scope of this paper).

Next, a description of the different components of a MILORD II module is made. Then, the most important operation is outlined: Combination. Combination is used to build up the hierarchical structure of modules. Refinement is used to perform inheritance between modules [9]. Given the space at hand the MILORD II description will necessarily be just a glimpse.

A. Modules

The basic KB units of MILORD II are modules. Modules are compound structures. The basic elements that compose a module are:

- Import interface: information to be provided by the user[1].
- Export interface: output result of a module.
- Object-level knowledge: an initial Object-level Theory.
- Meta-level knowledge: an initial Meta-Level Theory.

Next a detailed description of the two last components is presented.

Object-level Knowledge: The object-level knowledge of a module is composed of:

a) *Dictionary* : this component defines the fact identifiers and some of their attributes, for example their type. It also defines relations between facts, i.e. meta-predicate instances. These relations are defined at the object-level for the sake of compactness. However, they belong to the meta-level knowledge. For example:

```
Dictionary:
   Predicates:
      ...
  Temperature =
   Name: "Patient's Temperature (in centigrade)"
   Question: "Which is the patient's Temperature?"
   Type: Numeric
   Relation: less-rellevant-than AIDS
      ...
  ;; within the Temperature predicate definition the next
  ;; meta-predicate instance is present:
  ;;    less-rellevant-than(Temperature, AIDS)
```

b) *Rules*: this component represents the relational knowledge. For example:

```
R0005 if fever > 38.5 and shivers
  then conclude bacterian-disease is possible[2]
```

c) *Local logic*: it defines three main components of an *Inference System.*: (i) the set of truth-values, (ii) a renaming mapping between the truth-values of the submodules, if any, and the truth-values of the module that combines them and (iii) the connective operators used to combine and propagate the truth-values when making inference. A more complete description can be found in [3].

The next piece of code is an example of module definition containing rule and logic declarations.

[1] Modules can also import information from other modules via their declaration as submodules. See §IV.B.

[2] Given that rules always have intervals of the type *[a, 1]* these intervals are written in MILORD II as just *a*.

```
Module Previous_Treatment =
 Begin
  Import      Prev_Treat
  Export      Penicillin, Tetracycline
  Deductive knowledge
    Dictionary: not defined here
    Rules:
     R001 If Prev_Treat = (Peni)
          Then conclude Penicillin is sure
     R002 If Prev_Treat = (Peni)
          Then conclude Tetracycline is impossible
    Inference system:
     Truth-values = (impossible, sure)
     Connectives[1]:
        Conjunction:
            ( (impossible,  impossible,  impossible)
              (impossible,  possible,    possible)
              (impossible,  possible,    sure))
 end deductive
end
```

Meta-level knowledge: The current implementation of the meta-language allows the definition of meta-rules (as explained in previous sections), and the type of module execution, which can be *lazy* or *eager*. *Lazy* means that facts are evaluated, at the object-level, only when needed, i.e. imported facts and exported facts of submodules are asked only if they may be useful to compute the export interface of the module. On the contrary an *eager* module execution obtains, first of all, values for the imported facts and for the exported facts of the submodules and then the deductive knowledge is used. The interaction between OLT and MLT (reification/reflection step) is made after every import information is obtained, after every submodule execution and after every time OLT is extended by deduction.

B. Combination of modules

We say that a module *combines* other modules when it access to their exported facts. It can be done in two ways: (1) declaring submodules, and (2) applying a generic module.

The declaration of submodules is identical, syntactically, to the declaration of modules, and it is the key to the hierarchical organisation of KBs. The distinction between a KB and a piece of a KB is lost in MILORD II, and a KB will then be a module, possibly containing other modules as submodules.

A generic module can be seen as a module with some submodule names, the name of the arguments, not linked to any particular module. At application time argument names are linked to the particular modules given as parameters.

[1] Connectives may be defined as a truth-table, represented by rows, and following the order of the linguistic terms. In the conjuntion definition of this example module it is represented that: impossible ∧ impossible = impossible, impossible ∧ possible = impossible, etc.

In both cases, i.e. a module which combines its submodules or a generic module applied over some parameters, the resulting module may use, in its rules and meta-rules, the facts appearing in the export interfaces of the modules declared as submodules and/or the parameters. The access to the exported facts is made using a prefix mechanism. An exported fact is identified by two components: 1) a path of module names, separated by "/", indicating how to access to the fact in the hierarchy of modules, and 2) the name of the fact, i.e.

$$name_1 \,/\, name_2 \,/\, \ldots \,/\, name_n \,/\, fact\_name.$$

These combinations of modules build up a hierarchical structure of modules. Each module containing their particular object-level and meta-level theories.

V. CONCLUSIONS

In this paper, MILORD II, a meta-level architecture for complex reasoning tasks, has been presented. Its main characteristics are the multiple-valued local logics management, the reification/reflection processes and the hierarchical modular structure. The object and meta-level languages are formalised and their dynamic interaction described. A glimpse of the whole MILORD II shell is also given.

REFERENCES

[1] Agusti J., Sierra C., Sannella D. (1989): "Adding generic Modules to Flat rule-based languages: A low cost approach", in *Methodologies for Intelligent Systems*, vol 4, Zbigniew Ras ed., Elsevier.

[2] Agustí J., Esteva F., García P., Godo Ll., López de Mántaras R., Murgui Ll., Puyol J., Sierra C. (1991) : *Structured Local Fuzzy Logics in MILORD*, in *Fuzzy Logic for the Management of Uncertainty*, Zadeh and Kacprzyk eds., Wiley, (to appear).

[3] Agustí J., Esteva F., García P., Godo Ll., Sierra C. (1991): "Combining Multiple-valued Logics in Modular Expert Systems", in *Proc. 7th Conference on Uncertainty in AI*, Los Angeles, July.

[4] Barlett S. J., Suber P (eds.) (1987): *Self-reference: Reflections on reflexivity*, Martinus Nijhoff Publishers, Dordrecht.

[5] Harper R., MacQueen D., Millner R. (1986): *The definition of Standard ML*, Report ECS-LFCS-86-2, Dept. of Computer Science, Univ. of Edinburgh.

[6] Maes P., Nardi D (1988): *Meta-level Architectures and Reflection*, North-Holland, Amsterdam.

[7] Puyol J, Godo Ll., Sierra C. (1992): "A Specialization Calculus to Improve Expert Systems Communication", in *Proc. ECAI'92*, Wien.

[8] Sannella D. T., Wallen L. A. (1992): "A Calculus for the Construction of Modular Prolog Programs", *The Journal of Log. Programming*, pp. 147-177.

[9] Sierra C., Godo Ll. (1992): Specifying Simple Scheduling Tasks in a Reflective and Modular Architecture, in *Proc. Workshop on Formal Specification Methods for Complex Reasoning Tasks*, ECAI'92, Wien.

[10] Tan Y. H., Treur J. (1991) : "A Bi-Modular Approach to Non-Monotonic Reasoning", in *Proc. First World Congress on the Fundamentals of AI*, WOCFAI-91, pp. 461, 475, Paris.

[11] Treur J. (1991): "On the Use of Reflection Principles in Modelling Complex Reasoning", *International Journal of Intelligent Systems*, vol 6 pp. 277-294.