Contexts for nonmonotonic RMSes

Jérôme Euzenat

SHERPA project,
Laboratoire ARTEMIS/IMAG,
BP 53X,
F-38041 GRENOBLE
internet: Jerome.Euzenat@sherpa.imag.fr, uucp: euzenat@imag.fr

Abstract

A new kind of RMS, based on a close merge of TMS and ATMS, is proposed. It uses the TMS graph and interpretation and the ATMS multiple context labelling procedure. In order to fill in the problems of the ATMS environments in presence of nonmonotonic inferences, a new kind of environment, able to take into account hypotheses that do not hold, is defined. These environments can inherit formulas that hold as in the ATMS context lattice. The dependency graph can be interpreted with regard to these environments; so every node can be labelled. Furthermore, this leads to consider several possible interpretations of a query.

Reason maintenance systems (RMS) are aimed at managing a knowledge base considering different kinds of reasoning. Such a system is connected to a reasoner (or problem solver) which communicates every inference made. The RMS has in charge the maintenance of the reasoner's current belief base. RMSes developed so far focussed on nonmonotonic reasoning or multiple contexts reasoning. They record each inference in a *justification* that relates *nodes* representing propositional formulas plus a special atom ($\bot$) representing contradiction. A justification ($<\{i_1,...i_n\}\{o_1,...o_m\}>$: c) is made of an IN-list ($\{i_1,...i_n\}$) and an OUT-list ($\{o_1,...o_m\}$). Such a justification is said to be valid if and only if all the nodes in the IN-list are known to hold while those in the OUT-list are not; a node, in turn, is known to hold if and only if it is the consequent (c) of a valid justification. The recursion of the definition is stopped by nodes without justification and by the axioms that are nodes with a justification containing empty IN- and OUT-lists.

Jon Doyle's TMS [Doyle, 1979] proceeds by labelling the nodes of the graph with IN and OUT tags which reflect whether they are known to hold or not. A labelling respecting the constraints stated above is an admissble labelling and a labelling which labels the node $\bot$ OUT is a consistent labelling. The TMS algorithm finds a (weakly) founded labelling, i.e. a consistent admissible labelling which relies on no circular argument. The main work of the TMS occurs when it receives a new justification. It then has to integrate the justification in the graph and, if this changes the validity of the formula, it must propagate this validity: all the nodes that could be IN-ed because of the justified node and all those which could be OUT-ed are examined and updated. If an inconsistency occurs following the addition of a justification, the system backtracks on the justifications in order to invalidate a hypothesis — a formula inferred non monotonically — which supports the inconsistency.

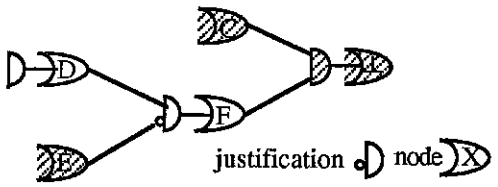

Figure 1. A dependency graph is here represented as a boolean circuit where or-gates are nodes and and-gates are justifications where the nodes in the IN-list come directly while nodes in the OUT-list come through a not-gate. Nodes that have a justification whose IN- and OUT-lists are empty (e.g. D) represent true formulas because they do not need to be inferred. White nodes and justifications are considered valid while hatched ones are invalid. Of course, the value propagation satisfies the rules implied by the circuit components. So, the formulas in the base are ensured to have a valid justification (i.e. corresponding to a valid inference).

Johan De Kleer's assumption-based TMS [De Kleer, 1986; 1988] is rather different. This system considers only monotonic inferences (with only IN-list: $<\{i_1,...i_n\}>$: c), but it deals with several contexts at a time. It considers initial formulas called hypotheses; so, the user can generate and test hypotheses with great efficiency. A set of hypotheses is called an environment and the set of all the environments constitutes a complete lattice structured by the "includes" relation (cf. Figure 2). Instead of labelling absolutely a node (with IN or OUT tags), each node is assigned a label consisting of the set of environments under which it is known to hold. An environment is consistent if $\bot$ is not known to hold in it and the computed labels are minimal in the sense that they do not contain comparable environments. After each inference, the system computes the set of environments that support the inference, inserts it in the label of the inferred node and propagates it through the graph. Then, in order to know if a formula is valid, it compares the current hypothesis set with the label of the node.

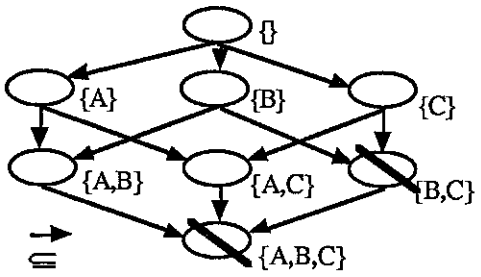

Figure 2. The environment lattice constructed with the hypotheses A, B and C in which the environment {B, C} is known as inconsistent.

As a summary, the TMS handles nonmonotonic inferences and is able to maintain the set of deduced formulas with regard to an axiom set. The ATMS, for its part, cannot accept nonmonotonic inferences, but is able to consider several contexts simultaneously. Merging both systems is needed in order to dispose of a RMS able to deal with nonmonotonicity in multiple contexts. This is the purpose of our context-propagation TMS (CP-TMS).

The first section shows the problem of doing it by extending the ATMS. Section 2 sketches the ideas underlying the CP-TMS. Its construction spreads through sections 3, 4, 5 and 6 by defining the valuations that environments represent, the interpretations that extend valuations to all the formulas, the labels tied to the formulas and their properties of completeness, correctness, minimality and consistency. Section 7 shows that queries can exploit labels in several ways. The last sections are dedicated to the description of a partial implementation (§8), a discussion of some shortcomings of the system (§9) and several solutions to these problems (§10).

1.ATMS and nonmonotonicity

Using an ATMS in order to deal with nonmonotonic reasoning seems attractive. In fact, the introduction of nonmonotonic inferences in the ATMS does not fit well. The main advantage of the ATMS is its use of the context lattice structure in order to infer that if a formula is valid in some environment, it will be under all its supersets. This is the strict definition of monotonicity. So, nonmonotonic inferences lead to important problems in the ATMS (cf. exemple 1).

Example 1.

Deciding under which environment the inference <{A}{B}>:D is valid is not possible. At first sight, {A} is a good candidate because A holds in it while B does not. But, since, in that case, D will be inherited from {A} to {A, B} (cf. Figure 2), in which B holds, {A} is not an adequate environment.

Several authors proposed a special use of the ATMS or similar systems in order to solve these problems [Dressler, 1989; De Kleer, 1988; Giordano and Martelli, 1990]. Here is Oskar Dressler's solution: for each node N, an hypothesis Out(N) can be created whose label represents the set of environments under which N does not hold. Inconsistency between N and Out(N) is dealt with by adding a constraint and completeness is achieved with the help of a special hyperresolution rule.

With this framework, nonmonotonic justifications can be taken into account. The justification (<{i_1,...i_n} {o_1,...o_m}>: c) is communicated to the ATMS by:

$$\forall k;\ 1 \le k \le m\ <\{o_k\}>: O$$

and

$$<\{i_1,\ldots\ i_n, OUT(O)\}>: c.$$

While this approach works well, it suffers from some drawbacks: nonmonotonicity is achieved by multiplying entities. This leads to a "monotonization" of the reasoning and a multiplication of inconsistent contexts (leading, in turn, to intensive hyperresolution). Moreover, additional out-hypotheses are not all significant element for the user.

Since such an approach consists in adding an interface level on the ATMS, it has the advantage to not modify it significantly. But, while it is conceptually simple, it reveals to be a bit cumbersome to work.

2.An alternative

As an alternative solution, a merge between TMS and ATMS is considered which strongly modifies these systems [Euzenat, 1989; 1990]. The main idea consists in using nonmonotonic inferences and the dependency graph of the TMS and labelling the nodes by multi-contexts labels of the ATMS. It means that each node is labelled by the environments such that an absolute (IN/OUT) labelling according to the environment specification will lead the node to hold (be IN).

As pointed out above, representing nonmonotonic inferences in the ATMS context lattice is not possible. Thus, a new kind of environment is defined taking into account the OUTness of a node. *Environments* being built on hypotheses, the new ones will contain the set of holding hypotheses and the set of non holding ones. These new environments are noted by

$$[H_1,\ldots H_n \mid H'_1,\ldots H'_m]$$

or

$$[\{H_1,\ldots H_n\} / \{H'_1,\ldots H'_m\}]$$

where the hypotheses $H_1,\ldots H_n$ hold while the hypotheses $H'_1,\ldots H'_m$ do not. The first set is called *axiom set* while the second is called *restriction*. In order to have a meaningful definition of these environments, the two sets are constrained to be disjoint. At first sight, the idea is not a new one: the two systems ART and MBR [Martins and Shapiro, 1988] have such a kind of environment but they do not share the same semantics.

An environment, for which each possible hypothesis belongs either to the axiom set or the restriction, is called a complete environment. It is assumed to describe the entire world state. Since any environment is generally not complete, it cannot represent *a* state of the world. Its semantics is that it represents the set of complete environments more complete than itself.

So the principle of the system is the same as the ATMS's but the nonmonotonic inferences are taken into account. The absolute labelling of the TMS being manifold, what is meant by a node "holding under an environment",

must be set. The possibilities are threefold (cf. Figure 3):

1) For each environment, the TMS chooses the absolute labelling it considers and the node is labelled according to it.
2) The node must hold under every possible labelling according to the environment.
3) The node must hold under at least one possible labelling according to the environment.

We choose the third solution: a node holds under an environment if it holds for at least one labelling according to each completion of the environment.

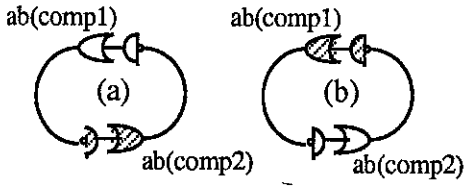

Figure 3. The inferences corresponding to these graphs are: If "Component 1 is abnormal" is not in the base Then hypothesize "Component 2 is abnormal" and If "Component 2 is abnormal" is out of the base Then hypothesize "Component 1 is abnormal". The picture represents the two admissible labelling of the graph.
• If the first option is chosen with (a) as reference labelling, then ab(comp1) holds under the empty environment while ab(comp2) does not.
• If the second option is chosen, then both ab(comp1) and ab(comp2) do not hold (for each one, there exists a labelling for which it does not hold).
• If the last option is chosen, then both ab(comp1) and ab(comp2) hold (for each one, there exists a labelling for which it does hold).

The next sections are dedicated to formally define the new environments and the relations between them. Environment semantics, concerning only hypotheses, are extended toward the whole node base with regard to the dependency graph. Hence, the meaning of labels associated to the nodes can be defined. This work leads to consider several interpretations of the queries against the base.

3.Environments

In the remainder are considered a set $\mathcal{H}$ of hypotheses and a set $\mathcal{N}$ of nodes (with naturally $\mathcal{H} \subset \mathcal{N}$). The set $\mathcal{C}$ of inconsistent nodes will be, without loss of generality, reduced to the set $\{\bot\}$ (thus $\bot \in \mathcal{N} \setminus \mathcal{H}$).

A *valuation* is a boolean function

$$\nu: H \longrightarrow \{0,1\}, \text{ where } H \subseteq \mathcal{H},$$

which states if the hypotheses in the set H hold or not. An environment [A/R] represents the valuation:

$$\begin{array}{rcl} \nu: A \cup R & \longrightarrow & \{0,1\} \\ N & \longmapsto & 1, \text{ if } N \in A \\ & & 0, \text{ if } N \in R \end{array}$$

Of course, A and R must be disjoint in order to avoid valuations that are not functions. Due to the unambiguous mapping between valuations and environments, each one can, subsequently, replace the other.

A valuation is *complete* if it ranges over the whole set of hypotheses ($H=\mathcal{H}$). A *completion relation* is defined upon the set of valuations: a valuation ν_1 is a completion of a valuation ν_2 (or ν_1 is more complete than ν_2) if and only if:

- ν_1 ranges over ν_2 ($H_2 \subseteq H_1$),
- the value given by ν_1 on ν_2's domain (H_2) is the same as the one given by ν_2 ($\forall h \in H_2, \nu_1(h)=\nu_2(h)$).

The set of the *completions* of ν is the set of the environments whose corresponding valuations are more complete than those corresponding to ν. In other words: [A"/R"] is more complete than [A/R] if and only if A is included in (or equal to) A" and R is included in (or equal to) R". The completion set of an environment is defined by:

$$\text{Comp}([A/R]) = \{ [A \cup A'/R \cup R']; R' \cup A' \subseteq \mathcal{H} \ \& \ (R \cap A') \cup (R' \cap A) \cup (A' \cap R') = \{\} \}.$$

This completion relation, among the set of environments definable from a set of hypotheses, is a partial order. But the graph representing environments structured by the completion relation is not a complete lattice anymore.

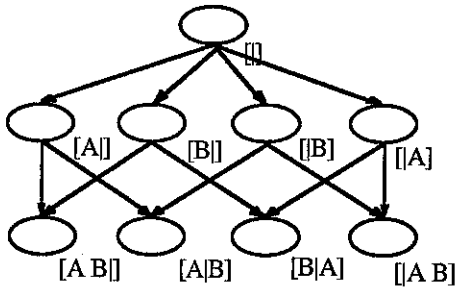

Figure 4. The graph of the completion relation upon the set of hypotheses reduced to {A, B}. There are three completeness levels with regard to the status assigned to the hypotheses. The last level (in which both hypotheses are refered to in the environments) is the set of all the complete environments.

This graph has good properties the ATMS graph lacks. In fact, if F is inferred from B not holding (<{}{B}>:F), then F can have []B] in its label. Moreover, the node F can be inherited from []B] to its completions ([A]B] and []A,B]): they preserve the condition that B does not hold.

A *maximal completion* of a valuation ν is a completion of ν that has the set of all the hypotheses as domain ($H=\mathcal{H}$). So the set of the maximal completions of an environment [A/R] is:

$$\text{Cmax}([A/R]) = \{[A \cup A'/R \cup R'] \in \text{Comp}([A/R]); A \cup A' \cup R \cup R' = \mathcal{H} \}.$$

4.Interpretations

Environments, with the help of valuations, allow to valuate hypotheses. In order to design a reason maintenance system, it is necessary to define the valuation of all the nodes. The interpretation notion, as that of logic, extends the valuations to the whole set of nodes. To that extent, the classical interpretation of the TMS dependency graph is used.

An *interpretation* from a valuation ν is a boolean function

$$I: \mathcal{N} \longrightarrow \{0,1\}$$

which respects the three following properties:

environment conformity: $\forall N \in \mathcal{N}, N \in H \Rightarrow I(N)=\nu(N)$

closedness: $\forall N \in \mathcal{N}$,
$\exists J \in$ LISTOFJUST[N];
$\forall N' \in$ IN-list[J] $I(N')=1$ & $\forall N' \in$ OUT-list[J] $I(N')=0$
$\Rightarrow I(N)=1$

(weak) global groundedness: There exists an ordering (<) of nodes whose interpretation is 1 such that:
$\forall N \in \mathcal{N}, I(N)=1$
$\Rightarrow \exists J \in$ LISTOFJUST[N];
$\forall N' \in$ OUT-list[J] $I(N')=0$
& $\forall N' \in$ IN-list[J] $(I(N')=1$ & $N'<N)$

This definition takes into account the (absolute) labelling of the dependency graph in case of nonmonotonic inferences. The conformity requirement to the environment (if $N \in H$ then $I(N)=\nu(N)$) avoids to consider, as an interpretation from that environment, a labelling that justifies a node of the restriction. It means that if the graph contains the justification <{A}{}>:B, there is no interpretation from [A|B] because the only possible labelling constrains B not to hold. This labelling leads, on the other hand, to an interpretation from [A|].

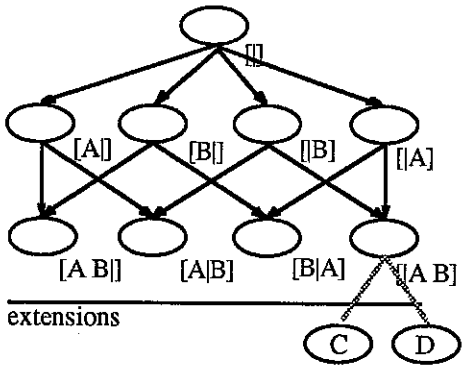

Figure 5. A classical example where the justification set is {<{}{A,B,C}>:D, <{}{A,B,D}>:C}. Under [|A B], there are two possible interpretations: {C} and {D}: the first one where D holds and A, B and C do not and the other where C holds and A, B and D do not.

The recursive definition of interpretations admits several interpretations from a sole valuation. They correspond to the multiple labelling of a TMS graph. The set of interpretations from environment [A/R] is noted IntInd([A/R]) and these interpretations are called *extensions*. They are represented in Figure 5 under the horizontal line.

In order to access extensions through environments, it is possible to bring the line down under the extensions by adding hypotheses C and D. Doing so, extensions will be addressable as maximal completions. But that solution has the drawback of adding hypotheses.

5.Labels

New environments have been defined and interpreted over the whole set of nodes according to the dependency graph. Their main advantage is in representing interpretations in a compact form. At that point, the semantics of labels (which are sets of environments) associated with nodes can be stated.

A *label* for a node N accounts for the set of valuations ν whose completions admit an interpretation I considering N as holding. So the label definition is the following:

LABEL[N] = {C ; $\forall \nu \in$ Comp(C), $\exists I \in$ IntInd(ν); I(N)=1 }.

This is in accordance with the choice made at §2 (i.e. to consider that a node holds under an environment if there exists an interpretation of each completion of it that valuates the node to 1). In other words, the *completeness* property of a label is achieved if

$\forall N \in \mathcal{N}, \forall \nu$,
[$\exists I \in$ IntInd(ν); I(N)=1] $\Rightarrow$ [$\exists C \in$ LABEL[N]; $\nu \in$ Comp(C)].

The *correction*, for its part, is expressed by

$\forall N \in \mathcal{N}, \forall C$,
[$C \in$ LABEL[N]] $\Rightarrow$ [$\forall \nu \in$ Comp(C), $\exists I \in$ IntInd(ν); I(N)=1].

This choice for label definition is significant. It is noteworthy in Figure 5, that both nodes C and D are labelled with [|A B]. An incorrect understanding of it, could lead to the conclusion that an interpretation from [|A B] exists under which C and D simultaneously hold. Which is not true.

6.Consistency

The RMS task is not only to find a (weakly) founded labelling of the graph but also to avoid inconsistent labelling. Our definition of consistent environments defines those which can appear in labels.

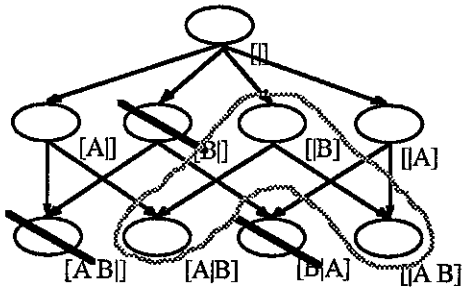

Figure 6. The graph of Figure 5 where the environments in which B holds are inconsistent. So, three of the environments are inconsistent ([B|], [B A|] and [B|A]) and only three of them can be included in labels: [|B], [A|B] and [|B A] (those for which every completion is consistent; they are represented surrounded with a doted line).

An interpretation is called *arguable* (resp. *non arguable*) if and only if $I(\perp)=0$ (resp. $I(\perp)=1$). This definition corresponds really to an inconsistent labelling to be avoided by the RMS. A valuation is said to be *consistent* if none of its interpretations is non arguable (all are arguable). A valuation is called *inconsistent* if it is not consistent. One can adopt another definition, which considers a valuation as consistent if there exists an arguable interpretation from it. Meanwhile, the actual choice for consistency matches the one concerning labels: an inconsistent valuation has a valuation of $\perp$'s label for completion.

For conceptual neatness, it is set that every completion of a valuation in a label must be consistent. Due to its

monotonic reasoning, it is obviously true for the ATMS, but in the present framework, the label definition must be changed by adding a constraint on labels:

$\forall N \in \mathcal{N}, \forall C,$
$[C \in LABEL[N]] \Rightarrow [\forall \nu \in Comp(C), \forall I \in IntInd(\nu); I(\bot)=0].$

In order to retrieve the consistent completions of a label, consistent completions of valuations are defined. A *consistent completion* ν' of a valuation ν is a completion of ν such that every interpretation of ν' is arguable. Thus, the set of consistent completions of ν is:

$$Ccons(\nu) = \{\nu' \in Comp(\nu); \forall I \in IntInd(\nu'), I(\bot)=0\}.$$

A *minimal consistent completion* ν' of a valuation ν is a consistent completion of ν which is a consistent completion of no other consistent completion of ν. Thus, the set of minimal consistent completions of ν is

$$Cmin(\nu) =$$
$$\{\nu' \in Ccons(\nu); \forall \nu'' \in Ccons(\nu), \nu' \in Ccons(\nu'') \Rightarrow \nu'' = \nu'\}.$$

In a label, each environment whose valuation has an inconsistent completion is replaced by its minimal consistent completions. This is a maximalist option; some others, weaker, can be adopted without reconsidering the system consistency [Euzenat, 1989]. While the ATMS approach suppresses only environments all the completions of which are inconsistent, the one considered here is closer to MBR's [Martins and Shapiro, 1988] which restricts environments in the same way (with its restriction list).

7.Queries

The *query* is the way for the reasoner, or an external module, to reach the formulas manipulated (as nodes) by the RMS. So, a query reflects the work of the RMS. It is a formula F associated with an environment [A/B] meaning "does the formula F hold in the context of [A/B]?". Such a query must be answered with yes or no.

In order to interpret it, queries must be defined. The problem consists in knowing the interpretations in which the user intends its query to be satisfied. Obviously, those queries are formulated against the real world but the query environment is incomplete with regard to it. Several attitudes can be adopted to face this incompleteness:

1) The formula must hold in every interpretation from every completion of the query environment.
2) The formula must hold in at least one interpretation from each completion of the query environment.
3) The formula must hold in each interpretation from at least one completion of the query environment.
4) The formula must hold in at least one interpretation from at least one completion of the query environment.

Complications can arise if the query environment is not only incomplete but also inconsistent (or it has at least one inconsistent completion). It is then possible to:

1) Report to the user this inconsistency, requiring him/her to complete its environment in a consistent fashion.
2) Compute the minimal consistent completions of the query environment in order to give a safe answer. So, the whole set of query interpretations can, in turn, be applied against the minimal consistent completions.

Some possible interpretations ranging from the stronger to the weaker are presented hereafter. According to the label definition, "to hold in a completion" means that "there exists an interpretation of that completion in which the node corresponding to the formula holds":

a) Does F hold in every completion of the valuation corresponding to [A/R]?
b) Does F hold in every minimal consistent completion of the valuation corresponding to [A/R]?
c) Does F hold in some completion of each minimal consistent completion of the valuation corresponding to [A/R]?
d) Does F hold in each completion of some minimal consistent completion of the valuation corresponding to [A/R]?
e) Does F hold in some completion of some minimal consistent completion of the valuation corresponding to [A/R]?
f) Does F hold in at least one completion of the valuation corresponding to [A/R]?

If there were no restriction on the label meaning, there should be twice as many possible interpretations.

Several sets corresponding to these query interpretations have been characterized. An important result is that these sets can be characterized with regard to both the query environment and its maximal completions [Euzenat, 1990]. This accounts for the primitive assertion that environments represent the "real worlds" which correspond to their maximal completions.

8.Implementation issues

Our system is implemented in Lisp and called CP-TMS (for context-propagation TMS). This implementation is correct provided that dependency graphs do not contain odd cycles or *alternate even cycles* (i.e. no cycles through OUT-lists).

This implementation is an extension of classical Doyle's and Goodwin's TMS [Goodwin, 1987]. It propagates labels through the graph, strongly connected component (SCC) by strongly connected component, and, inside each SCC, node by node. It acts as if labels were logical formulas in disjunctive normal form:

$$(a_{11} \wedge \ldots a_{1p_1} \wedge \neg b_{11} \wedge \ldots \neg b_{1q_1})$$
$$\vee \ldots (a_{n1} \wedge \ldots a_{np_n} \wedge \neg b_{n1} \wedge \ldots \neg b_{nq_n}).$$

Then the propagation strictly conforms to the following rules:

- A node label is the disjunction of its justification labels.
- A justification label is the conjunction of its antecedents (OUT-list plus nodes in the IN-list).
- An OUT-list label is the conjunction of the negation of the labels of all its nodes.

Minimality is tested during disjunction and consistency during conjunction (each negation is followed by a conjunction). In fact, this algorithm is nearly the same as the one proposed by Drew Mac Dermott [Mac Dermott, 1983], except that nodes are ordered before examined and, since hypotheses are allowed to be justified, that the algorithm is not correct in case of alternate even cycles.

Label structure is implemented with bit-vectors and stored in a normalized form (hypotheses are ordered). The system can answer every specified query type.

Consistency recovery was first implemented in a naive way, inspecting each label. But this is not realistic: the size complexity of the resulting system is exponential with the number of hypotheses supporting constraints. The consistency property used in the ATMS is aimed at reducing the label sizes; at the opposite, in the CP-TMS, consistency increases the label size. Several solutions for it are discussed in [Euzenat, 1989], the more radical one drops out the consistency property of labels. It does not matter because consistency is checked at query processing time. This last solution has also been implemented, saving space and time.

9.Discussion

The presented system suffers from several shortcomings we briefly discuss here.

The implementation only partially fulfills the specifications. If the graph contains alternate even cycles, the CP-TMS will miss some extensions (it will choose one). This is due to the propagation algorithm which is local while generating several extensions is a global property of the graph.

The framework presented here sets the interpretation of environments and does not allow another interpretation to coexist. Transforming both the system and established properties for its specifications from "to hold in at least one interpretation" to "to hold in every interpretation" is straightforward. However, both interpretations cannot be considered simultaneously in the same system while this capability can be useful for many applications.

10.Further developments

Filling in the lacks of the algorithm is the most important work; it will lead to reconsider the specifications. To that extent, research on classical TMS and its multiple extensions has been pursued in order to characterize each possible labelling. The aim is to find out the graph configurations (called generators) that lead to multiple extensions. It has begun by further investigating SCC-based propagation algorithms [Quintero, 1989]. While minimal support graph SCC are not sufficient to characterize those extension generators, we used complete support graph SCC.

Once exhibited multiple extension generators, it is possible to design a perfect RMS: the one in which non determinism can be directed during the propagation and backtracking processes. So, any of the admissible extensions can be reached.

This characterization of multiple extension generators is only a first step. This work has to be reintroduced in the work on CP-TMS. As a matter of fact, propagating, through the graph, the configurations that lead to a node holding or not, reveals if the environment must be in the label of the node and so fully implement CP-TMS specifications. A similar work has already been done for network default theories [Lévy, 1989].

This leads to name extensions and so, to manipulate and query them one by one. Hence, the aim of considering both kinds of environment interpretations (a node should hold under at least one or all of its interpretations) will be achieved. To that extent, it will be necessary to adapt the environment structure, taking generators into account.

11.Conclusion

In order to use, in the same RMS, nonmonotonic inferences and multiple context reasoning, the CP-TMS had been designed. It is based on a new definition of the environments and a clear interpretation of them according to the nonmonotonic dependency graph.

A comparison of this work with several others and a theoretical justification of the presented framework can be found in [Euzenat, 1990]. Moreover, an old but detailed description of the implementation, together with several alternative solutions for consistency that should lead to reduce complexity, is discussed in [Euzenat, 1989].

References

[De Kleer, 1986] Johan De Kleer. An assumption-based TMS. *Artificial intelligence*, 28(2):127-162, 1986.

[De Kleer, 1988] Johan De Kleer. A general labeling algorithm for assumption-based truth maintenance. In *Proceedings seventh national conference on artificial intelligence*, pages 188-192, Saint-Paul, MN USA, 1988, American Association for Artificial Intelligence.

[Doyle, 1979] Jon Doyle. A truth maintenance system. *Artificial intelligence*, 12(3):231-272, 1979.

[Dressler, 1989] Oskar Dressler. An extended basic ATMS. *Lecture notes on computer science (Lecture notes on artificial intelligence)*, 346:144-163, 1989.

[Euzenat, 1989] Jérôme Euzenat. Un système de maintenance de la vérité à propagation de contextes (in French). Research report 779-I, IMAG, Grenoble, FR, February 1989.

[Euzenat, 1990] Jérôme Euzenat. Un système de maintenance de la vérité à propagation de contextes (in French). PhD thesis, Université Joseph Fourier, Grenoble, FR, 1990.

[Giordano and Martelli, 1990] Laura Giordano, Alberto Martelli. An abductive characterization of the TMS. Proceedings of the ninth european conference on artificial intelligence, pages 308-313, Stockholm, SE, 1990.

[Goodwin, 1987] James Goodwin. A theory and system for nonmonotonic reasoning. *Linköping studies in science and technology*, 165, 1987.

[Lévy, 1989] François Lévy. Contribution à la réalisation d'un raisonneur à profondeur variable: le système de maintenance (in French). PhD thesis, Université Paris-Nord, Villetaneuse, FR, 1989.

[Mac Dermott, 1983] Drew Mac Dermott. Contexts and data dependencies: a synthesis. *IEEE Transactions on pattern analysis and machine intelligence*, 5(3):237-246, 1983.

[Martins and Shapiro, 1988] João Martins, Stuart Shapiro. A model for belief revision. *Artificial intelligence*, 35(1):25-79, 1988.

[Quintero, 1989] Jose Alejandro Quintero-Garcia. Parallélisation de la maintenance de la vérité tirant parti des composantes fortement connexes (in French). DEA (master) thesis, INPG, Grenoble, FR, 1989.