Étendre le TMS (vers les contextes)

Jérôme Euzenat

Étendre le TMS (vers les contextes)
Extending the TMS

Jérôme Euzenat

RÉSUMÉ -

Les systèmes de maintenance de la vérité ont été conçus pour raisonner à l'aide de connaissance incomplète. Un système de maintenance de la vérité qui combine les avantages des TMS — autorisant l'utilisation d'inférences non monotones — et des ATMS — considérant le raisonnement sous plusieurs contextes simultanément — est présenté. Il maintient un graphe de dépendances entre les objets utilisés par un système de raisonnement et propage à travers ce graphe les contextes dans lesquels les nœuds sont valides. Une théorie de l'interprétation des contextes est présentée. Elle garantit certaines bonnes propriétés aux contextes manipulés par l'implémentation. Les réponses aux requêtes peuvent alors être interprétées sur la base théorique ainsi posée.

Mots clés :

Raisonnement hypothétique — Raisonnement multi-monde — Raisonnement non monotone — Systèmes de maintenance de la vérité.

ABSTRACT -

Truth maintenance systems are designed to help reasoning with incomplete knowledge. A truth maintenance system which merges the advantages of both TMS — allowing nonmonotonic inferences — and ATMS — viewing the reasoning process under several contexts simultaneously — is introduced. It maintains a dependency graph over the objects manipulated by a problem solver, and propagates the contexts in which the nodes are valid through it. Contexts are interpreted theoretically ensuring good properties of contexts while manipulated by the implementation. Answers provided after a request can then be interpreted regarding this characterization.

Keywords :

Hypothetical reasoning — Multiple-world reasoning — Nonmonotonic reasoning — Truth maintenance systems.

Bull/CEDIAG & projet Sherpa
Laboratoire ARTEMIS/IMAG, BP53X, F-38041 GRENOBLE Cedex
internet: Jerome.Euzenat@sherpa.imag.fr, uucp: euzenat@imag

Actes 7ième congrès AFCET-INRIA Reconnaissance des formes et intelligence artificielle
pp581-596, Paris (FR), 1989

Étendre le TMS (vers les contextes)

Extending the TMS

Jérôme Euzenat

RÉSUMÉ -

Les systèmes de maintenance de la vérité ont été conçus pour raisonner à l'aide de connaissance incomplète. Un système de maintenance de la vérité qui combine les avantages des TMS — autorisant l'utilisation d'inférences non monotones — et des ATMS — considérant le raisonnement sous plusieurs contextes simultanément — est présenté. Il maintient un graphe de dépendances entre les objets utilisés par un système de raisonnement et propage à travers ce graphe les contextes dans lesquels les nœuds sont valides. Une théorie de l'interprétation des contextes est présentée. Elle garantit certaines bonnes propriétés aux contextes manipulés par l'implémentation. Les réponses aux requêtes peuvent alors être interprétées sur la base théorique ainsi posée.

Mots clés :

Raisonnement hypothétique — Raisonnement multi-monde — Raisonnement non monotone — Systèmes de maintenance de la vérité.

ABSTRACT -

Truth maintenance systems are designed to help reasoning with incomplete knowledge. A truth maintenance system which merges the advantages of both TMS — allowing nonmonotonic inferences — and ATMS — viewing the reasoning process under several contexts simultaneously — is introduced. It maintains a dependency graph over the objects manipulated by a problem solver, and propagates the contexts in which the nodes are valid through it. Contexts are interpreted theoretically ensuring good properties of contexts while manipulated by the implementation. Answers provided after a request can then be interpreted regarding this characterization.

Keywords :

Hypothetical reasoning — Multiple-world reasoning — Nonmonotonic reasoning — Truth maintenance systems.

Bull/CEDIAG & projet Sherpa
Laboratoire ARTEMIS/IMAG, BP53X, F-38041 GRENOBLE Cedex
internet: Jerome.Euzenat@sherpa.imag.fr, uucp: euzenat@imag

[2] Jérôme Euzenat
Étendre le TMS (vers les contextes)
Actes 7ième congrès AFCET-INRIA «Reconnaissance des Formes et Intelligence Artificielle», Paris (FR), 29 novembre-1er décembre 1989, pp581-586 (Dunod, Paris (FR), ISBN 2-90-367767-0)

Étendre le TMS
(vers les contextes)

Jérôme Euzenat

1.Introduction

Les raisonnement menés tous les jours souffrent de l'incomplétude de la description du monde réel. C'est cette description connue de l'opérateur qui est fournie au programme. Diverses réponses ont été apportées pour pouvoir raisonner malgré cette incomplétude. Au niveau logique, il existe des logiques de défauts et des logiques non monotones qui prennent en compte le caractère non monotone du raisonnement mené. Au niveau algorithmique, diverses approches sont possibles, qui reviennent soit à compléter la description et à revenir dessus si une incohérence ou une inadéquation est détectée, soit à examiner en parallèle toutes les complétions possibles de la description. Ces deux approches se retrouvent dans deux types de systèmes de maintenance de la vérité: TMS et ATMS.

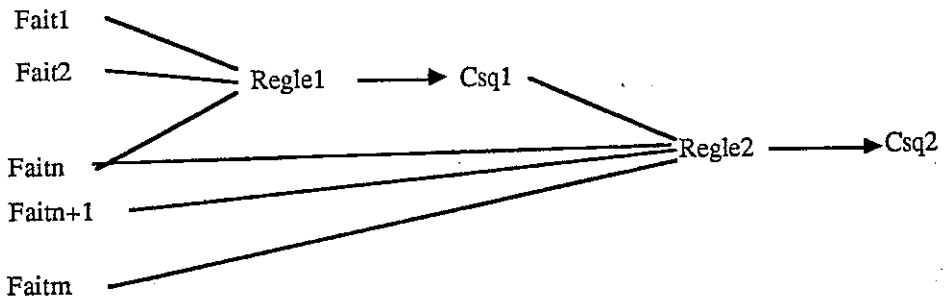

figure 1: Deux inférences successives: la première, en appliquant la règle Règle1 aux formules Fait1,...Faitn, produit la formule Csq1 — appelée conséquent —, la seconde, en appliquant la règle Règle2 aux formules Csq1, Faitn,...Faitm, produit la formule Csq2.

Une inférence est définie comme une opération atomique permettant d'obtenir une nouvelle formule à partir de formules valides (voir figure 1). La propriété de monotonie, pour un système d'inférence, garantit la dérivation des théorèmes dérivés d'un certain nombre d'axiomes à partir d'un sur-ensemble de ces axiomes. Cette propriété s'exprime pour un système logique par «Si $A \vdash \alpha$ alors $A,\Gamma \vdash \alpha$».

Les systèmes logiques classiques possèdent la propriété de monotonie. Divers systèmes, logiques ou non, qui ne possèdent pas cette propriété de monotonie ont été étudiés. Un grand nombre de systèmes implémentés — bases de données et mécanismes de défaut pour les plus répandus — ont un comportement réellement non monotone. Un exemple typique de ces inférences est l'inférence par défaut où l'absence d'une donnée dans la base — ou l'absence de réponse à une requête dans une période de temps finie — sert de fondement à une inférence. Il est alors clair que si la donnée est ajoutée ultérieurement à la base, l'inférence n'est plus valide.

La présence des inférences non monotones dans les bases de connaissance, conjuguée à la possibilité d'enregistrer les résultats de ces inférences, pose problème. En effet, la modification de la validité d'un fait ayant été utilisé dans une inférence peut entraîner l'invalidité de l'inférence. Pour maintenir la cohérence de la base il faut alors supprimer le fait inféré et ses conséquents de la base de connaissance, ce qui ne peut se faire facilement. Pour assurer cette tâche, il est possible d'utiliser des systèmes de maintenance de la vérité.

Ces dernières années, de tels systèmes se sont généralisés et beaucoup de logiciels commerciaux offrent la possibilité de gérer un raisonnement hypothétique soit en explorant plusieurs contextes à la fois, soit en procédant à une gestion de dépendances, soit en utilisant les deux approches. Un des problèmes avec les systèmes rencontrés est qu'ils ne présentent pas une sémantique très claire de leur action. L'ATMS, pourtant, est fondé sur une sémantique immédiate, et les logiques non monotones offrent un cadre attrayant pour les problèmes de non monotonie.

Le système présenté ici, le CP-TMS, se veut un système capable, à la fois de considérer plusieurs complétions de la description du monde en parallèle et d'autoriser l'utilisation d'inférences non monotones. Par ailleurs, le comportement de ce système est explicité par la signification de ce que sont les complétions (ou «environnements»).

Les deux parties qui suivent exposent le fonctionnement et la philosophie des systèmes de base que sont le TMS et l'ATMS. La quatrième partie est une brève revue des systèmes tentant de concilier les deux approches ainsi qu'une critique de l'approche la plus répandue. La cinquième partie pose les bases de ce que constitue un environnement pour un système autorisant à la fois des inférences non monotones et l'exploration parallèle de différentes complétions. Ces bases permettent de justifier l'algorithme rapidement décrit dans la sixième partie, et l'exploitation des résultats détaillée dans la septième partie. La huitième partie comparera le système obtenu aux systèmes préexistants et la conclusion mettra en évidence les problèmes liés à l'approche utilisée.

Le détail de l'algorithme implémenté, certains aménagements de cet algorithme ainsi que les preuves des théorèmes se trouvent dans un rapport de recherche ([Euzenat 89]).

2.Systèmes de maintenance de la vérité à propagation

Le principe du système de maintenance de la vérité à propagation, ou TMS pour "truth maintenance system" [Doyle 79], est d'enregistrer pour chaque inférence la formule inférée au sein d'un nœud et de lui associer une justification représentant l'inférence. Une justification est un couple d'ensembles de nœuds, le premier ensemble étant l'ensemble des nœuds dont la présence dans la base est nécessaire à l'inférence (IN-liste) tandis que le second ensemble est celui des nœuds dont l'absence dans la base est nécessaire à l'inférence (OUT-liste). Les justifications ont donc la structure suivante: <{Fait1,...Faitn} {NFait1,...NFaitm}>. L'ensemble des nœuds considérés par le système, associés à leurs justifications, forme un graphe orienté nommé graphe de dépendances (voir figure 2).

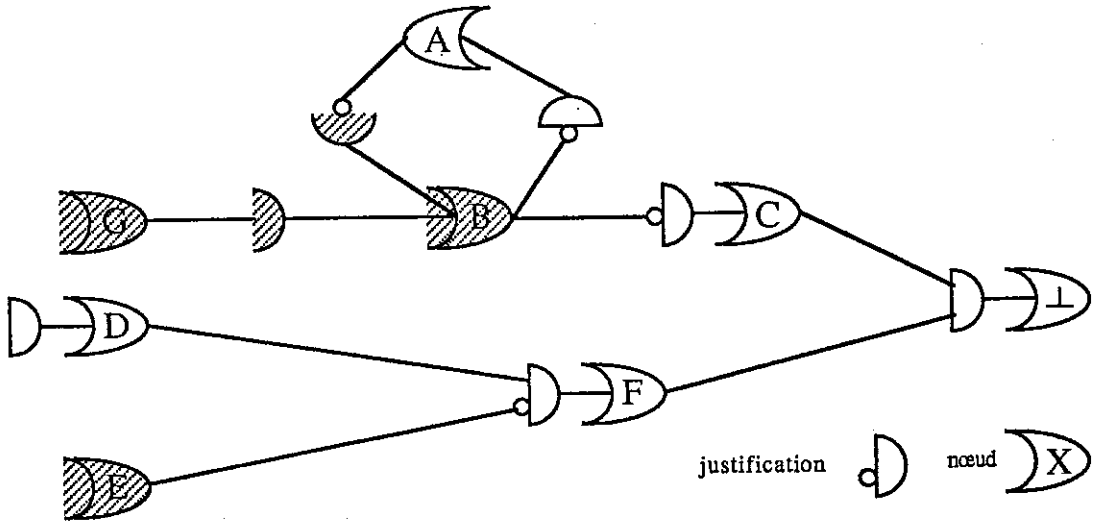

figure 2: Un graphe de dépendances est représenté comme un circuit logique dont les portes «ou» sont des nœuds et les portes «et» des justifications où les nœuds de la IN-liste arrivent directement, alors que les nœuds de la OUT-liste passent au travers d'un inverseur. Les nœuds dont une justification a ses deux listes vides représentent des formules toujours valides car elles n'ont pas besoin d'être inférées, ce sont les prémisses du raisonnement (D). Les nœuds et justifications en blanc sont considérés comme valides alors que ceux en grisé sont invalides. Bien sûr, la propagation de ces valeurs est faite selon les règles dictées par les composants du circuit. Les faits dans la base sont ainsi assurés d'avoir une justification valide — c'est-à-dire correspondant à une inférence valide.

Le système de maintenance de la vérité agit sur cette structure principalement quand une justification correspondant à une inférence lui est fournie. Il se charge alors de propager la validité nouvelle introduite par cette inférence au sein du graphe. Puis, si des faits contradictoires sont présents simultanément dans la base à la suite de cette inférence, le système invoque un mécanisme de rétrogression[1] chargé de supprimer cette contradiction.

La procédure de propagation agit elle aussi en deux passes. La première est chargée de déterminer si chaque noeud a une raison bien fondée d'être dans (IN) ou hors (OUT) de la base, c'est-à-dire qu'il possède soit une justification valide soit toutes ses justifications invalides en considérant que tous les nœuds qui ont le nœud représentant la formule inférée comme antécédent ne sont ni dans la base ni hors de la base. La seconde passe consiste à trouver des justifications faiblement fondées. Cette passe est utile quand il existe des cycles dans le graphe de dépendances. Le système cherche alors à attribuer arbitrairement, mais de manière consistante avec le graphe, l'état du nœud (IN ou OUT). Il faut distinguer entre cycles pairs — représentant une configuration de graphe valide — et cycles impairs — représentant un paradoxe — suivant la parité du nombre d'inverseurs à franchir pour aller d'un nœud à lui-même dans un cycle. Par exemple, dans le graphe de la figure 2, le cycle pair entre A et B autorise deux configurations du graphe.

---

[1] Aussi appelé «retour-arrière».

La procédure de rétrogression est activée quand un nœud contradiction est IN, elle tente alors de supprimer ce nœud de la base. Pour cela elle choisit l'une des hypothèses de ce nœud qui va être invalidée. Une hypothèse est un nœud IN dont la justification supportante possède sa OUT-liste non vide, c'est-à-dire que l'absence dans la base de certaines données entraîne la validité de cette hypothèse. Elle sera donc invalidée en ajoutant dans la base un des nœuds de cette seconde liste.

Les systèmes de maintenance de la vérité à propagation permettent donc, à partir d'une base de données initiale et d'un ensemble d'inférences produites, éventuellement non monotones, de garantir la validité des faits dans la base vis-à-vis de ces inférences et données initiales. Ils permettent en outre, en cas de violation de ce qui représente une contrainte d'intégrité, de rétablir la cohérence de la base en la complétant.

3.Systèmes de maintenance de la vérité à contextes

Le principe des systèmes de maintenance de la vérité à contextes, ou ATMS pour "assumption-based TMS" [De Kleer 86a], est d'enregistrer les contextes dans lesquelles les formules sont valides, au lieu d'enregistrer leur validité instantanée. Ainsi, quand une inférence permet de conclure la formule C à partir des formules Fait1,...Faitn, un ensemble de données — appelé environnement — va être ajouté à l'ensemble des environnements de C — appelé étiquette. Cet environnement est l'intersection des environnements de Fait1,...Faitn. Une hypothèse est enregistrée avec une étiquette contenant l'environnement composé d'elle-même. C'est donc sur celles-ci que seront bâtis les environnements des formules inférées. Les inférences gérées par les systèmes de maintenance de la vérité à contextes ne sont jamais non monotones, seules les formules nécessaires à l'inférence sont considérées.

Le comportement du système est très clairement défini puisque, pour une formule considérée, il construit des étiquettes avec les propriétés suivantes:

- Consistance: aucun des environnements de l'étiquette n'inclut un environnement connu comme inconsistant.
- Complétude: pour tout ensemble E connu d'hypothèses permettant de dériver la formule considérée, il existe un environnement de l'étiquette qui contient E.
- Correction: tous les environnements de l'étiquette permettent d'inférer la formule considérée.
- Minimalité: il n'existe pas d'environnement de l'étiquette qui en contienne un autre.

Ainsi, pour l'inférence représentée graphiquement dans la figure 1, le conséquent Csq2 a pour environnement [Fait1,...Faitn, Faitn+1,...Faitm] sachant que chaque Faiti ($1 \leq i \leq m$) a pour environnement [Faiti].

Les environnements peuvent être considérés comme étant structurés dans un treillis fondé sur la relation d'inclusion (voir figure 3). La condition de minimalité revient alors à enregistrer le plus petit environnement dans lequel l'inférence est valide et le treillis permet de considérer cette inférence comme valide dans tous les environnements supérieurs.

À chaque inférence produite, le système construit la nouvelle étiquette pour la formule inférée. Si cette formule est contradictoire, ce qui correspond à la violation d'une contrainte d'intégrité, alors l'environnement construit est réputé comme contradictoire et tous les environnements contenant celui-ci sont ôtés de toutes les étiquettes des autres nœuds.

Une requête est interprétée non plus dans l'absolu, comme c'était le cas avec le système à propagation et comme c'est le cas dans la plupart des bases de données, mais en relation avec un contexte courant. Le sens de la requête n'est donc plus «est-ce que C?» mais plutôt «est-ce que C si Fait1,...Faitn?». Si le contexte courant est inclus dans l'un des environnements de l'étiquette de C, le système répondra par l'affirmative.

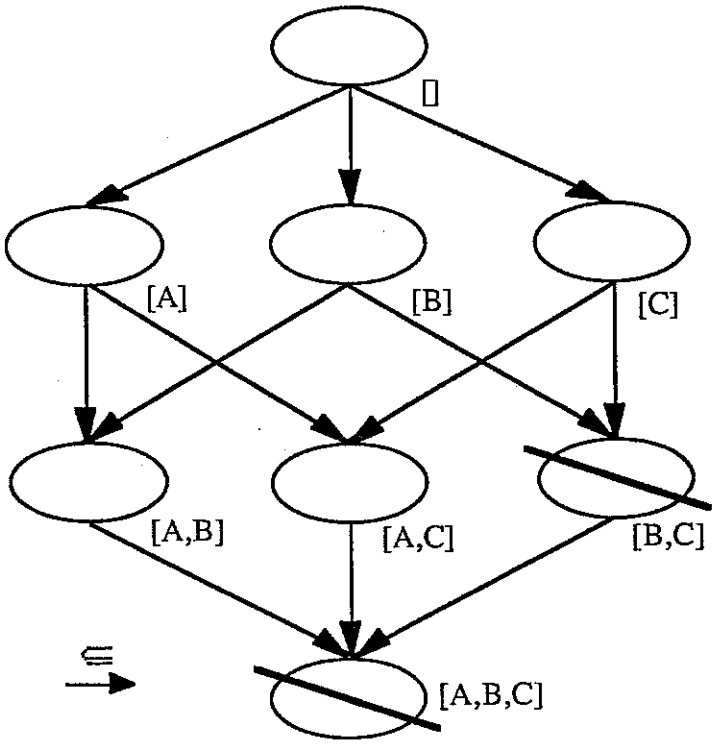

figure 3: Le treillis de contextes structuré par la relation d'inclusion et construit sur les hypothèses A, B et C dans lequel le contexte [B C] est connu comme inconsistant.

Les systèmes de maintenance de la vérité à contextes permettent, à partir d'un ensemble d'inférences produites, nécessairement monotones, d'enregistrer les prémisses nécessaires à la validité de chaque formule. Ils enregistrent en outre les environnements conduisant à une violation des contraintes d'intégrité. Les réponses à une requête se font en fonction d'un contexte courant, fragment de la base, dans lequel se place l'utilisateur.

4.Systèmes étendus

Le système de maintenance de la vérité décrit par Jon Doyle [Doyle 79] ne développe pas la notion de contexte alors que le système défini par Johan De Kleer [De Kleer 86a] ne permet pas l'utilisation d'inférences non monotones. Aussi serait-il très avantageux de pouvoir créer un système disposant des deux notions; diverses tentatives ont déjà été faites dans ce sens.

4.1.Extensions de l'ATMS

C'est ainsi que plusieurs extensions de l'ATMS vers les inférences non monotones ont été proposées. Le but de telles extensions est de pouvoir exprimer des inférences non monotones et de les faire prendre en compte par l'ATMS. Le problème principal est que l'ATMS est fondé sur les propriétés de monotonie de l'inclusion ensembliste. Les propositions qui ont été faites nécessitent donc la génération de nombreux environnements dont certains seront inconsistants.

4.1.1. De Kleer 86

Johan De Kleer [De Kleer 86b] tente d'introduire les inférences non monotones à l'aide d'une primitive CHOOSE représentant la disjonction ($O \veebar N$). Ainsi la justification non monotone $<\{i1,\ldots in\}\{o1,\ldots om\}>$: c[2] est traduite à l'aide de deux nouvelles propositions mutuellement exclusives O et N représentant respectivement la non conformité et la conformité de la OUT-liste à la justification. Elles sont introduites à l'aide des justifications suivantes:

$$\{\} : O \veebar N$$
$$\{O, N\} : \bot^3$$
$$\forall j;\ 1 \le j \le m,\ \{i1,\ldots in, oj\} : O$$

la justification initiale devient alors

$$\{i1,\ldots in, N\} : c.$$

---

2 La formulation j: f, signifie que la formule f est justifiée par la justification j. Dans le cas de l'ATMS, les justifications ne contenant qu'une IN-liste, celle-ci est simplement placée devant la formule f.

3 Dans tout ce qui suit, le symbole $\bot$ (lire inconsistant) permet d'exprimer une contrainte. Comme à tous les autres symboles fournis au système de maintenance de la vérité, il lui est associé un nœud marqué comme inconsistant (voir figure 2).

Ce mécanisme en lui-même est déjà lourd à utiliser, mais la difficulté est encore renforcée par l'utilisation au sein de l'ATMS d'une règle d'«hyper-résolution» destinée à raisonner sur les environnements inconsistants à l'aide du CHOOSE.

4.1.2. Dressler 88

Oskar Dressler propose une autre implémentation [Dressler 88]: pour chaque nœud N, un nœud OUT(N) est créé dont l'étiquette n'est jamais calculée — il s'agit de l'ensemble des contextes dans lesquels N n'est pas valide. Cette méthode introduit donc l'hypothèse du monde clos dans le système de maintenance de la vérité. Il est, par ailleurs, créé une hypothèse Out(N) et les justifications

$$\{Out(N)\} : OUT(N)$$
$$\{Out(N), N\} : \bot$$

sont fournies au système. Ainsi, pour la justification examinée plus haut, le système l'enregistre comme:

$$\forall j;\ 1 \le j \le m\ \{oj\} : O$$
$$\{i1,\ldots in, OUT(O)\} : c.$$

Enfin, la complétude est obtenue en traitant l'inconsistance grâce à la règle

$$\frac{E \cup \{Out(N)\} : \bot}{E : N}$$

qui remplace l'utilisation du CHOOSE et le traitement des requêtes est modifié de façon à répondre positivement quand il existe une extension du contexte d'interrogation dans lequel la formule est valide.

4.1.3. De Kleer 88

De Kleer [De Kleer 88] a récemment proposé un système semblable: à chaque hypothèse H utilisée dans la OUT-liste d'une justification non monotone, est associé un nœud ¬H ne pouvant être justifié. Il est ainsi possible de remplacer CHOOSE({O,N}) par

$$\{\neg O, \neg N\} : \bot$$

et l'algorithme de propagation se chargera de dériver les étiquettes correctes, cependant il n'assurera plus la complétude des étiquettes. Par ailleurs, à chaque découverte d'un environnement inconsistant E contenant des hypothèses H pour lesquelles ¬H a été défini, l'environnement E\{H} sera ajouté à l'étiquette de ¬H évitant ainsi la règle d'hyper-résolution. Ce dernier système n'est pas complètement décrit, en particulier Johan De Kleer ne mentionne pas la signification des étiquettes obtenues.

Ces trois premières extensions de l'ATMS se proposent d'introduire de nouveaux mécanismes en son sein et génèrent un nombre considérable de nœuds et de justifications. En particulier, les extensions multiples ne sont manipulables que parce que les membres de OUT-listes sont forcement des hypothèses. Ils modifient peu, par contre, le fonctionnement de l'ATMS et utilisent au contraire toutes ses propriétés.

4.2.Critique d'ART

Le logiciel ART [Williams 84, Clayton 87] a été développé parallèlement à l'ATMS. Bien que les algorithmes qu'il emploie ne soient pas connus, non plus d'ailleurs qu'une — hypothétique — caractérisation du fonctionnement du programme, il est vraissemblable que son fonctionnement soit différent de celui de l'ATMS malgré une interface semblable. ART est certainement plus proche du système MBR [Martins& 83, 88].

Comme l'ATMS, ART autorise la création d'hypothèses (ou "viewpoints") et évalue ses inférences au sein des environnements les plus généraux dans lesquels elles peuvent prendre place. Mais il autorise aussi la présence d'antécédents négatifs dans les règles d'inférence. Par voie de conséquence, les environnements manipulés sont de la forme C|C1,...Cn correspondant à l'ensemble des "viewpoints" plus spécifiques que C sauf ceux plus spécifiques que C1,...Cn. Partant de là, le système semble se comporter de manière similaire à l'ATMS mais il faut remarquer un certain nombre de différences:

(1) Une justification représentant une inférence non monotone pourrait donner lieu à une étiquette similaire à celle donnée pour les justifications classiques. Cependant, l'algorithme permettant d'obtenir cette étiquette est très complexe, il doit utiliser des moyens permettant de trouver des environnements plus généraux et plus spécifiques à partir d'environnements avec restrictions. La structure des environnements s'éloigne donc de celle développée par De Kleer, le système ne peut exploiter les mêmes idées simples. Au lieu de persister à ne considérer qu'une étiquette par fait, le système ne considère pour un fait qu'un seul environnement avec plusieurs restrictions. Ceci oblige à dupliquer les faits quand un environnement est éclaté. C'est pénalisant car les environnements sont bien plus souvent éclatés que dans le simple ATMS. Ce principe revient, à l'instar de MBR, à ne plus considérer qu'une justification par nœud.

Exemple 1:

Soit la suite de règles suivantes activées l'une après l'autre[4]:

```
(defrule ctxt1 () => (sprout (assert A))) -> C1
(defrule ctxt2 A => (sprout (assert B))) -> C1.1
(defrule ctxt3 A => (sprout (retract A))) -> C1.2
(defrule ctxt4 B => (sprout (retract B))) -> C.1.1.1
```

On a alors les contextes suivants A: C1|C1.2, B: C1.1|C1.1.1. Après l'utilisation de la règle

```
(defrule infC A (not B) => (assert C))
```

le contexte de C pose problème puisqu'il s'agit de C1.1.1 et C1|C1.1,C1.2, le fait C sera donc représenté par deux faits chacun ayant l'un de ces environnements pour étiquette.

(2) Le système offre la possibilité de masquer dans un environnement la présence d'une formule qui doit y être présente au vu du graphe des contextes et de celui des dépendances. Plus ce genre de possibilité est utilisée, plus le sens d'un environnement diffère du sens intuitif qui pouvait être rencontré chez De Kleer. Comme dans les systèmes à base de règles classiques, l'utilisation du `retract` n'a donc pratiquement aucune base logique.

(3) Le système développé par Johan De Kleer est totalement monotone, il tire toutes ses bonnes propriétés de sa monotonie, en particulier si une formule est valide dans un contexte, elle l'est dans tous les contextes plus complets. ART introduit la non monotonie avec l'utilisation des antécédents négatifs. Si la formule F est inférée grâce à la règle $F1 \wedge \neg F2 \Rightarrow F$ et que F2 n'a été trouvé valide nulle part, alors l'étiquette de F sera celle de F1. Si ultérieurement le fait F2 est inféré dans un contexte où F1 est valide, il faudrait reconsidérer l'étiquette de F. Ceci n'est pas fait de façon automatique, mais peut être commandé par l'utilisation des dépendances logiques ("logicals"). Le système de dépendances logiques devrait être, en toute rigueur, un TMS.

(4) L'héritage de l'inconsistance n'obéit pas aux mêmes règles que l'héritage des faits: ART invalide tous les environnements plus spécifiques de ces derniers, or il peut en être qui ne soient pas inconsistants (voir exemple 2).

Exemple 2:

Soit la suite de règles suivantes activées l'une après l'autre[5]:

```
(defrule hypA () => (hypothesize (assert A))) -> C1
(defrule hypB A => (hypothesize (assert B))) -> C1.1
(defrule poisA A (not B) => (poison ""))
```

Dans un tel programme, le contexte de A devrait être déclaré comme inconsistant ("poisoned") mais pas celui de B. ART va supprimer le contexte de A ainsi que tous les contextes plus spécifiques que celui de A — entre autres celui de B.

Le problème soulevé ici est que l'instruction `poison`, utilisée comme les autres instructions en partie droite, n'a pas le même champ d'application que les autres instructions. Cela peut tout à fait être compris et utilisé proprement mais le manque d'homogénéité est flagrant.

5.Étendre les environnements

À partir d'un ATMS ou d'un système similaire, l'idée initiale pour l'étendre vers un système intégrant des inférences non monotones est d'introduire la notion de dépendances entre les faits inférés et leurs antécédents. Mais cette notion est à elle seule insuffisante, il faut alors introduire un véritable TMS. Le problème est que certaines fonctions du TMS et de l'ATMS font double emploi. Il est donc tout naturel de concevoir un système intégré offrant les fonctionnalités d'un TMS et d'un ATMS.

Au lieu d'étendre l'ATMS, en ajoutant une couche au-dessus ou des mécanismes internes, afin qu'il puisse utiliser des justifications non monotones, l'approche présentée ici étend le TMS afin qu'il puisse utiliser les environnements. Cette approche suppose donc de modifier grandement le TMS afin qu'il ne tienne plus compte de l'état dans lequel se trouve un objet mais qu'il se contente de propager l'étiquette de cet objet. Le système présenté sera identifié par le nom de CP-TMS (pour «système de maintenance de la vérité à propagation de contextes»).

L'utilisation de justifications non monotones implique rapidement une modification du type d'environnements manipulés par le système. Ces environnements auront le même aspect que ceux manipulés par le logiciel ART. Un environnement [A1,...An|R1,...Rm] désignera donc l'ensemble des environnements complets où A1,...An sont valides et R1,...Rm sont invalides. De tels environnements pourront aussi être désignés par [{A1,...An}/{R1,...Rm}]. Comme dans l'ATMS, ils peuvent être classés grâce à la relation «est plus complet que» (voir figure 4).

---

[4] Les règles sont définies par leur nom (ici ctxt1...4) ainsi qu'une partie droite et gauche séparées par le symbole =>. L'instruction `sprout` crée un nouveau contexte dans lequel vont s'exécuter les instructions qui lui sont passées en argument. Après chaque définition de règle figure le nom du contexte créé par le `sprout`.

[5] L'instruction `hypothesize` créée un contexte et y exécute les instructions qui lui sont passées en argument. Ces instructions justifierons le contexte. L'instruction `poison` déclare le contexte courant comme inconsistant.

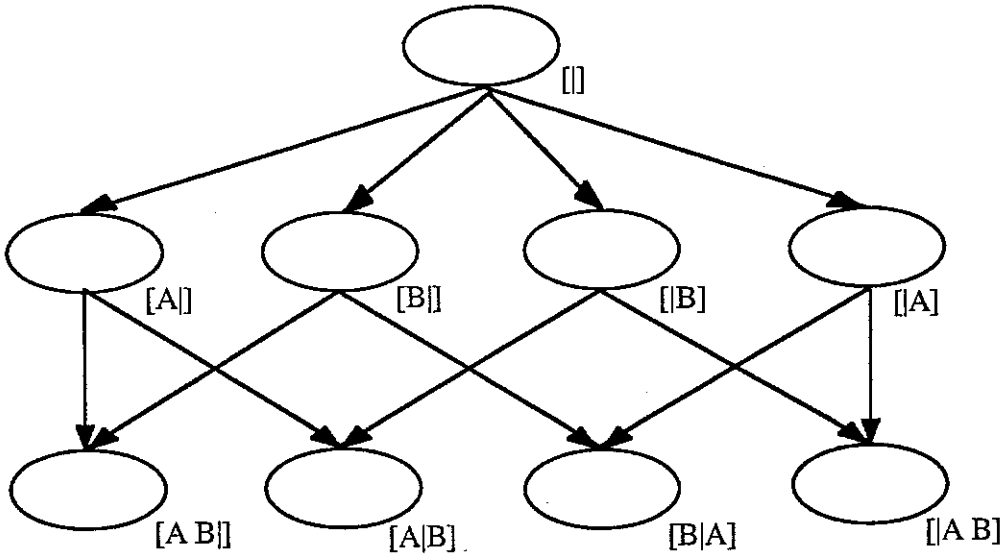

figure 4: Cette figure présente le graphe de la relation «est plus complet que» avec seulement deux hypothèses possibles: A et B. Le niveau 2 (où les deux hypothèses sont présentes dans les environnements) constitue l'ensemble des environnements complets.

Malheureusement, l'ensemble de tous les environnements sur un ensemble d'hypothèses $\mathfrak{H}$ muni de cette relation ne constitue pas un treillis, et qui plus est, la nature non monotone des inférences interdit l'héritage des formules inférées suivant la relation. Il est donc nécessaire pour caractériser le comportement du programme d'aller plus loin dans la spécification et la signification des environnements.

Considérons un ensemble $\mathfrak{H}$ d'hypothèses et un ensemble $\mathfrak{N}$ de nœuds ou formules (avec, bien entendu, $\mathfrak{H}\subseteq\mathfrak{N}$). Un environnement complet (qui décrit donc complètement l'état du monde) est un environnement où la validité de toutes les hypothèses de $\mathfrak{H}$ est connue. En général, un environnement [A/R] n'est pas complet, il ne peut donc représenter complètement l'état du monde. La signification d'un tel environnement est qu'il représente l'ensemble des environnements complets plus complets que lui.

Cette intuition est explicitée dans l'ensemble de définitions et de propositions qui suit, les démonstrations des théorèmes pouvant être trouvées dans [Euzenat 89].

5.1.Valuations

Une *valuation* est un fonction propositionnelle ν: $H\longrightarrow\{0,1\}$, $H\subseteq\mathfrak{H}$ qui fixe la validité de certaines hypothèses et l'invalidité d'autres. Un environnement [A/R] d'une étiquette représente une valuation fournie en extension:

$$\begin{array}{ccl} A\cup R & \longrightarrow & \{0,1\} \\ n & \longmapsto & \text{si } n\in A\ \nu(n)=1 \\ & & \text{si } n\in R\ \nu(n)=0. \end{array}$$

Une valuation est dite une *valuation complète* si $H=\mathfrak{H}$.

Une valuation ν est dite *plus complète* qu'une valuation ν' si et seulement si $H'\subseteq H$. L'ensemble des complétions de ν est l'ensemble des environnements dont la valuation correspondante est plus complète que celle correspondant à ν:

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

La condition supplémentaire ajoutée dans la définition de l'ensemble permet d'éviter d'avoir des environnements ne signifiant rien comme ceux où des éléments de la restriction apparaissent dans l'ensemble d'axiomes.

Théorème 1: $Comp([A/R]) = \{[A\cup A'/R\cup R'];A'\cup R'\subseteq\mathfrak{H}\setminus(A\cup R)\ \&\ A'\cap R'=\{\}\ \}$

L'ensembles des complétions maximales de [A/R] est l'ensemble des complétions complètes de [A/R] : Cmax([A/R]) = {[A∪A'/R∪R']∈Comp([A/R]); A∪A'∪R∪R'=$\mathcal{H}$ }.

Théorème 2:

Cmax([A/R]) = {[A∪A'/R∪R']∈Comp([A/R]); A'∪R'=$\mathcal{H}$\(A∪R) & A'∩R'={} }

La *valuation complétée canoniquement* de ν est la valuation $\nu+$ où $\forall h \in H\ \nu+(h)=\nu(h)$ et $\forall h \in \mathcal{H}\backslash H\ \nu+(h)=0$, c'est-à-dire une complétion sous l'hypothèse du monde clos.

5.2.Interprétation

La notion d'interprétation recouvre celle présente dans les logiques classiques, elle va permettre d'étendre une valuation à l'ensemble de formules manipulables par le système. L'extension se fait par le biais du graphe de dépendances qui permet de propager la validité à partir des éléments de l'environnement.

Une *interprétation* à partir de la valuation ν est une fonction propositionelle I: $\mathcal{N} \longrightarrow \{0,1\}$ construite à l'aide des règles suivantes:

$\forall n \in \mathcal{N}$,
si $\nu(n)=1$ alors I(n)=1
sinon, si $\exists j \in$ LISTEDEJUST[n]; $\forall n' \in$ INLISTE[j] I(n')=1 & $\forall n' \in$ OUTLISTE[j] I(n')=0
alors I(n)=1
sinon I(n)=0.

À une valuation peuvent correspondre plusieurs interprétations comme le montre cette définition récursive qui peut être transformée en équation de point fixe. L'ensemble des interprétations construites à partir de la valuation [A/R] sera noté IntInd([A/R]).

Une interprétation I est dite *soutenable* (respectivement *insoutenable*) si et seulement si I($\perp$)=0 (respectivement I($\perp$)=1).

Une valuation ν est dite *consistante* si aucune des interprétations construites à partir de ν n'est insoutenable. Si une valuation n'est pas consistante elle est dite *inconsistante.*

Une *complétion consistante* ν' d'une valuation ν est une valuation telle que

$\forall h \in \mathcal{H}$; $\nu(h)=1 \Rightarrow \nu'(h)=1$, $\forall h \in \mathcal{H}$; $\nu(h)=0 \Rightarrow \nu'(h)=0$,

et $\forall$I', interprétation à partir de ν', I' soit soutenable. L'ensemble des complétions consistantes de ν est caractérisé par

Ccons(ν)={$\nu' \in$Comp(ν); $\forall$I$\in$IntInd(ν'), I est soutenable}.

Une *complétion consistante minimale* d'une valuation ν est une complétion consistante ν' de ν telle qu'il n'y ait pas d'autres complétions consistantes de ν dont ν' soit une complétion consistante. L'ensemble des complétions consistantes minimales de ν est caractérisé par

Cmin(ν)={$\nu' \in$Ccons(ν); $\neg(\exists \nu'' \in$Ccons(ν); $\nu' \in$Ccons(ν''))}

Théorème 3: Si ν est consistante, alors l'ensemble des complétions consistantes minimales de ν est restreint à {ν}.

Les étiquettes ne doivent contenir que des environnements représentant des valuations consistantes. Cette conception entraîne comme au sein de l'ATMS l'inconsistance pour tous les environnements plus complets qu'un environnement inconsistant précis et bien sûr uniquement pour ceux-là. Ainsi, tous les environnements plus généraux qu'un environnement plus spécifique de l'étiquette de $\perp$ devront être atomisés en plusieurs environnements représentant des valuations consistantes.

Le principe retenu est donc que, dans une étiquette, tout environnement représentant une valuation inconsistante doit être remplacé par les environnements représentant ses complétions consistantes minimales.

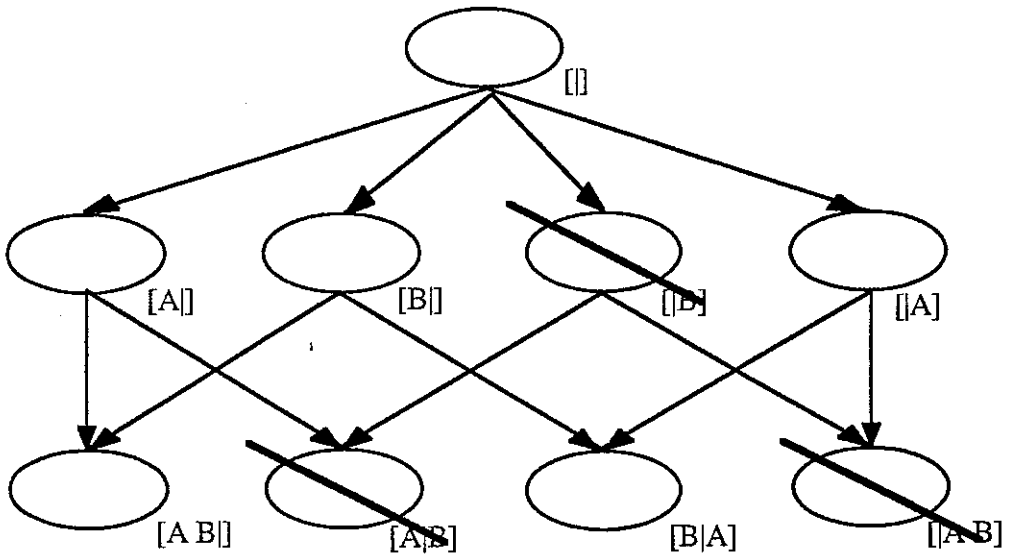

figure 5: Cette figure présente le graphe de la figure 4 où un environnement est inconsistant (celui où B est invalide). Trois des environnements représentés sont donc inconsistants (ceux impliquant l'invalidité de B). Et seuls trois environnements peuvent apparaître au sein des étiquettes: [B|], [B|A] et [B,A|] (ceux dont aucune complétion n'est inconsistante).

À une valuation peuvent correspondre plusieurs complétions consistantes et plusieurs complétions consistantes minimales.

5.3.Les étiquettes

Une étiquette d'un nœud N représente un ensemble de valuations ν qui permettent de construire une interprétation I soutenable et considérant N comme valide.

Si un environnement E est dans l'étiquette du nœud N, cela signifie que N est valide dans une interprétation compatible induite par la valuation correspondant à E. En d'autres termes la complétude des étiquettes des nœuds est réalisée si

$\forall \nu, \forall n \in \mathcal{N}, \exists I \in IntInd(\nu)$

$I(n)=1 \Rightarrow \exists [A/R] \in ETIQUETTE[n]; \forall h \in A\ I(h)=1\ \&\ \forall h \in R\ I(h)=0.$

La correction quant à elle s'exprime par

$\forall \nu, \forall n \in \mathcal{N}, \exists I \in IntInd(\nu)$

$\exists [A/R] \in ETIQUETTE[n]; \forall h \in A\ I(h)=1\ \&\ \forall h \in R\ I(h)=0 \Rightarrow I(n)=1.$

Le travail du CP-TMS sera donc d'assurer à travers tout le graphe de dépendances la complétude, la correction et la consistance des étiquettes des nœuds au regard des définitions données ci-dessus.

6.Principe des algorithmes

À chaque formule mentionnée par le système de raisonnement est associé un nœud dans le graphe de dépendances du système de maintenance de la vérité. Contrairement à ce qui se passait pour le TMS, le nœud ne possède plus un état IN ou OUT mais une étiquette correspondant à une formule fonction des hypothèses. Le travail du CP-TMS est de s'assurer que les étiquettes des nœuds sont conformes aux dépendances.

Les principes des algorithmes utilisés sont les mêmes que ceux de l'algorithme de Jon Doyle, à savoir une phase de propagation activée à chaque fois qu'une nouvelle justification est fournie suivie d'une phase de rétrogression activée quand l'étiquette d'un nœud inconsistant est modifiée.

Le principe de la propagation des étiquettes est simple: pour chaque justification, l'étiquette est la conjonction des étiquettes des nœuds de sa IN-liste avec la conjonction des négations des étiquettes des nœuds de sa OUT-liste. Pour chaque nœud, l'étiquette est calculée en faisant la disjonction des étiquettes de ses justifications. L'algorithme mémorise les étiquettes de chaque nœud de façon à modifier le minimum à chaque justification ajoutée. Les caractéristiques de l'algorithme sont les suivantes:

- La stratégie du support est abandonnée. Cette stratégie permettait de ne recalculer l'étiquette d'un nœud que si un antécédent précis responsable de l'état actuel du nœud voit son état modifié. Les états d'un nœud n'existant plus, il n'est plus question de maintenir une telle stratégie et l'étiquette du nœud doit être constamment remise à jour dès que l'étiquette de l'un de ses antécédents change. La propagation est toujours incrémentale mais suite à l'abandon de la stratégie du support elle concerne cependant tous les nœuds conséquents. Cette modification de l'incrémentalité permet de résoudre le problème (3) de ART.
- La décomposition du graphe de dépendances en composantes fortement connexes utilisée par James Goodwin [Goodwin 87] est conservée. Elle permet de ne remettre à jour les nœuds d'une composante que si tous les nœuds des composantes précédentes ont déjà été examinés.
- La rétrogression ne se fait plus en changeant l'état d'un nœud mais en rétablissant la consistance des étiquettes au sein de tout le graphe.

6.1.Propagation

La principale modification des algorithmes concerne la propagation des étiquettes. L'*étiquette* est une formule, qui, comme toute formule, peut s'exprimer en forme normale disjonctive:

$(a11\wedge\ldots a1p_1\wedge\neg b11\wedge\ldots\neg b1q_1)\vee\ldots(an1\wedge\ldots anp_n\wedge\neg bn1\wedge\ldots\neg bnq_n)$

un *environnement* sera l'un des membres de cette disjonction. Cette forme normale disjonctive peut s'exprimer par une forme compactée — unique — rassemblant tous les disjoints ayant le même ensemble de formules positives:

$(a11\wedge\ldots a1p_1\wedge((\neg b111\wedge\ldots\neg b11q_{11})\vee\ldots(\neg b1m(1)1\wedge\ldots\neg b1m(1)q_{1m(1)}))\vee\ldots$
$(an1\wedge\ldots anp_n\wedge((\neg bn11\wedge\ldots\neg bn1q_{n1})\vee\ldots(\neg bnm(n)1\wedge\ldots\neg bnm(n)q_{nm(n)}))$

chaque conjonction $ai1\wedge\ldots aip_i\wedge((\neg bi11\wedge\ldots\neg bi1q_{i1})\vee\ldots(\neg bim1\wedge\ldots\neg bimq_{im}))$ sera nommée un *environnement global*, chaque conjonction $ai1\wedge\ldots aip_i$ sera appelée un *ensemble d'axiomes*, chaque disjonction $(\neg bi11\wedge\ldots\neg bi1q_{i1})\vee\ldots(\neg bim1\wedge\ldots\neg bimq_{im})$ sera appelée *ensemble de restrictions* et chaque conjonction $(\neg bij1\wedge\ldots\neg bijq_{ij})$ sera nommée *restriction*. Un environnement est donc la conjonction d'un ensemble d'axiomes et d'une restriction.

C'est sous cette forme compactée que seront manipulées et enregistrées les étiquettes dans l'implémentation. Les étiquettes seront représentées par des structures de la forme suivante:

$\{<\{a11\ldots a1p_1\}\{\{b111\ldots b11q_{11}\}\ldots\{b1m(1)1\ldots b1m(1)q_{1m(1)}\}\}>\ldots$
$<\{an1\ldots anp_n\}\{\{bn11\ldots bn1q_{n1}\}\ldots\{bnm(n)1\ldots bnm(n)q_{nm(n)}\}\}>\}$

Les étiquettes sont donc des ensembles de couples représentant les environnements globaux, le premier élément d'un environnement global est sa liste d'axiomes représentée par un ensemble de nœuds, son second élément est son ensemble de restrictions. L'ensemble de restrictions est un ensemble d'ensembles de nœuds représentant les restrictions.

6.1.1. Correction et complétude

Les algorithmes développés pour opérer la propagation sont donc des algorithmes capables d'opérer des négations, disjonctions et conjonctions d'étiquettes. Ils sont simplement dérivés des algorithmes permettant de transformer une formule quelconque en forme normale disjonctive avec certaines optimisations liées au fait que la forme de la formule initiale est connue a priori (l'algorithme n'itère donc pas jusqu'à obtenir la forme voulue, les opérations à effectuer sont connues à l'avance).

L'algorithme utilisé correspondant à des transformations valides de formules logiques, la correction et la complétude des étiquettes des nœuds est alors garantie.

6.1.2. Minimalité

Il n'existe malheureusement pas de formule minimale pour un tel type de formules. Le programme se charge d'implémenter un type de minimalité — nommé minimalité faible — garantissant que deux environnements d'une étiquette ne peuvent avoir de complétions communes. Cette formulation est équivalente à celle de De Kleer si elle est utilisée avec l'ATMS, mais elle n'a pas les mêmes conséquences sur les étiquettes. Pour obtenir la faible minimalité il ne faut pas que dans une étiquette se trouve un environnement tel qu'un autre environnement soit plus général. Autrement dit,

$\min(\text{étiquette}) = \{\text{env}\in\text{étiquette};\ \forall\text{env}'\in\text{étiquette}\ \text{env}\neq\text{env}' \Rightarrow \neg(\text{env}'\supseteq\text{env})\ \}.$

en utilisant la définition suivante de la relation de généralité — «est plus général» correspond alors à «est moins complet» —:

$$[A1/R1]\sqsupseteq[A2/R2] \Leftrightarrow A2\subseteq A1\ \&\ R2\subseteq R1.$$

La minimalité est vérifiée lors de la disjonction des étiquettes sachant que la conjonction de deux étiquettes minimales rend une étiquette minimale.

Cet algorithme rend des étiquettes faiblement minimales dans le sens où il ne rajoute pas d'environnements qui soient comparables à d'autres environnements. Par contre, il ne fait jamais de fusion d'environnements. Par exemple, l'ajout de l'étiquette {[a,b|]} à l'étiquette {[a|b], [|a]}, toutes deux minimales donnera l'étiquette {[a|], [|a]} au lieu de {[|]} qui semble plus minimale.

6.1.3. *Consistance interne*

La consistance interne est ainsi nommée car elle ne dépend pas des contraintes présentes dans le système. Une telle consistance se borne à cerner des étiquettes bien formées sans considérer si ces étiquettes contiennent des environnements inconsistants. Plutôt que les configurations inconsistantes ce sont les configurations impossibles — ou paradoxales — qui sont éliminées. L'absence de restrictions dans l'ATMS fait que la notion d'inconsistance interne n'apparaissait pas.

La première cause d'inconsistance interne dans un graphe est le cycle impair, et en particulier quand une hypothèse fait partie d'un cycle impair. Si c'est le cas, il existe dans l'étiquette de ce nœud un environnement contenant le nœud lui-même dans sa restriction: il faut éliminer cet environnement. Un exemple d'environnement à éliminer est le suivant:

ETIQUETTE[A]={..., [...|...A...], ...}

La seconde cause d'inconsistance interne est tout simplement un environnement inconsistant qu'il faut supprimer: il s'agit d'environnements ayant un nœud commun entre leur ensemble d'axiome et leur restriction. C'est le cas par exemple de l'environnement

[...A...|...A...].

Pour rétablir la consistance interne, toutes les restrictions créant une inconsistance interne sont ôtées de leur environnement. L'algorithme ainsi conçu intervient avant d'établir les étiquettes des nœuds pour la première et après la conjonction d'étiquette pour la seconde.

6.1.4. *Consistance externe*

Les environnements doivent posséder la propriété de consistance externe — c'est-à-dire qu'ils ne permettent pas de dériver $\perp$. Les étiquettes doivent être purgées des environnements inconsistants — c'est-à-dire des environnements comparables à un environnement d'un nœud inconsistant. L'algorithme de propagation garantit l'utilisation d'environnements consistants et que de tels environnements ne peuvent apparaître au cours de la phase de propagation. Le principe est simple: la partie valide d'un environnement E correspond à la formule

$$E \wedge \neg \text{ETIQUETTE}[\perp].$$

Pour s'assurer, à la propagation, que toutes les étiquettes présentes dans le système sont consistantes, les fonctions utilisées lors de la propagation sont modifiées de façon à ce que chaque étiquette calculée soit vérifiée comme consistante en s'assurant qu'aucun environnement d'étiquette ne permette de dériver $\perp$. La méthode utilisée consiste a disloquer tous les environnements plus généraux qu'un environnement de l'étiquette de $\perp$ de façon à n'en conserver que la partie disjointe de l'étiquette de $\perp$.

6.2.Rétrogression

L'inconsistance externe est l'inconsistance ne dépendant pas des nœuds auxquels sont attachées les étiquettes, mais de l'étiquette de $\perp$. Le traitement de l'inconsistance externe se fait de deux façons: à la propagation en interdisant d'ajouter des environnements inconsistants (voir §6.1.4) et en fin de propagation, quand $\perp$ a été dérivé, en supprimant du graphe tous les environnements inconsistants. Il faut alors rétablir la consistance externe en supprimant cette nouvelle étiquette de toutes les étiquettes des nœuds où elle se trouve.

L'approche à mettre en œuvre ici est très lourde puisqu'elle consiste, pour tous les nœuds contenant dans leur étiquette un environnement plus complet qu'un environnement de la nouvelle étiquette de $\perp$, à éclater cet environnement afin de ne conserver que la partie incomparable avec l'environnement de l'étiquette de $\perp$. Il existe cependant des moyens d'améliorer les performances de l'algorithme.

Un certain nombre d'autres méthodes de rétrogression plus proche que celles existant pour le TMS sont proposées dans [Euzenat 89].

6.3.Limites de ces algorithmes

L'algorithme décrit ici n'est cependant pas un modèle d'incrémentalité, il ne vise qu'à être correct. Un algorithme bien plus incrémental ne propageant que des différences d'étiquettes a été développé; malheureusement, à cause des inférences non monotones, ce dernier système est incorrect dans le cas général. Il l'est cependant s'il est utilisé de concert avec un système de raisonnement produisant un raisonnement linéaire ou si le nombre de justifications par nœud est restreint à une unique justification — comme c'est le cas pour ART ou MBR. Un tel système a de meilleures performances et l'avantage de détecter les cycles impairs.

L'algorithme retenu ne détecte pas les cycles impairs, il en supprime seulement certains (voir §6.1.3). Il est possible d'adapter l'algorithme à la détection des cycles impairs. Cela peut se faire en adoptant la solution proposée par James Goodwin [Goodwin 87]. Ainsi, lors de la propagation dans une composante fortement connexe, l'algorithme conserve le nom du nœud duquel il est parti et la parité des dépendances traversées. Il est ainsi capable de se rendre compte quand il revient sur le nœud de départ si le cycle est pair ou impair.

7.Requêtes

Une requête est une question de la forme «la formule f est-elle valide dans le contexte [A/R]?» (ce qui sera noté [A/R]?f). Avant de pouvoir répondre à une telle requête, il faut savoir ce que signifient les étiquettes des nœuds, les contextes et les requêtes.

7.1.Les contextes d'interrogation

Le contexte fourni lors d'une requête a la forme [A/R]. Il n'y pas a priori de raison pour que ce contexte détermine une valuation complète. Le problème est de savoir à quelle interprétation renvoie ce contexte d'interrogation, car un tel contexte détermine plusieurs interprétations. Indépendamment de l'interprétation choisie, il est possible de mettre en évidence différents ensembles correspondant à un contexte.

L'ensemble des formules dérivées de [A/R] est l'ensemble des nœuds qui sont toujours valides sous les hypothèses [A/R], c'est-à-dire les nœuds dont [A/R] est une complétion d'un environnement de leur étiquette:

$$Ctxt(\diamond) = \cup_{C;\diamond\in Comp(C)} \{n\in\mathcal{N};\ C\in ETIQUETTE[n]\ \}$$

Cette définition rejoint celle de Johan De Kleer [De Kleer 86a]. Il faut remarquer que cette définition n'exige pas que les restrictions de $\diamond$ soient exclues de $Ctxt(\diamond)$.

L'ensemble des formules nécessaires dans $\diamond$ est l'ensemble des nœuds qui doivent être valides dans toutes les interprétations construites à partir de $\diamond$:

$$Nec(\diamond) = Ctxt(\diamond)\cup\cap_{C\in Comp(\diamond)\setminus\{\diamond\}} Nec(C)$$

Conséquence: Si $x\in Ctxt(\diamond)$ alors $x\in Nec(\diamond)$

Cette définition qui correspond à une équation de point fixe peut être rendue plus opérationnelle grâce au théorème suivant.

Théorème 4: $Nec(\diamond) = \cap_{C\in Cmax(\diamond)} Ctxt(C)$

Il ressort de ce théorème que l'ensemble des formules nécessaires dans un environnement correspond à l'ensemble des formules valides dans le contexte de tous les environnements qui sont plus complets que lui.

Théorème 5: $\forall n\in\mathcal{N}, (n\in Nec(\diamond) \Rightarrow \forall \diamond'\in Cmax(\diamond), \exists I\in IntInd(\diamond'), I(n)=1)$

Ce théorème permet d'assurer une certaine complétude entre le concept de nécessité et l'interprétation des environnements telle qu'elle a été décrite plus haut. Malheureusement, ce résultat n'est pas satisfaisant car il n'y a pas équivalence: le quantificateur existentiel signale que le système caractérisé choisit l'interprétation dans laquelle il se place.

L'ensemble des nœuds possibles dans $\diamond$ est l'ensemble des nœuds qui sont valides dans une interprétation construite à partir de $\diamond$:

$$Pos(\diamond) = \cup_{C\in Comp(\diamond)} Ctxt(C)$$

Théorème 6: $Pos(\diamond) = \cup_{C\in Cmax(\diamond)} Ctxt(C)$

Encore une fois, ce théorème permet de faire correspondre la notion de possibilité à l'intuition vis-à-vis des contextes complets. Le résultat suivant assure la complétude entre interprétations et environnements:

Théorème 7: $\forall n\in\mathcal{N}, (n\in Pos(\diamond) \Rightarrow \exists \diamond'\in Cmax(\diamond); \exists I\in IntInd(\diamond'), I(n)=1)$

Enfin, ce dernier théorème met en évidence la hiérarchie entre les ensembles définis ci-dessus et permet donc de se faire une idée de la précision et de la portée des résultats qu'ils permettent d'obtenir:

Théorème 8: $Ctxt(\diamond)\subseteq Nec(\diamond)\subseteq Pos(\diamond)$

Il est, par ailleurs, possible que certaines complétions du contexte [A/R] soient inconsistantes. Il est alors commun de se rabattre sur les complétions consistantes minimales d'un tel contexte pour lesquelles il faut refaire la construction précédente.

7.2.Signification possible des requêtes

Le problème est de savoir interpréter une requête, la forme générale d'une requête doit être [A/R]?f. Le problème est d'accorder un sens aux requêtes. Il est clair que ces requêtes sont faites en fonction du monde réel mais le contexte d'interrogation est incomplet par rapport à celui-ci. Plusieurs attitudes peuvent être adoptées vis-à-vis de cette incomplétude:

1- La formule doit être un théorème dans toutes les complétions du contexte d'interrogation.

2- La formule doit être un théorème dans la complétion canonique du contexte d'interrogation (hypothèse du monde clos appliquée au contexte d'interrogation).
3- La formule doit être un théorème dans une complétion quelconque du contexte d'interrogation.

Des complications peuvent être introduites si non content d'être incomplet, le contexte d'interrogation est inconsistant (au sens de «une complétion du contexte d'interrogation est inconsistante»). Il est alors possible de:

1- Simplement signaler à l'utilisateur cette inconsistance en lui demandant de compléter son contexte de façon non contradictoire.
2- Calculer les complétions consistantes minimales de son contexte d'interrogation pour lui donner une réponse. Deux nouvelles possibilités s'offrent alors:
   1- La formule doit être un théorème dans toutes les complétions consistantes minimales du contexte d'interrogation.
   2- La formule doit être un théorème dans une complétion consistante minimale du contexte d'interrogation.

   Et ici encore, les trois premiers choix sont possibles pour déterminer ce que signifie être un théorème dans une complétion consistante minimale qui est un contexte d'interrogation comme un autre (mais forcément consistant).

Certaines significations sont exposées ci-dessous dans un ordre qui va de la plus difficile à la plus facile à satisfaire:

a) F est-elle un théorème dans toutes les complétions de la valuation correspondant à [A/R]?
b) F est-elle un théorème dans toutes les complétions consistantes minimales (au sens de toutes les complétions de toutes les complétions consistantes minimales) de la valuation correspondant à [A/R]?
c) F est-elle un théorème dans certaines complétions de toutes les complétions consistantes minimales de la valuation correspondant à [A/R]?
d) F est-elle un théorème démontrable uniquement à l'aide d'inférences non monotones sous l'hypothèse du monde clos et dans un contexte contenant [A/R] — ce qui est noté [A/R]$\vdash$—f?
e) F est-elle un théorème dans la valuation complétée canoniquement à partir de la valuation correspondant à [A/R]? Il est à noter que cette valuation peut elle-même être inconsistante.
f) F est-elle un théorème dans une complétion consistante minimale (au sens de toutes les complétions d'une complétion consistante minimale) de la valuation correspondant à [A/R]?
g) F est-elle un théorème dans certaines complétions d'une complétion consistante minimale de la valuation correspondant à [A/R]?
h) F est-elle un théorème dans une complétion de la valuation correspondant à [A/R]?

L'interprétation (d) est difficile à justifier dans la mesure où les étiquettes ne contiennent pas d'informations quant à la nature du raisonnement tenu.

Les interprétations (a) et (h) sont les plus extrêmes, (a) exige que toutes les complétions de [A/R] permettent de dériver F. (h) demande simplement s'il est possible, dans un contexte où la connaissance est plus complète que F soit un théorème.

(b) et (f) correspondent à un système considérant les complétions consistantes minimales de [A/R], c'est-à-dire que si [A/R] est inconsistant ([A/R] étant incomplet il doit donc être complété pour redevenir consistant et fidèle à la réalité), il est complété par le système afin d'évaluer la requête dans une complétion consistante minimale correspondant plus à la réalité. Il faut noter que (a) et (b) d'une part et (h) et (g) d'autre part sont équivalents si une complétion inconsistante est considérée comme permettant de dériver n'importe quelle formule.

(c) et (g) considèrent les complétions consistantes minimales comme (b) et (f), mais exigent qu'il n'y ait qu'une complétion des complétions consistantes minimales dans lesquelles F soit un théorème comme (h).

Les ensembles Nec($\wp$) et Pos($\wp$) correspondent respectivement aux ensembles de nœuds pour lesquels la réponse aux requêtes (a) et (h) est positive. Ces résultats sont obtenus grâce aux théorèmes 5 et 7 ainsi qu'à la complétude et correction des étiquettes des nœuds. La caractérisation de Nec($\wp$) et Pos($\wp$) ne se préoccupe pas de la consistance des valuations, il est possible d'obtenir l'équivalent de Nec($\wp$) et Pos($\wp$) pour les requêtes (b), (c), (f) et (g) en introduisant les notions de complétions consistantes minimales.

Il faut noter que ce type d'interprétation des contextes d'interrogation pourrait être étendu à l'interprétation des étiquettes si celle-ci n'était fixée d'avance. Il y aurait alors 2^6=36 interprétations possibles d'une requête. Dans le système actuellement implémenté, toutes les approches cohabitent, c'est-à-dire qu'il est possible d'appeler la fonction correspondant à l'interprétation que l'on désire donner à sa requête.

8.Comparaisons avec les précédentes approches

8.1.Le TMS de Jon Doyle

Si toutes les étiquettes sont {} (OUT) ou {[]} (IN) le travail de propagation est le même que celui fait par le TMS. Cependant, l'efficacité est moindre car il est nécessaire de tenir compte des opérations sur les contextes. En cas de cycles impairs, le système présenté bouclera contrairement à l'implémentation de James Goodwin. La stratégie du support étant abandonnée, la procédure de propagation restera cependant assez efficace puisqu'elle s'arrêtera dès qu'une étiquette n'est pas modifiée. Elle n'aura pourtant pas toute l'efficacité du système de Doyle car il faudra examiner toutes les composantes fortement connexes.

La principale différence est dans la rétrogression. Il ne semble plus possible de faire de la rétrogression de la même manière que le faisait le système de Jon Doyle. En particulier, le système présenté ici est incapable de rétablir la consistance après qu'une inconsistance ait été découverte. Il vaut mieux, pour faire de la rétrogression, utiliser la notion d'hypothèse présente dans le système, il est alors possible d'implémenter des mécanismes spécifiques de rétablissement de la consistance qui font un travail similaire à celui de la rétrogression [Euzenat 89].

L'avantage du CP-TMS est bien sûr de pouvoir raisonner simultanément dans plusieurs contextes.

8.2.L'ATMS de Johan De Kleer

Le système présenté perd aussi en généralité par rapport au système de maintenance de la vérité à contextes de De Kleer [De Kleer 86a]. En effet, il n'y a plus une forme unique — correspondant à la forme normale disjonctive — pour exprimer un contexte mais plusieurs formes, cela ne facilite pas les algorithmes de recherche. Le treillis perd alors beaucoup de son intérêt puisqu'il ne suffit plus de comparer les environnements de l'étiquette d'un nœud au contexte courant pour savoir s'il est valide mais qu'il faut ensuite comparer les exceptions.

Il est possible d'utiliser le CP-TMS à la place de l'ATMS. Tous les environnements seront alors sans restrictions. Le CP-TMS sera moins efficace car il devra tenir compte des opérations sur les restrictions et ne profitera pas des structures de vecteurs ni surtout de la minimalité.

Mais, de manière surprenante, cette extension du TMS est en fait une extension de l'ATMS, toutes les procédures évoquées se retrouvent en effet dans un récent article de De Kleer [De Kleer 88] plus complet quant à l'exposition de la propagation que la définition de l'ATMS [De Kleer 86a]:

- PROPAGATE correspond à l'algorithme général de propagation d'étiquette.
- NOGOOD correspond au rétablissement de la consistance externe dans tout le réseau.
- WEAVE correspond au calcul de l'étiquette affectée à une justification avec vérification de la minimalité et de la consistance externe.
- Enfin UPDATE correspond au calcul et à la mise à jour de l'étiquette d'un nœud avec vérification de la minimalité.

La différence avec l'algorithme de l'ATMS est que la partie propagation de contextes est bien plus complexe à cause de la présence des justifications non monotones et donc bien plus exposée que dans les premiers papiers de Johan De Kleer où elle était implicite. Les problèmes posés par ces types d'environnements ont entraîné:

- Une nouvelle formulation de la minimalité.
- Une notion de consistance interne qui était absente dans les environnements simples et une redéfinition de la consistance externe.
- Une multiplication des interprétations données à une requête.

L'avantage majeur du CP-TMS est qu'il permet d'exprimer simplement des inférences avec antécédents négatifs.

Cette extension du TMS vers les contextes est aussi une extension de l'ATMS vers les justifications non monotones. En fait le système ressemble beaucoup plus au second qu'au premier (et superficiellement à ART). Le CP-TMS tente de concilier les avantages des deux systèmes et permet donc l'expression de manière simple et naturelle d'inférences de type défaut et la manipulation de contextes.

9.Conclusion

L'extension du système de maintenance de la vérité à propagation qui a été présenté permet d'unifier les concepts présents à la fois dans le TMS et dans l'ATMS. Il permet donc de raisonner dans divers contextes simultanément tout en manipulant des inférences non monotones. Le CP-TMS est caractérisé théoriquement par l'explicitation de la signification des environnements manipulés et par l'utilisation d'opérations pertinentes par rapport à cette signification. Les résultats obtenus permettent d'expliquer les réponses aux requêtes fournies par le système.

Cependant les inférences non monotones ne sont pas mieux traitées que par le TMS. En effet, la présence de cycles pairs dans le graphe de dépendances peut occasionner diverses interprétations construites à partir d'une valuation — ceci seulement si aucun nœud du cycle n'est déclaré comme hypothèse. Dans un tel cas, les réponses aux requêtes fondées sur l'existence d'une interprétation peuvent sembler quelque peu insuffisantes. Ce problème conduit à examiner diverses réponses [Euzenat 89].

Quoiqu'il en soit, le CP-TMS est un premier pas vers l'implémentation de systèmes de maintenance de la vérité liant TMS et ATMS dont l'action est clairement définie. Il a été implémenté en Lisp et répond aux caractéristiques décrites ici. Il permet de déclarer hypothèses et inférences et répond à différents types de requêtes.

Remerciements

Je tiens à remercier Raymond Sclison et Philippe Vignard des remarques dont ils m'ont fait part lors d'une première présentation de ce travail ainsi que Patrice Uvietta, François Rechenmann et Laurent Buisson pour leurs commentaires sur ce papier. Bien entendu, toute erreur subsistante est entièrement de mon fait.

Références

[Clayton 86] Bruce Clayton, ART programming tutorial 2: a first look at viewpoints, Inference corporation, Los Angeles (CA, US), 1986

[De Kleer 86a] Johan De Kleer, An assumption-based TMS, *Artificial intelligence* 28-II, pp127-162, 1986

[De Kleer 86b] Johan De Kleer, Extending the ATMS, *Artificial intelligence* 28-II, pp163-196, 1986

[De Kleer 88]Johan De Kleer, A general labeling algorithm for assumption-based truth maintenance, Actes 7ième AAAI, pp188-192, Saint-Paul (MN, US), 1988

[Doyle 79] Jon Doyle, A truth maintenance system, *Artificial intelligence* 12-III, pp231-272, 1979

[Dressler 88] Oskar Dressler, Extending the basic ATMS, Actes 8ième ECAI, pp535-540, München (DE), 1988

[Euzenat 89] Jérôme Euzenat, Le système de maintenance de la vérité à propagation de contextes, Rapport de recherche 779-I, Laboratoire ARTEMIS/IMAG, Grenoble (FR), 1989

[Goodwin 87] James Goodwin, A theory and system for non monotonic reasoning, *Linköping studies in science and technology* 165, 1987

[Martins& 83] João Martins, Stuart Shapiro, Reasoning in multiple belief spaces, Actes 8ième IJCAI, pp370-373, Karlsruhe (DE), 1983

[Martins& 88] João Martins, Stuart Shapiro, A model for belief revision, *Artificial intelligence* 35-I, pp25-79, 1988

[Williams 84] Chuck Williams, ART: The advanced reasoning tool, conceptual overview, Inference corporation, Los Angeles (CA US), 1984

CR category: I.2.3 Deduction and theorem proving [ARTIFICIAL INTELLIGENCE]: *nonmonotonic reasoning and belief revision.*
General term: algorithms, theory.