THÈSE

présentée par Jérôme Euzenat

pour obtenir le titre de docteur
de l'université Joseph Fourier - Grenoble 1

(arrêté ministériel du 5 juillet 1984)

spécialité informatique

◊◊◊

Un système de maintenance de la vérité à propagation de contextes

◊◊◊

Date de soutenance: 16 février 1990

Composition du jury:

M. Laurent Trilling	**(président)**
Mme Marie-Odile Cordier	**(rapporteur)**
M. Daniel Kayser	**(rapporteur)**
M. François Rechenmann	
M. Jean-Marc Pugin	

Thèse préparée au sein du laboratoire ARTEMIS/IMAG

A context-propagation based truth maintenance system

Abstract

Hypothetical reasoning consists in completing available knowledge to achieve reasoning. In order to help hypothetical reasoning, specific algorithms are designed which efficiently deal with hypothesis and their consequences, and moreover allow an automatic hypothesis generation. This last consideration leads to implement a nonmonotonic reasoning.

Truth maintenance systems (TMS) record the inferences, made by a reasoning system, as a dependency graph and insure the coherence of the formulas to be found in the knowledge base. Two kinds of truth maintenance systems were considered:

- Propagation-based TMS deal with nonmonotonic inferences and propagate validity through the graph. Labels associated with formulas represent an interpretation of the graph.
- Context-based TMS are restricted to monotonic inferences but use labels representing contexts in which the formulas must be present. They allow to reason under several contexts simultaneously.

This work designs a new system that merges the advantages of both systems. It allows to reason simultaneously under several contexts with the help of nonmonotonic inferences. For this purpose, new environments able to take into account the absence of an hypothesis are defined. They are associated with an interpretation which is extended all over the dependency graph according to the propagation-based TMS' interpretation.

A system corresponding to this characterization, the CP-TMS was implemented as an extention of propagation-based TMS. That implementation is described here before to be discussed.

Keywords

- Automated reasoning
- Hypothetical reasoning
- Nonmonotonic reasoning
- Truth maintenance

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

Un système de maintenance de la vérité à propagation de contextes

Résumé

Le raisonnement hypothétique consiste à compléter la connaissance disponible afin de poursuivre un raisonnement. L'aide aux utilisateurs de systèmes de raisonnement hypothétique nécessite la conception d'algorithmes spécifiques, pour pouvoir gérer efficacement les hypothèses et leurs conséquences et pour permettre de poser automatiquement des hypothèses. Cette dernière exigence conduit à implémenter un raisonnement non monotone.

Les systèmes de maintenance de la vérité enregistrent les inférences produites par un système de raisonnement sous forme d'un graphe de dépendances et se chargent de garantir la cohérence des formules présentes dans une base de connaissance. Deux types de systèmes de maintenance de la vérité ont été proposés:

- Les systèmes à propagation acceptent des inférences non monotones et propagent la validité absolue au sein du graphe de dépendances. L'étiquetage obtenu représente une interprétation du graphe.
- Les systèmes à contextes n'acceptent que des inférences monotones mais propagent des étiquettes dénotant les contextes dans lesquels les formules doivent être présentes. Ils permettent donc de raisonner sous plusieurs contextes simultanément.

Le but de ce travail est de concevoir un système qui combine leurs avantages. Il permet de raisonner simultanément sous plusieurs contextes à l'aide d'inférences non monotones. Pour cela, des environnements capables de tenir compte de l'absence d'hypothèses sont définis. Une interprétation est associée à ces environnements et est étendue aux nœuds du graphe de dépendances, en accord avec l'interprétation des systèmes à propagation. Cela permet d'établir la signification des étiquettes associées aux nœuds du graphe, et de proposer de multiples possibilités de soumettre des requêtes au système.

Un système correspondant à cette caractérisation, le CP-TMS, est implémenté comme une extension des systèmes de maintenance de la vérité à propagation. Cette implémentation est décrite ici, puis critiquée.

Mots-clé

- Mécanisation du raisonnement
- Raisonnement hypothétique
- Raisonnement non monotone
- Maintenance de la vérité

Remerciements

D’un naturel plutôt avare en remerciements, je profite de cette rubrique, dédiée à cet effet pour réparer quelques années d’omission. Que soient donc remerciés:

- Laurent Trilling, professeur à l’université Joseph Fourier, pour me faire l’honneur de présider le jury de cette thèse.
- Marie-Odile Cordier, professeur à l’université de Rennes 1, et Daniel Kayser, professeur à l’université de Paris nord, pour me faire l’honneur de juger ce travail mais aussi pour le temps qu’ils ont bien voulu me consacrer et leurs conseils pertinents sur la rédaction de ce travail.
- Jean-Marc Pugin, ingénieur au CEDIAG, pour les discussions intéressantes que nous avons eues et pour m’avoir fait l’honneur de participer à ce jury.
- François Rechenmann, directeur de recherche à l’INRIA, pour sa confiance, ses conseils et son soutien tout au long de ce travail, mais aussi pour l’esprit de curiosité et d’ouverture qu’il fait régner au sein de son équipe.

- Patrice Uvietta, pour ses relectures attentives des (nombreux) documents que j’ai pu rédiger, mais aussi pour son humour et sa culture insondable.
- Laurent Buisson, pour ses relectures accrocheuses, et pour tous les *échanges* que nous avons pu avoir depuis trois ans que nous partageons le même bureau.

Sans eux, ce travail ne serait certainement pas écrit avec les mêmes mots.

Ce travail a été réalisé conjointement au laboratoire ARTEMIS à Grenoble et au sein de la société Cognitech dans un premier temps puis au Centre d'Expertise et de Développement en Intelligence Artificielle du Groupe Bull dans un second. Somme toute, la poursuite d'un travail de recherche au sein de différents milieux, si elle cause certains problèmes, n'est pas une expérience désagréable. Je dois donc aussi remercier:

- Toute l'équipe du projet Sherpa, Valérie Favier, Pierre Fontanille, Olga Mariño, Gia-Toan Nguyen, Ana Simonet, avec une mention spéciale pour les empoignades avec Dominique Rieu. Les collègues du laboratoire ARTEMIS: Jose-Luis «Gringo» Aguirre, Yves «N'Balawe N'Ga N'Dud et même N'Papa» Durand, Alejandro Quintero, Bertrand «Mac» Rousseau et Danielle «Bouchon» Ziebelin.
- Michel Clerget qui m'accueillit dans son département à Cognitech ainsi que Philippe Vignard, Raymond Sclison et Philippe Terret du même département pour la collaboration fructueuse que nous avons eue.
- Rémi Lescœur et toute l'équipe développement du CEDIAG/Bull qui m'ont accueilli après mon départ de Cognitech. Jean-Paul Billon, Olivier Coudert et Jean-Christophe Madre du centre de recherche du groupe Bull et Libero Maesano du CEDIAG pour les discussions instructives sur les systèmes de maintenance de la vérité et toutes sortes d'autres choses.
- Les secrétaires Claudine Meyrieux du laboratoire ARTEMIS et Andrée Régnier du CEDIAG pour leur efficacité et leur gentillesse.
- La société GSI-ERLI au travers de son directeur technique, ainsi que la société de reprographie Jack Suire au travers de ses sympathiques patrons, pour m'avoir fait profiter de leur matériel.
- Mes parents pour m'avoir soutenu et supporté mes incessants voyages entre Paris et Grenoble.

Tous ont contribué d'une manière ou d'une autre au développement de ce travail.

Si vous pensez avoir été oublié, ou pour toute remarque, envoyez moi un mot à

Jérôme Euzenat (service consommateur)
Laboratoire ARTEMIS/IMAG, BP 53X, 38041 GRENOBLE Cedex

je vous ferais une dédicace personnelle.

Jérôme Euzenat

Sommaire

Les chapitres sont numérotés en romain et en continuité dans toute la thèse. Les conventions de numérotation sont de numéroter figures, exemples, définitions, lemmes et théorèmes à partir de 1 et en continuité dans toute la thèse. Les paragraphes à l'intérieur d'un chapitre sont numérotés et référencés par des séquences de chiffres arabes (1.1.1); les renvois à des paragraphes de chapitres extérieurs sont réalisés en faisant précéder la référence du numéro du chapitre (en romain).

III. Fonctionnement du CP-TMS

IV. Critiques et perspectives

Conclusion

Références

Preuves

Index

Introduction

Le raisonnement hypothétique est utilisé en présence d'information incomplète. Des hypothèses sont posées pour compléter la connaissance afin de pouvoir mener à bien un raisonnement qui sans cela serait bloqué. Par exemple, connaissant un individu de 35 ans nommé Claude et ne connaissant pas son sexe, il est impossible d'en déduire que cet individu a fait son service militaire. Poser l'hypothèse que cette personne est de sexe masculin aidera à poursuivre plus avant le raisonnement. Une telle facilité doit pouvoir permettre de considérer un problème suivant différentes hypothèses et donc de modifier l'ensemble d'hypothèses, soit en en ajoutant soit en en supprimant. Ces hypothèses pourront être abandonnées soit parce qu'une connaissance plus complète du monde montre qu'elles sont fausses, soit parce qu'elles entrent en contradiction avec l'information disponible.

Un système permettant le raisonnement hypothétique doit tout d'abord offrir une première facilité, celle de poser des hypothèses. On distingue deux acteurs susceptibles de poser des hypothèses, l'utilisateur d'un système ou ce système lui-même. Si le rôle est dévolu à l'utilisateur, la marge de manœuvre du système est relativement restreinte, il peut noter le statut d'hypothèses de certains axiomes mais ne connaît rien des raisons pour lesquelles une hypothèse a été posée ou abandonnée; il doit faire appel à l'utilisateur en cas de problèmes au cours du raisonnement. En revanche, si le rôle de poser les hypothèses est dévolu au système, l'utilisateur perd le contrôle des processus hypothétiques. Il peut cependant guider le système, soit sur sollicitation de celui-ci, soit à l'aide de connaissance communiquée a priori au système. Utiliser des mécanismes de génération automatique d'hypothèses conduit à implémenter un raisonnement non monotone, mais autorise la révision automatique de la connaissance. Par exemple, si, en l'absence de connaissance du sexe d'un individu nommé Claude, le système est capable de poser lui-même l'hypothèse que celui-ci est un homme, il devra retirer cette hypothèse si le fait que cet individu est une femme est ajouté. Un raisonnement est qualifié de non monotone quand la théorie — l'ensemble des théorèmes dérivés d'un ensemble d'axiomes — ne croît pas de manière monotone avec le nombre des axiomes. Ainsi, la théorie obtenue sans considération sur le sexe de l'individu contient le fait qu'il est de sexe masculin, à cause de l'hypothèse posée, alors que celle obtenue en sachant qu'il est de sexe féminin ne contient plus ce fait.

L'aide aux utilisateurs de systèmes de raisonnement hypothétique nécessite la conception d'algorithmes spécifiques, pour pouvoir gérer efficacement les hypothèses. L'implémentation d'un raisonnement hypothétique est traditionnellement faite à l'aide de systèmes de maintenance de la vérité (ou TMS pour "truth maintenance systems"). Le système de maintenance de la vérité est chargé de maintenir l'ensemble de ce qui est valide (`Déduits` sur la figure 1). Un mécanisme permettant le raisonnement hypothétique est donc composé d'*une partie produisant le raisonnement* (par exemple un moteur d'inférence), et

d'*un mécanisme de gestion des hypothèses et des conséquences qui en découlent* (le système de maintenance de la vérité). Ces deux mécanismes distincts interagissent suivant un protocole précis (voir figure 1).

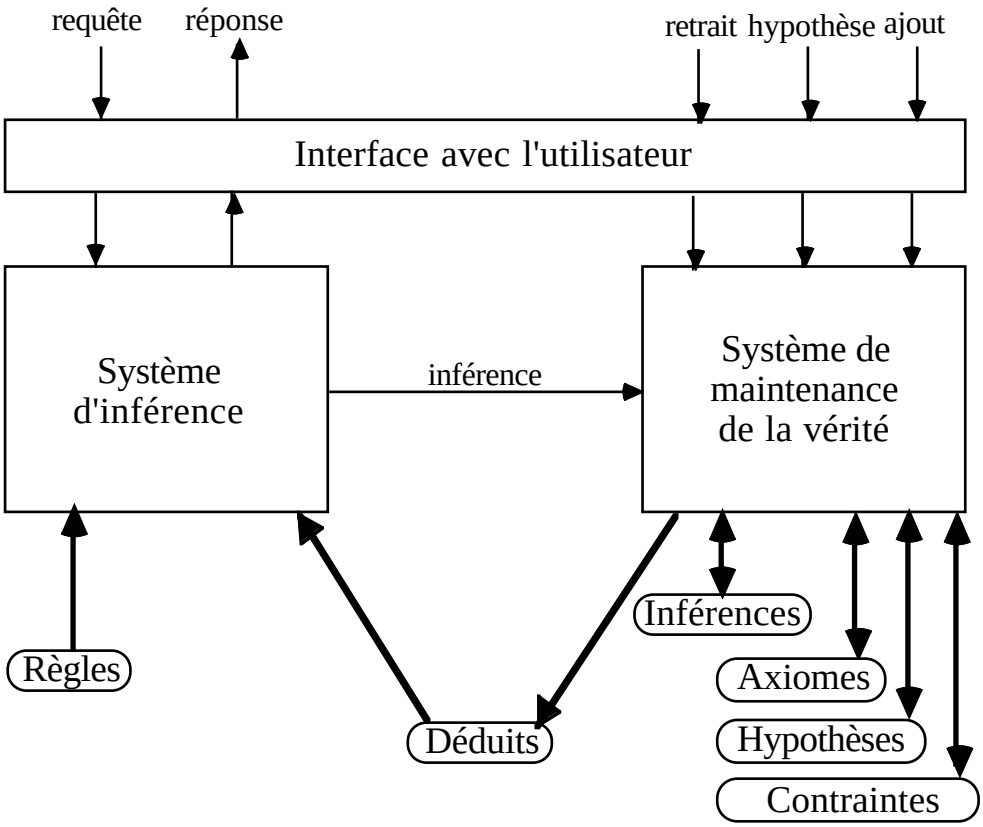


figure 1. Interface générale entre le système d'inférence et le système de maintenance de la vérité.

Une première constatation est que les systèmes de maintenance de la vérité sont des systèmes capables de représenter un raisonnement produit par un système d'inférence et de manipuler cette représentation. Il est en effet inconcevable de reproduire les raisonnements autant de fois qu'une théorie différente est considérée. Le rôle d'un système de raisonnement hypothétique sera donc de pouvoir passer rapidement d'une théorie à une autre en tirant parti d'une représentation du raisonnement. Pour cela, le système d'inférence communique au système de maintenance de la vérité chaque inférence qu'il produit sous la forme d'une justification. Ces justifications constituent un graphe de dépendances (`Inférences`). Le système de maintenance de la vérité reçoit aussi communication des axiomes du raisonnement, des hypothèses posées et des contraintes imposées à la base. De cette façon, les actions d'ajout et de retrait d'axiomes, ainsi que d'ajouts d'hypothèses peuvent lui être adressées directement.

En échange de quoi, le système de maintenance de la vérité est chargé de tenir à jour l'ensemble de ce qui a été inféré et doit être présent dans la base (`Déduits`), ensemble sur lequel le système d'inférence peut s'appuyer pour produire de nouvelles inférences.

Il existe deux grands types de systèmes de maintenance de la vérité: les systèmes à propagation et les systèmes à contextes. Les premiers utilisent des inférences non monotones pour poser automatiquement des hypothèses (par exemple: en l'absence de l'information «Claude est une femme» poser que «Claude est un homme») et se chargent d'établir la validité absolue des items («Claude doit donc avoir fait son service militaire»). Les seconds ont connaissance des hypothèses posées par l'utilisateur et établissent, pour chaque item inféré, sous quelles hypothèses il doit être considéré comme présent dans la base (par exemple: sous l'hypothèse que «Claude est un homme», «Claude doit avoir fait son service militaire»). Ceci permet de passer facilement d'un ensemble d'hypothèses à un autre: c'est le raisonnement multi-contextes. Dans ces derniers systèmes, les inférences ne peuvent pas être non monotones. Les deux types de systèmes sont capables de supprimer les hypothèses si elles mènent à une contradiction (par exemple: «Claude est un homme» et «Claude porte une jupe au bureau»). Le premier fera alors en sorte que les hypothèses posées ne soient plus considérées (ainsi, il va poser que «Claude est une femme» et supprimer de la base que «Claude doit avoir fait son service militaire»). Le second ne considérera plus les hypothèses incriminées («Claude est un homme», par conséquent, «Claude doit avoir fait son service militaire» ne doit plus être présent sous aucune hypothèse).

Ces deux systèmes ne peuvent être mis sur le même plan, il est clair qu'ils correspondent à deux conceptions différentes. Le premier ne doit pas être utilisé pour obtenir toutes les solutions à un problème, ni le second pour obtenir une solution unique. Plutôt que de les opposer, il vaut mieux les proposer à l'utilisateur et au concepteur, afin qu'ils choisissent le plus adapté à leur problème. Il est même souhaitable de les combiner afin de faire du raisonnement non monotone sur certains aspects du problème, et du raisonnement multi-contextes sur d'autres. Cela peut, par exemple, être utile pour mener un raisonnement à la fois hypothétique et temporel. Le raisonnement temporel nécessite bien souvent un raisonnement non monotone (pour exprimer la persistance: «Ce livre est sur le bureau au temps t_0», «tant qu'aucune information ne dit qu'il a été déplacé entre t_0 et t_i, supposer qu'il y est toujours à t_i»). Les hypothèses peuvent toujours être utiles («Supposons que Pierre en ait eu besoin hier et qu'il l'ait emporté chez lui»).

Ce travail s'attache à concevoir un système, appelé CP-TMS, permettant le raisonnement à la fois non monotone et multi-contextes. Les deux systèmes de maintenance de la vérité déjà connus seront donc exposés dans le premier chapitre. L'idée centrale de notre travail est que le caractère non monotone des inférences peut être pris en compte au sein des configurations d'hypothèses (appelées environnements) sous lesquelles les items doivent être considérés comme présents dans la base. Le travail d'un CP-TMS consiste alors à établir les environnements dans lesquels les formules sont présentes. Ainsi, «Claude doit avoir fait son service militaire» est présent dans un contexte où l'hypothèse «Claude est un homme» est posée, ou dans un contexte dont l'hypothèse «Claude est une femme» est absente.

Le deuxième chapitre définit donc des environnements qui permettent de tenir compte de cette non monotonie. Il précise comment interpréter ces environnements et comment les affecter aux formules. Ceci peut alors servir de spécification à une implémentation qui est présentée en détail dans le troisième chapitre. Elle est strictement une extension de celle des TMS à propagation mais se révèle être aussi une extension des TMS à contextes. Dans le dernier chapitre, cette implémentation est comparée aux autres et critiquée de façon à orienter la suite de la recherche vers une implémentation plus conforme aux spécifications des environnements. Cette spécification est, à son tour, discutée.

PREMIER CHAPITRE

Systèmes de maintenance de la vérité

Ce chapitre expose les différents systèmes de maintenance de la vérité qui ont pu être proposés: systèmes à propagation (§1) et systèmes à contextes (§2). Les limites de ces systèmes seront soulignées, avant d'exposer d'autres systèmes construits dans la perspective d'unifier les deux approches proposées (§3).

Premier chapitre

Systèmes de maintenance de la vérité

Systèmes de maintenance de la vérité

Les *systèmes de maintenance de la vérité* sont les algorithmes les plus éprouvés pour le traitement du raisonnement hypothétique. Comme cela a été évoqué, deux types de systèmes existent qui remplissent des tâches différentes et complémentaires: un étiquetage d'inférences non monotones ou un étiquetage multi-contextes. Afin de mettre en évidence comment ils remplissent respectivement leur tâche, ces deux systèmes vont être exposés.

Les deux paragraphes concernant les systèmes de maintenance de la vérité suivront grossièrement le même plan. Tout d'abord, la fonction du système sera définie et sa situation par rapport à l'interface commune exposée plus haut (voir figure 1) sera précisée. Puis, des exemples permettront de montrer comment procède l'algorithme pour arriver à ses fins. Enfin, les critiques adressables aux systèmes existants seront exposées lors de la conclusion.

Parmi ces critiques figure celle de la spécialisation de chacun des deux systèmes dans une tâche bien précise. Cette critique a déjà été faite et a donné lieu à des systèmes étendus tentant d'intégrer les deux aspects de la gestion du raisonnement hypothétique. Ces extensions seront exposées dans le troisième paragraphe de ce chapitre avant d'être critiquées à leur tour.

Les chapitres suivants permettront d'exposer la solution que nous présentons pour gérer ces deux aspects du raisonnement hypothétique. Elle se présente comme un prolongement des deux systèmes principaux exposés ici.

1.Systèmes de maintenance de la vérité à propagation

1.1.Principe

Un système de maintenance de la vérité est chargé, compte tenu d'un certain nombre d'inférences produites sur un ensemble de formules, de maintenir les raisons de croire ou non dans la validité des formules inférées. Tous les *systèmes de maintenance de la vérité à propagation* suivent globalement le même schéma introduit par Jon Doyle [Doyle 78, 79]. C'est un système auquel un moteur d'inférence fournit, sous forme de justifications, des photographies des inférences qu'il produit. Le système de maintenance de la vérité tient donc à jour la base de faits (`Déduits`, dans la figure 2) sur laquelle le moteur d'inférence — appelé ici plus généralement système de raisonnement — produit ses inférences.

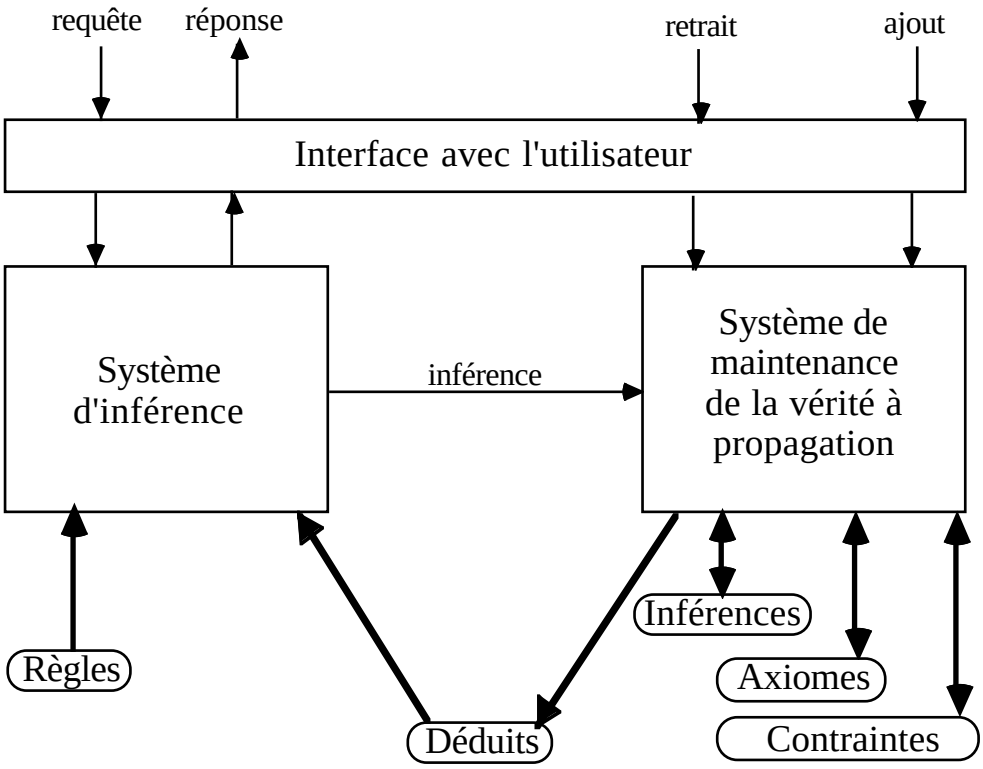


figure 2. Relations entre un système de raisonnement et le système de maintenance de la vérité à propagation. Contrairement au cadre général fixé par la figure 1, le système à propagation n'accepte pas d'hypothèses posées par l'utilisateur.

Le rôle du système est alors de maintenir la base de faits (`Déduits`) dans un état cohérent par rapport à toutes les données dont il a connaissance (`Axiomes`, `Inférences` et `Contraintes`). Pour cela, il va construire un graphe et établir avec son aide la présence (IN) et l'absence (OUT) de ses composants tout en contraignant les nœuds représentant des inconsistances à être absents (OUT).

À chaque nouvelle formule qui lui est communiquée, le système de maintenance de la vérité fait donc correspondre un *nœud*. Le système de raisonnement signale au système de maintenance de la vérité les inférences faites sous forme de *justifications* pour les faits inférés. Une justification d'un nœud est une paire d'ensembles de nœuds. Le premier ensemble est appelé *IN-liste*, le second *OUT-liste*. Intuitivement, une telle justification représente une inférence qui est valide en la présence des éléments de la IN-liste et en l'absence de ceux de la OUT-liste.

Le système de maintenance de la vérité manipule et maintient donc deux sortes d'entités: des nœuds représentant les formules et les justifications des nœuds. Il a trois actions de base sur ces entités:

- créer de nouveaux nœuds.
- ajouter une justification pour un nœud.
- marquer un nœud comme contradictoire.

L'ensemble des justifications (`Inférences`) communiquées au système de maintenance de la vérité est enregistré en un *graphe de dépendances* dans lequel les justifications sont les dépendances et les nœuds les sommets du graphe. Ce graphe est représenté à l'aide du symbolisme utilisé pour les circuits logiques. Une justification est alors une porte ET dont les entrées correspondant aux nœuds de la OUT-liste passent par un inverseur (dénoté sur les dessins par un simple rond). Un nœud est une porte OU dont les entrées sont des justifications. La figure 3 présente une liste de schémas caractéristiques de ce graphe de dépendances.

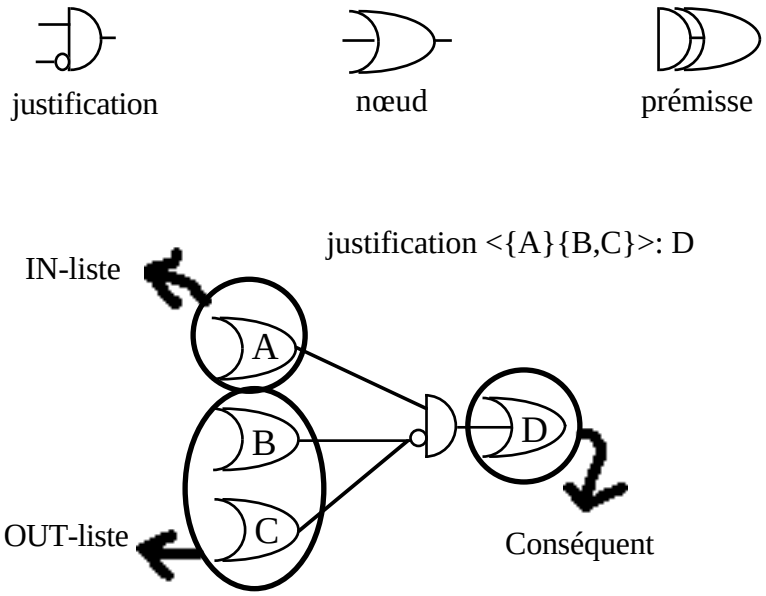


figure 3. Symbolisme utilisé dans la représentation des graphes de dépendances. D est présent dans la base si A est présent et B et C sont absents.

Jon Doyle propose les fondements suivants pour l'interprétation du graphe de dépendances:

- Pour un nœud F[1], si F a au moins une justification valide, F fait partie de l'ensemble des croyances courantes, F est dit *IN*. Si F n'a pas de justification valide, F est dit *OUT*.
- Une justification est *valide* si et seulement si tous les nœuds de sa IN-liste sont IN et tous les nœuds de sa OUT-liste sont OUT. Elle est dite *invalide* sinon.

Cette définition récurrente n'est pas infinie puisqu'il peut exister des nœuds sans justifications qui sont alors forcément OUT et des justifications ayant des IN- et OUT-listes vides qui sont forcément valides. Un étiquetage des nœuds du graphe correspondant aux critères ci-dessus est nommé un *étiquetage admissible* du graphe de dépendances.

Un nœud sera dit *en accord* ou *accordé* à une justification s'il est IN et fait partie de l'ensemble IN de la justification ou s'il est OUT et fait partie de son ensemble OUT. Dans le cas contraire, le nœud sera dit *en désaccord* ou *désaccordé* avec cette justification.

[1] Par convention, les nœuds du graphe sont nommés par la formule qu'ils représentent.

Tout au long de cette partie, les graphes étiquetés seront décrits à l'aide du symbolisme présenté ci-dessus en hachurant les nœuds OUT et justifications invalides et en laissant en blanc les nœuds IN et justifications valides.

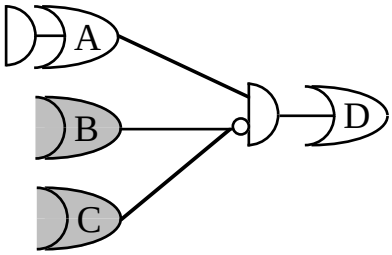


figure 4. Graphe de dépendances. Les composants représentés en hachuré sont invalides. A est un axiome, B et C sont présentement faux et D est inféré. L'étiquetage correspond donc bien aux contraintes logiques du «circuit» représenté et à la définition d'un étiquetage admissible.

Il est possible d'identifier divers nœuds à leurs justifications. Ainsi <{}{}> correspond à un axiome du raisonnement (une formule toujours vraie), <IN-liste {}> à une déduction classique et <{} OUT-liste> à une assertion présente par défaut.

Le travail du système de maintenance de la vérité consiste alors à trouver un *étiquetage faiblement fondé* (ou "not well-founded") des nœuds du graphe. Un étiquetage faiblement fondé est un étiquetage admissible dans lequel la validité d'un nœud ne dépend pas de sa propre validité. Il est ainsi nommé par rapport à un *étiquetage fortement fondé* (ou "well-founded") dans lequel la validité d'un nœud ne dépend pas de sa propre validité ou invalidité.

Cette distinction émerge tout droit de l'algorithme de Jon Doyle; elle sera donc mieux comprise après son exposé (voir en particulier figures 8 et 9). Une définition plus pragmatique revient à dire que l'étiquetage fortement fondé est celui trouvé par la première passe de l'algorithme du TMS et l'étiquetage faiblement fondé celui trouvé par la seconde. L'intérêt de la distinction est que l'étiquetage fortement fondé, s'il existe, est unique; ce n'est pas forcément le cas de l'étiquetage faiblement fondé. Cet étiquetage permet de déterminer les formules qui doivent être considérées comme présentes ou absentes de l'ensemble des formules déduites.

L'étiquetage est obtenu en effectuant la propagation de la validité et de l'invalidité au sein du graphe de dépendances construit comme indiqué ci-dessus (voir §1.2). Une fois un étiquetage obtenu, le système doit aussi vérifier que celui-ci ne viole pas de contraintes, c'est-à-dire qu'il ne met pas IN certains nœuds exprimant l'inconsistance. Si c'est le cas, un mécanisme de rétrogression est invoqué, pour forcer un autre étiquetage du graphe en ajoutant certaines justifications (voir §1.3).

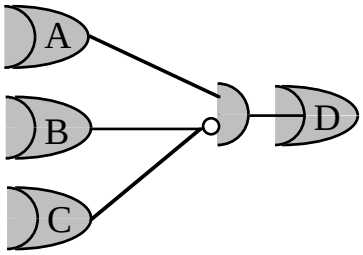
1.2.Propagation

L'algorithme de *propagation* permet d'obtenir un étiquetage faiblement fondé des nœuds du graphe. Les algorithmes utilisés par les systèmes de maintenance de la vérité sont incrémentaux, c'est-à-dire qu'ils réagissent à l'ajout dans le graphe d'une justification et étiquettent à nouveau la partie affectée par cette justification.

Pour chaque inférence correcte, le système de raisonnement fournit au système de maintenance de la vérité une justification. Cette justification est alors ajoutée au réseau de dépendances.

- Si la justification est invalide, ou si le nœud était déjà OUT grâce à une autre justification, l'étiquetage n'est pas affecté.
- Si le nœud justifié était OUT et la justification valide, il faut alors mettre à jour le réseau de dépendances en propageant les conséquences de cette nouvelle justification.

La mise à jour de ce réseau consiste en un algorithme de satisfaction de contraintes qui opère en deux passes sur une partie du réseau circonscrite à l'ensemble des nœuds pouvant être affectés par la modification. Le programme connaît, grâce aux données enregistrées, un ensemble restreint de nœuds qui peuvent être affectés: la clôture transitive des conséquents des justifications.

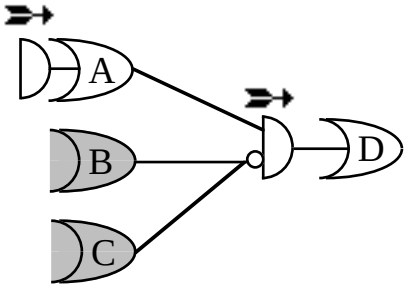


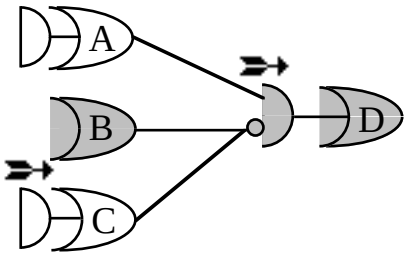
5a. 1e justification, j1= <{A}{B,C}>: D

Les nœuds sont créés automatiquement. Ils sont tout d'abord OUT (ils n'ont aucune justification). La justification j1 est invalide car A, dans sa IN-liste, est OUT. Dès lors, aucune propagation n'est nécessaire.

5b. 2e justification, j2= <{}{}>: A

Une telle justification dénote que A est posé comme axiome. Il est IN sans dépendre d'autres nœuds, la justification j2 étant valide inconditionnellement, A le devient aussi. Cette validité est propagée à la justification j1 dont tous les éléments de la IN-liste (A) sont IN et tous les éléments de la OUT-liste (B, C) sont OUT. Suite à la validité de j1, D devient IN.





5c. 3e justification, j3= <{}{}>: C

C'est au tour de C de devenir axiome, C devient donc IN. Cette validité doit être propagée à la justification j1. Contrairement à l'étape précédente, la validité de C entraîne l'invalidité de la justification j1 puisque C fait partie de sa OUT-liste. Cette invalidité doit, à son tour, être propagée pour mettre D OUT.

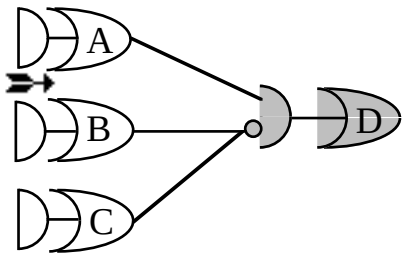
Par ailleurs, l'algorithme de Doyle utilise la *stratégie du support* pour faire la propagation. Celle-ci consiste à n'effectuer une propagation que si le nœud changeant d'état risque de modifier l'état de ses conséquents. Ainsi, si un nœud est IN, c'est que l'une de ses justifications est IN. Si une autre de ses justifications devient IN, cela n'aura aucune incidence sur l'état du nœud. Par contre, si ce nœud est OUT, c'est que toutes ses justifications sont invalides, dès lors, si l'une d'elles change d'état (devient valide) le nœud va devenir IN. Il est donc nécessaire, pour les nœuds IN, de ne considérer qu'une seule justification qui sera appelée *justification supportante* alors que pour les nœuds OUT toutes les justifications sont à considérer.

En poursuivant ce raisonnement plus avant, il faut noter que si une justification est invalide, tous ses antécédents doivent avoir leur état en accord avec la justification pour que celle-ci devienne IN. Il suffit donc de ne s'intéresser qu'à un seul des antécédents en désaccord avec la justification: tant que celui-ci n'est pas modifié la justification ne deviendra pas valide. Pour une justification valide, par contre, il suffit qu'un antécédent change d'état pour que cette justification devienne invalide: il faut donc considérer tous les antécédents d'une justification valide.

Ces antécédents à considérer sont appelés *nœuds supportants*. Si un nœud supportant d'un autre nœud change d'état il y a un risque pour que ce soit aussi le cas du second, mais tant que les nœuds supportants ne changent pas d'état il n'y a aucun risque. La stratégie du support consiste donc à ne propager les changements de validité des nœuds que le long des liens de support.

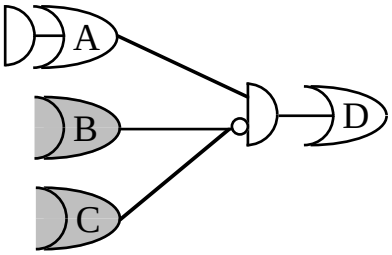
5d. 4e justification, j4= <{}{}>: B

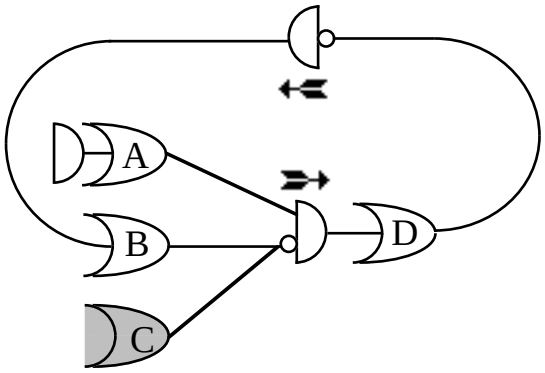
C'est maintenant B qui devient axiome et est donc mis IN comme précédemment. Mais, grâce à la stratégie du support, cette validité n'a pas à être propagée à la justification j1. En effet, le système a enregistré que le changement d'état de C est une condition *nécessaire* au changement de validité de j1. Ainsi, tant que C n'aura pas changé d'état, rien ne sera propagé vers j1.



Le fonctionnement de l'algorithme présenté plus haut n'est qu'une approximation de celui de Jon Doyle et correspond grossièrement à la première passe du système. En effet, la présence de cycles dans le graphe ne permet pas de garantir que cette passe trouvera un étiquetage admissible de celui-ci.

6a. reprise de la figure 5b





6b. j5= <{}{D}>: B

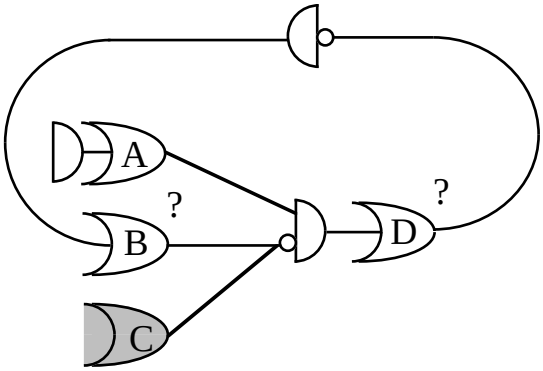
Cette justification formant un cycle dans le graphe, la propagation risque de ne jamais s'arrêter. Pour établir l'état de D, il faut connaître celui de B et réciproquement. Les figures suivantes montrent comment procède l'algorithme.

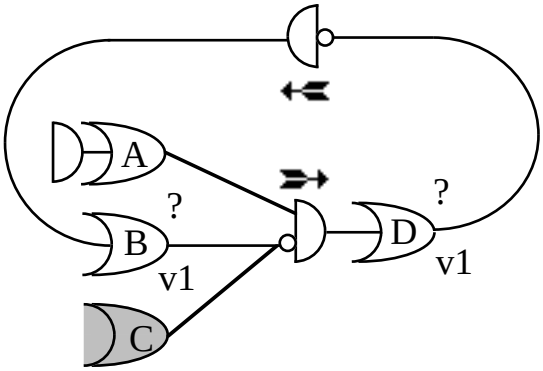
Pour trouver un étiquetage faiblement fondé du graphe, le système effectue la propagation en deux phases qui sont exposées ci-dessous. Il masque l'étiquette (*blanchissage*) de tous les nœuds qu'il aura à considérer et, lors de chaque passe, marque les nœuds qu'il a déjà examinés afin de ne pas boucler.

- La première passe trouve les supports fortement fondés s'ils existent, et en particulier, si le graphe est dénué de cycle. Cette passe permet d'attribuer des valeurs sûres aux nœuds qui ne sont pas partie prenante dans un cycle du graphe, ou qui ont une justification valide hors du cycle dans lequel ils sont impliqués. Mais, après celle-ci, les nœuds impliqués dans des cycles, et non validés par ailleurs par une justification externe au cycle, n'ont toujours pas d'étiquette.
- La seconde passe tente de déterminer de façon arbitraire une validation compatible avec le graphe. Elle prend un nœud auquel elle tente d'affecter l'état OUT. Les nœuds conséquents vont donc être déterminés IN ou OUT en fonction des justifications du cycle.

6c. Blanchissage

L'ensemble des nœuds concernés par la propagation consécutive à l'ajout de j5 (B et D) sont blanchis de leur état, c'est-à-dire qu'il prennent l'état inconnu (?).





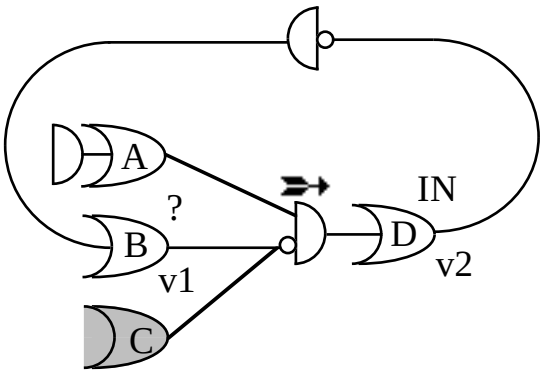
6d. Recherche des supports fortement fondés

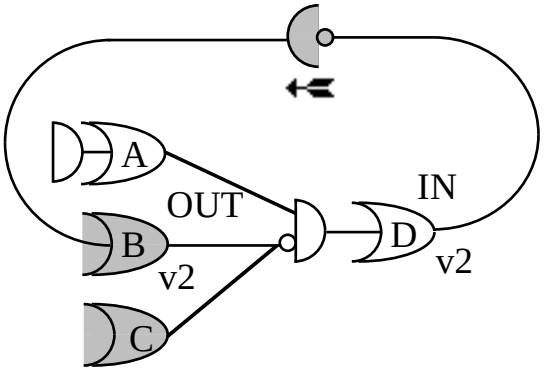
Partant de D, l'algorithme tente d'en établir la validité. L'antécédent B étant marqué ?, il ne peut le faire, D est donc marqué comme visité (v1) et la recherche se poursuit sur B où le même problème conduit à la même action. Tous les nœuds marqués (?) ayant été visités, cette première phase se termine sans avoir pu trouver d'étiquetage.

6e. Recherche des supports faiblement fondés

De nouveau, le système reprend sa recherche d'un support en D. Cette fois-ci, il considère que tous les nœuds encore étiquetés (?) sont OUT. La justification j1 est alors valide et D peut être étiqueté IN et marqué comme visité (v2). L'algorithme peut ensuite passer aux conséquents de D encore marqués (?).

Le fait de commencer par étiqueter les nœuds comme OUT permet d'obtenir un étiquetage faiblement fondé du graphe, c'est-à-dire qu'un nœud n'est jamais étiqueté IN à cause de lui-même (soit c'est à cause d'un antécédent, soit il est OUT).





6f. Recherche des supports faiblement fondés (suite)

D possède maintenant un état (IN) ce qui fait que la justification j5 est invalide. B ayant alors toutes ses justifications invalides, il peut être étiqueté OUT et marqué comme visité (v2).

L'algorithme pourrait se poursuivre de nouveau en D si l'étiquetage de B pouvait changer quelque chose à celui de D. Par chance, ce n'est pas le cas, l'algorithme peut donc s'arrêter.

Il faut noter que l'étiquetage obtenu est purement arbitraire. Si la recherche avait commencé au nœud B, l'étiquetage eût été celui de la figure 7.

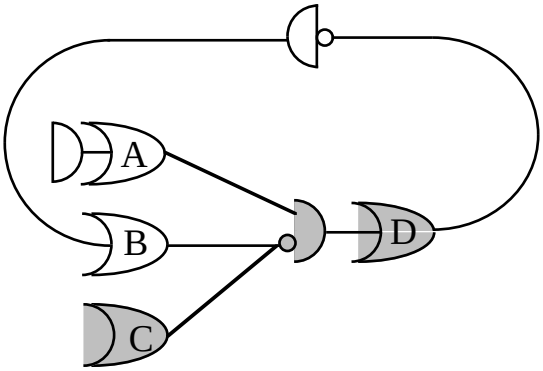


figure 7. L'un des deux étiquetages faiblement fondés possibles à l'issue de la seconde passe de l'algorithme.

Il faut remarquer que c'est à l'occasion d'un cycle dans le graphe que deux étiquetages différents apparaissent. À cet égard, il est possible de faire une typologie des cycles apparaissant dans un graphe de dépendances.

Deux types de circularités sont possibles: les *cycles pairs* — qui, sur les dessins traversent un nombre pair d'inverseurs — et les *cycles impairs* — qui traversent un nombre impair d'inverseurs. Les cycles pairs seront eux-mêmes divisés en *cycles pairs alternés* — qui traversent au moins un inverseur (et donc au moins deux) — et *cycles pairs stratifiés* — qui n'en traversent aucun.

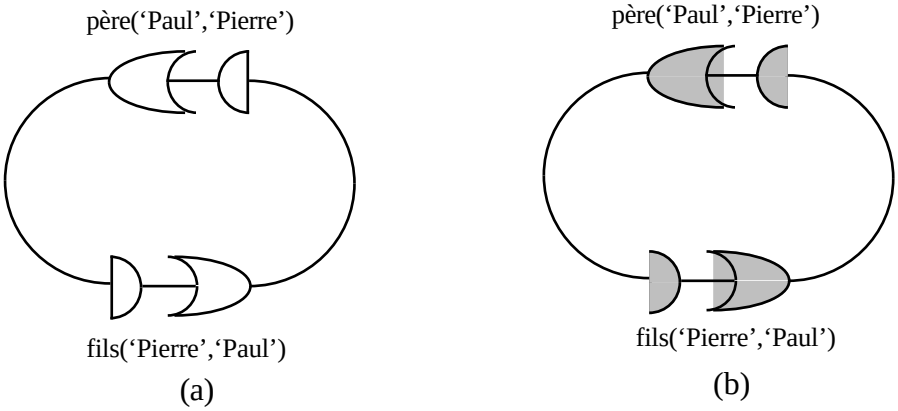


figure 8. Cycle pair stratifié. Les graphes correspondent aux deux inférences: si «Pierre est le fils de Paul» alors «Paul est le père de Pierre» et si «Paul est le père de Pierre» alors «Pierre est le fils de Paul». Ils constituent donc des cycles pairs stratifiés. Il en existe deux étiquetages admissibles, mais l’un d’entre eux est faiblement fondé (b) alors que l’autre est infondé (a).

À un cycle pair stratifié peut correspondre aussi deux affectations mais l’algorithme ne considère que celle où toutes les assertions sont OUT car il n’est pas normal qu’elles soient IN uniquement parce qu’elles se soutiennent l’une l’autre. L’affectation obtenue est faiblement fondée, l’autre est infondée.

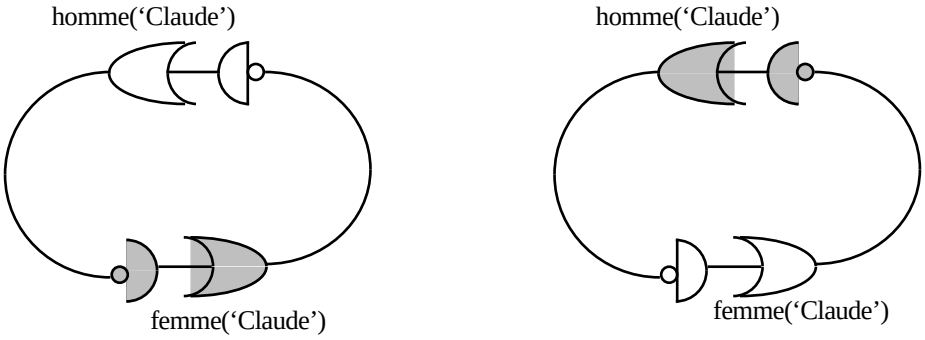
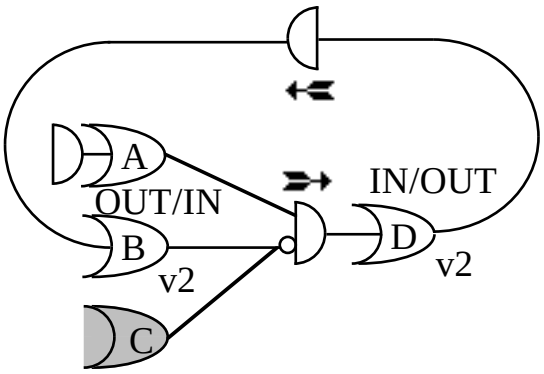


figure 9. Cycle pair alterné. Les graphes correspondent aux inférences si «Claude est une femme» est absent de la base alors poser que «Claude est un homme» et si «Claude est un homme» est absent de la base poser que «Claude est une femme». Ils constituent des cycles pairs alternés. Leurs étiquetages sont deux étiquetages faiblement fondés: la validité d’un item ne dépend que de l’invalidité d’un autre.

Un cycle pair alterné correspond à une alternative pour le système: c’est par un tel cycle qu’est exprimé, par exemple, un «ou exclusif». L’algorithme mettra certains nœuds du cycle IN et les autres OUT de façon cohérente, mais totalement arbitraire, par rapport au graphe de dépendances. Les choix ainsi faits pourront être revus en cas de contradiction ultérieure grâce au mécanisme de rétrogression qui est décrit plus loin.

À un cycle impair correspond un paradoxe dans l'ensemble des justifications et il n'y a donc pas d'affectation possible de la validité aux nœuds du cycle [Úrbanski 87]. Les cycles impairs ont la fâcheuse propriété de faire parfois boucler la seconde passe de l'algorithme. Si l'algorithme se trouve devant un conséquent OUT — qui a forcément été étiqueté au cours de cette seconde passe — pour lequel il trouve une nouvelle justification valide, ce nœud va redevenir IN. C'est malheureusement ce qui se passera si le cycle examiné est un cycle impair et c'est pourquoi l'algorithme peut ne pas se terminer quand il en rencontre un. Par contre, l'algorithme présenté ici trouve une assignation correcte si le graphe ne contient pas de cycles impairs.



10. Graphe de la figure 5b auquel est ajouté la justification j5= <{D}{}>: B.

Le travail sera le même jusqu'à la seconde phase. Le nœud D sera étiqueté IN au vu de l'état inconnu (?) de B considéré comme OUT. Cet étiquetage sera propagé au nœud B via la justification j5. B deviendra donc *IN*. Mais contrairement à ce qui s'est passé avant, cette nouvelle étiquette peut changer celle d'un des conséquents (D), la propagation reprendra donc vers D via j1, D deviendra OUT. De même, cette nouvelle étiquette va pouvoir influer sur l'état de B via j5…

L'algorithme ne s'arrêtera plus.

Il faut noter que l'algorithme ne boucle pas toujours car les cycles impairs peuvent être contraints par d'autres justifications qui forcent un des nœuds du cycle à être IN (voir figure 11).

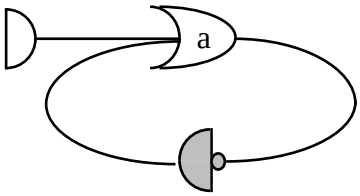


figure 11. Une justification permettant l'affectation non contradictoire d'un cycle impair.

Dans l'absolu, l'algorithme de maintenance de la vérité à propagation ne peut donc être tenu pour sûr en présence de cycles impairs. Néanmoins, si le graphe en est exempt, l'algorithme trouvera un des étiquetages possibles, permettant ainsi de proposer au système d'inférence une base de faits cohérente sur laquelle il pourra faire ultérieurement des inférences.

1.3.Rétrogression

Après avoir effectué la propagation consécutive aux justifications fournies au système, celui-ci doit vérifier qu'aucune *contrainte* n'est violée. Pour plus de compréhension, une contrainte sera représentée par un nœud nommé ⊥ et sa violation par l'étiquetage comme IN de ce nœud. Par exemple, la contrainte spécifiant que Claude ne peut être à la fois un homme et une femme s'exprimera par la justification:

<{homme('Claude'),femme('Claude')}{}>: ⊥

Le rôle du système de maintenance de la vérité est donc d'éviter que le nœud ⊥ soit étiqueté IN. Pour cela, il procède à la propagation de la manière évoquée ci-dessus, puis, si ce nœud est IN, il invoque le mécanisme de *rétrogression* (ou retour-arrière).

Le principe mis en œuvre s'appuie sur le fait que le raisonnement se produit en connaissance incomplète et que s'il y a inconsistance, c'est qu'une hypothèse posée l'a été à tort. L'idée est de remettre en cause certaines hypothèses afin d'éliminer les nœuds contradiction de l'ensemble des nœuds IN. Les hypothèses étant posées par le système de raisonnement sous la forme d'inférences non monotones, celui-ci est donc parfaitement capable de les isoler. Grâce au graphe de dépendances, il est aussi capable d'isoler celles qui ont permis d'étiqueter le nœud ⊥ comme IN. Une fois ces hypothèses localisées, il va tenter de bloquer la génération de l'une d'entre elles (appelée la *coupable*). Pour que celle-ci devienne OUT, il faut que l'un des faits qui faisaient défaut et qui permettaient ainsi l'inférence ne fasse plus défaut. Cela se fait simplement en complétant la connaissance disponible, c'est-à-dire en ajoutant à la base un nœud (appelé l'*élu*) membre de la OUT-liste de la justification qui soutient la génération de l'hypothèse.

Pour cela, une justification, justifiant la présence de l'élu, est ajoutée à la base. Cette justification oblige l'élu à être présent quand toutes les autres conditions de la violation de la contrainte sont réunies.

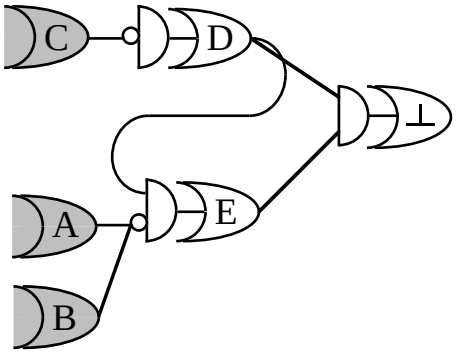
Voyons plutôt comment cette rétrogression agit. L'algorithme exposé ici n'est pas exactement celui de Jon Doyle, il est plus simple mais enrichi de certaines remarques faites par Charles Petrie [Petrie 87]. Fonctionnellement, l'algorithme doit ajouter dans la base une justification valide pour un nœud actuellement OUT. La justification de ce nœud doit permettre:

(1) d'invalider (mettre OUT) le nœud contradictoire.
(2) que tout redevienne normal si l'une des raisons de la contradiction venait à changer d'état. C'est-à-dire que, si cela n'est pas nécessaire pour la consistance de la base, le nœud ne doit pas être validé par cette justification.
(3) de ne pas ajouter de cycle impair dans le graphe de dépendances.

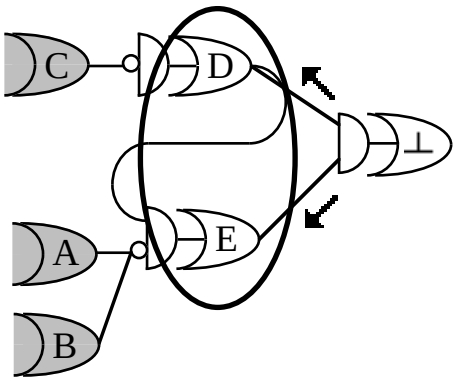
12a. Détection d'inconsistance.

Examinons le graphe ci contre, dont on peut aisément vérifier qu'il est correctement étiqueté.

Le nœud ⊥ y est étiqueté IN. Il faut donc procéder à la rétrogression. L'ensemble des nœuds antécédents du nœud inconsistant est constitué des nœuds A, B, C, D et E.



L'algorithme de Jon Doyle utilise les *hypothèses* ancêtres du nœud inconsistant à invalider, c'est-à-dire les nœuds qui sont à la fois IN et justifiés par une justification non monotone. Il isole toutes les *hypothèses maximales,* c'est-à-dire celles qui n'ont pas parmi leurs conséquents d'autres hypothèses. Choisir de remettre en cause une hypothèse maximale entraîne moins de modifications au sein du graphe que de remettre en cause une hypothèse non maximale, l'algorithme tente donc de faire le moins possible de modifications dans la base. Il choisit la coupable parmi ces hypothèses et l'élu parmi les antécédents non monotones de la coupable.



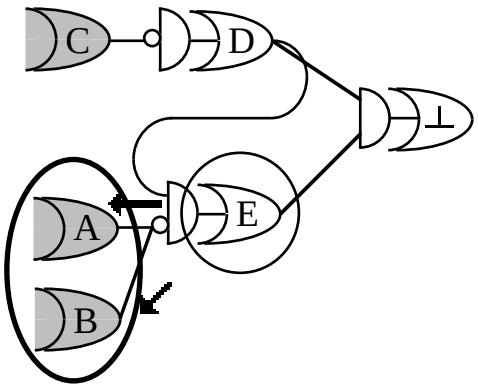
12b. Hypothèses.

Parmi les antécédents il est possible d'isoler deux hypothèses: D et E, ce sont les seuls nœuds à être IN et à être justifiés par une justification non monotone.

12c. Hypothèses maximales, coupables et antécédents non monotones.

Parmi ces hypothèses, une seule est maximale: E. En effet, D étant antécédent de E, si la validité de D est changée, cela changera aussi celle de E. Dans le but de faire un minimum de modification de l'étiquetage, la rétrogression se fera donc sur E qui n'oblige pas à remettre en cause l'étiquette de D. E sera donc la coupable.

L'élu doit être choisi dans les antécédents non monotones de E. Il en existe deux: A et B.

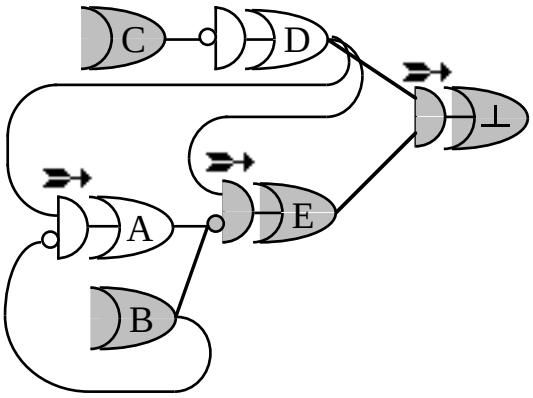


La justification ajoutée va permettre d'assurer la présence de l'élu. Elle possède une IN-liste composée:

- de l'ensemble des hypothèses antécédentes du nœud $\bot$ sans le coupable
- de la IN-liste de la justification du coupable.

La OUT-liste contient la OUT-liste de la justification de la coupable sans l'élu. Pour une coupable faisant partie des hypothèses (Hyp) et une justification J entre elle et l'élu, la justification est la suivante:

<Hyp\{coupable}∪INLISTE[J] OUTLISTE[J]\{elu}>: elu.



12d. Élu et justification.

Si le nœud A est l'élu, alors la rétrogression va lui ajouter une justification. Les hypothèses {D, E} sans le coupable {E} en constitueront la IN-liste et les antécédents non monotones {A, B} sans l'élu {B} la OUT-liste.

La justification ajoutée est donc <{D}{B}>: A. On peut vérifier que l'étiquetage du graphe est de nouveau correct et que l'inconsistance a disparu.

L'algorithme de Jon Doyle utilise donc la notion d'hypothèse conjuguée avec la stratégie des hypothèses maximales et en dernier ressort un choix au hasard. Mais il va beaucoup plus loin car il enregistre la rétrogression au sein des dépendances. En effet, l'ensemble des hypothèses maximales isolées par l'algorithme correspond à l'implication[2]

$$H_1 \wedge \ldots H_p \Rightarrow \bot.$$

Si l'hypothèse H_j est inférée par

$$I_1 \wedge \ldots I_n \wedge \neg O_1 \wedge \ldots \neg O_m \Rightarrow H_j.$$

la formule O_i peut être choisie pour invalider H_j. La rétrogression s'effectue alors en ajoutant au sein du système l'inférence

$$I_1 \wedge \ldots I_n \wedge H_1 \wedge \ldots H_{j-1} \wedge H_{j+1} \wedge \ldots H_p \wedge \neg O_1 \wedge \ldots \wedge \neg O_{i-1} \wedge \neg O_{i+1} \wedge \ldots \neg O_m \Rightarrow O_i.$$

[2] Pour une plus grande compréhension, les inférences sont ici présentées sous forme d'implications, les membres de la OUT-liste étant niés.

Ainsi, si l'un des antécédents positifs devenait OUT (ici valide) ou l'un des antécédents négatifs devenait IN (ici invalide), la rétrogression n'aurait plus lieu d'être et il ne serait plus nécessaire de conserver la formule O_i ajoutée à la base. La phase de *revalidation* (c'est-à-dire de rétablissement des hypothèses incriminées à tort, "unouting") en cas de choix d'une autre hypothèse à remettre en cause devient automatique (voir figure 15).

Il faut noter que l'étiquetage obtenu est, encore une fois, arbitraire puisqu'il dépend du choix de l'hypothèse et de l'élu. Si l'élu avait été le nœud B, l'étiquetage eût été celui de la figure 13. Il y a donc deux sources de «non déterminisme» dans l'étiquetage par un système de maintenance de la vérité à propagation, qui sont:

- les cycles pairs,
- le choix des nœuds considérés lors de la rétrogression qui se décompose en deux choix: celui de l'hypothèse à abandonner et celui de l'élu à ajouter.

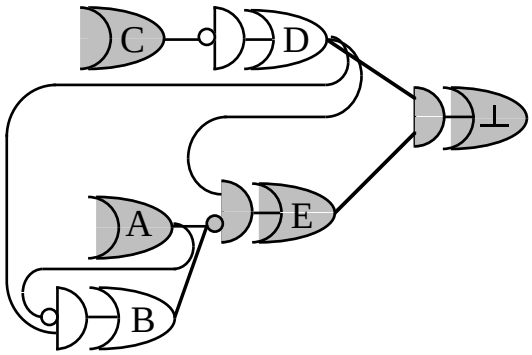


figure 13. L'autre possibilité pour le système de supprimer l'inconsistance.

Jusqu'ici, cet algorithme ne remplit pas la troisième condition (ne pas ajouter de cycles impairs) ce qui est grave dans la mesure où ces derniers peuvent ultérieurement faire boucler la propagation. Charles Petrie [Petrie 87] a construit un algorithme vérifiant, à chaque fois qu'il résout une contradiction, si la justification ajoutée n'introduit pas de cycle impair. Cette méthode s'intègre aisément dans l'algorithme décrit ci-dessus.

Soit Rep-OUT(N) l'ensemble des nœuds accessibles à partir de N en traversant un nombre impair de OUT-listes et Rep-IN(N) l'ensemble des nœuds accessibles à partir de N en traversant un nombre pair de OUT-listes; un cycle impair actif sera ajouté au système si:

$$[\text{IN-liste}(J) \cap \text{Rep-OUT}(N)] \cup [\text{OUT-liste}(J) \cap \text{Rep-IN}(N)] \neq \{\}$$

où J est la justification à ajouter et N l'élu. Cependant ces données ne sont pas suffisantes pour déterminer si un cycle impair est ajouté dans l'absolu et ne pourra donc pas resurgir plus tard. L'ajout d'un cycle dans le graphe est détecté si un nœud de la clôture transitive du conséquent de l'élu est impliqué dans la nouvelle justification. Petrie signale que le test de l'ajout d'un cycle impair dépend du nombre de OUT-listes traversées par le cycle sans préciser son algorithme. Si un cycle impair est détecté, l'algorithme tente d'assurer la présence d'un autre élu, et s'il n'y a plus d'élu possible, d'invalider un autre coupable. Quand il n'y a pas d'autres hypothèses, l'algorithme s'arrête en signalant qu'il ne peut résoudre la contradiction.

Il existe, toutefois, une classe de graphes qui ne permettent pas de résoudre leur contradiction avec l'algorithme décrit (voir figure 14).

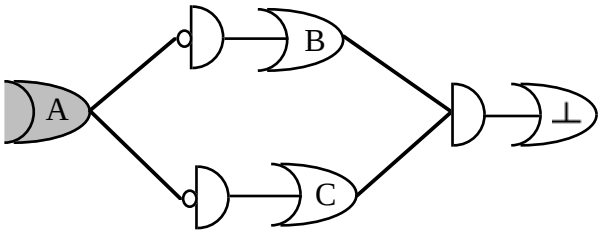


figure 14. Le graphe le plus simple contenant une contradiction insoluble par l'algorithme précédent mais contenant des hypothèses[3]. La rétrogression entraînerait la création de cycles impairs. Pour résoudre le problème, il faut justifier A avec la justification <{}{}>.

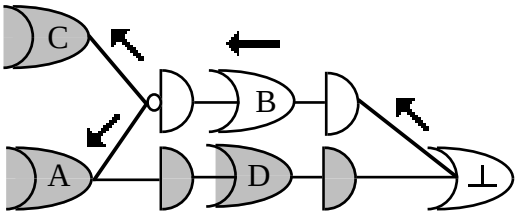
L'algorithme exposé ci-dessus peut nécessiter plusieurs itérations car il arrive qu'il ne résolve pas l'inconsistance la première fois. En effet, le nœud ajouté à la base peut amener d'autres sources d'inconsistances. C'est le cas pour le graphe suivant.

[3] Graphe communiqué par Charles Petrie.

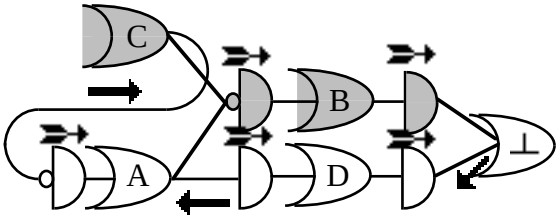
15a. Première rétrogression.

Examinons le graphe ci contre, dont on peut aisément vérifier qu'il est correctement étiqueté.

Le nœud ⊥ y est étiqueté IN. Il faut donc procéder à la rétrogression. L'ensemble des nœuds antécédents du nœud inconsistant est constitué des nœuds A, B, C et D. L'ensemble des hypothèses maximales est constitué du nœud B ayant A et C pour antécédents non monotones. Si le système choisit d'ajouter A à la base, il construira la justification présente sur le dessin suivant: <{}{C}>: A

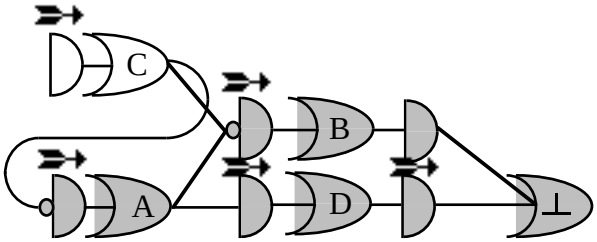


15b. Propagation et seconde rétrogression.



Après cette première rétrogression, le nœud ⊥ est toujours IN. Relancer une nouvelle fois la rétrogression va permettre d'isoler un ensemble d'hypothèses (A) qui ne possède qu'un seul antécédent non monotone (C). Celui-ci sera donc justifié par une nouvelle justification: <{}{}>: C

15c. Seconde propagation et rétablissement de la consistance.



Tout rentre alors dans l'ordre, le nœud ⊥ est étiqueté OUT. Il faut noter au passage que comme l'invalidité du nœud A ne risque plus d'engendrer une inconsistance, celui-ci est redevenu OUT.

Grâce à ce mécanisme, la rétrogression remplit bien les conditions qui lui ont été assignées plus haut, à savoir étiqueter tous les nœuds ⊥ comme OUT en complétant la base de connaissance si c'est nécessaire.

1.4.Conclusion

Depuis sa première présentation par Doyle, l'algorithme de maintenance de la vérité a beaucoup évolué. Les innovations vont dans le sens d'une plus grande rapidité de l'algorithme (décomposition du graphe de dépendances en composantes fortement connexes [Goodwin 87], arrêt de la propagation dès qu'elle devient inutile [Euzenat 88], matérialisation des composantes et parallélisation de l'algorithme [Quintero 89]) et d'une meilleure maîtrise de l'étiquetage résultant (rétrogression dirigée par les règles [Petrie 87], rétrogression détectant les étiquetages multiples [Euzenat 89c]). Comme cela a été montré, les algorithmes sont incomplets, cependant ils sont utiles pour résoudre la plupart des problèmes rencontrés.

L'évolution des algorithmes va vers une plus grande compréhension par le programme de la topologie du graphe de dépendances et par une résolution plus «intelligente» des problèmes posés.

D'une manière générale, le principal reproche fait à l'algorithme de Jon Doyle est qu'il est lent et qu'il prend beaucoup de place. Il a été amélioré mais, vu le travail produit par un tel système, il faut s'attendre à ce qu'il consomme beaucoup de ressources.

Johan De Kleer [De Kleer 84] a relevé un certain nombre d'insuffisances des algorithmes de maintenance de la vérité à propagation quand ils sont utilisés à certaines tâches précises:

- Il est très compliqué, avec le système de Doyle, d'agir sur les hypothèses pour constater les changements d'état. En effet, ôter ou réactiver une hypothèse relance tout le travail de propagation.
- Le mécanisme de rétrogression dirigée par les dépendances agit sans vision globale du raisonnement, il change parfois sans résultat l'état des nœuds et provoque de nouvelles contradictions au cours de sa résolution (voir figure 15). Le système de maintenance de la vérité en devient très lent.
- Quand une donnée a été remise en cause (OUT), et qu'elle est revalidée, le système de maintenance de la vérité parcourt tout le réseau de dépendances pour retrouver ce qui avait déjà été inféré.
- Le problème de l'état unique: le système de maintenance de la vérité, lors de la rétrogression comme de la propagation, amène le système dans un état particulier et unique. Or il existe d'autres étiquetages possibles. Par ailleurs, la rétrogression ne tient pas compte de ces autres étiquetages. Il existe peut-être d'autres moyens de résoudre cette contradiction qui mèneraient à ces étiquetages plus intéressants pour le module de raisonnement.

Ces reproches sont fondamentaux et c'est à partir d'eux que Johan De Kleer a bâti son système de maintenance de la vérité à contextes. Ils révèlent entièrement les lacunes du système à propagation auxquelles remédie le système à contextes.

En résumé, le système de maintenance de la vérité à propagation remplit bien son rôle de mémorisateur des inférences. Il est capable de rétablir la cohérence de la base face à chaque nouvelle inférence et à chaque découverte de contradiction. Il est capable de gérer un seul ensemble d'axiomes à la fois mais supporte les inférences non monotones.

2.Systèmes de maintenance de la vérité à contextes

Les systèmes de maintenance de la vérité à contextes, au lieu de propager la validité absolue des nœuds du graphe de dépendances, propagent les ensembles d'hypothèses sous lesquelles ces nœuds sont IN[4]. La présentation qui suit est principalement fondée sur l'ATMS (pour système de maintenance de la vérité fondé sur les hypothèses, "assumption-based TMS") de Johan De Kleer [De Kleer 86a]. Tout d'abord, le principe du système est introduit (§2.1) avant d'en définir les structures de données importantes (§2.2). Ensuite, les procédures activées consécutivement à la déclaration d'une justification ou à la découverte d'un nouvel ensemble d'hypothèses inconsistant (§2.3) sont décrites. La conclusion examinera les problèmes et limitations des systèmes à contextes.

2.1.Principe

Comme le système de maintenance de la vérité à propagation, le système de maintenance de la vérité à contextes a pour but de conserver les données inférées par un système de raisonnement et de déterminer leur validité. Afin de permettre au module de raisonnement d'avoir accès à tous les états possibles, De Kleer propose d'enregistrer les ensembles d'hypothèses sous lesquelles un nœud est présent plutôt que le fait qu'un nœud soit présent. Le système sera alors capable de raisonner simultanément avec divers ensembles d'hypothèses (appelés ici environnements).

L'architecture fonctionnelle d'un tel système sera remarquablement semblable à celle d'un système de maintenance de la vérité à propagation (voir figure 16). Le protocole de communication entre le système de maintenance de la vérité et le module de raisonnement est complètement défini dans [De Kleer 86c]. Le système de raisonnement fournit au système de maintenance de la vérité les inférences qu'il produit sous forme de justifications: c'est le système de maintenance de la vérité qui introduit les résultats de ces inférences dans la base de faits que manipule le système de raisonnement.

[4] Dans toute la partie consacrée aux systèmes à contextes, «IN» et «OUT» seront respectivement remplacés par «valide» et «invalide» en raison de la parenté avec la logique.

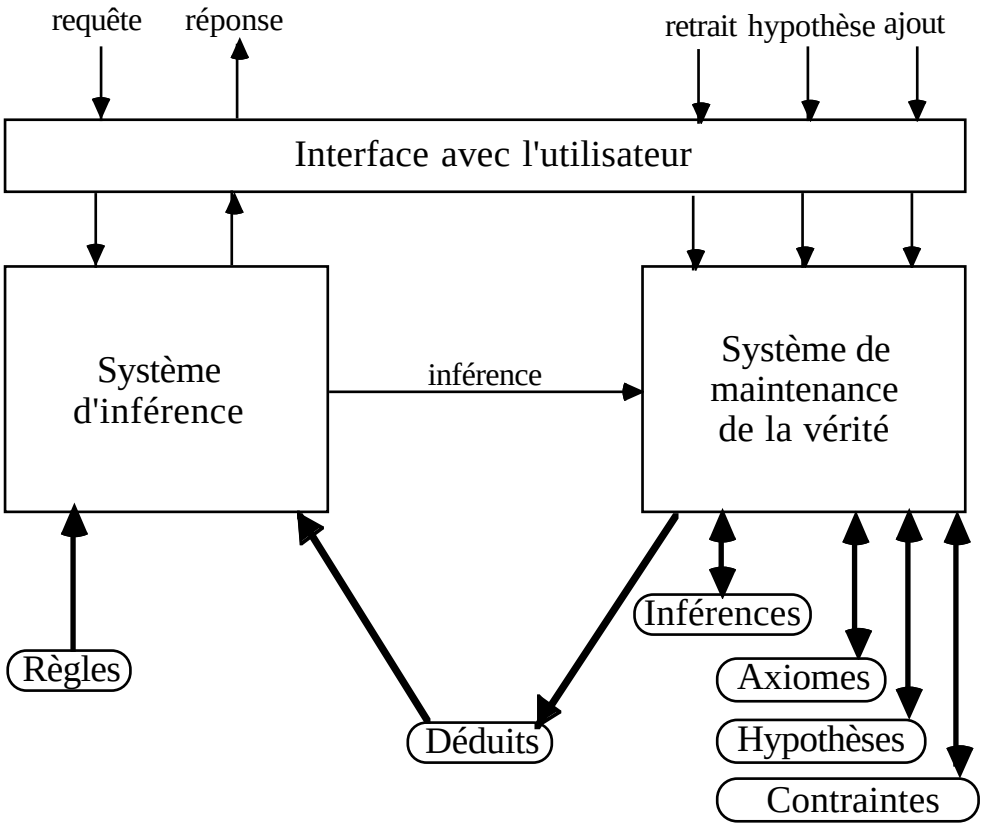


figure 16. Architecture d'un système utilisant un système de maintenance de la vérité à contextes. Ce schéma correspond à celui de la figure 1. Ici, contrairement au système de maintenance de la vérité à propagation (voir figure 2), les hypothèses sont prises en compte mais les inférences considérées doivent être monotones.

Le système de maintenance de la vérité à contextes possède trois opérations de base:

- créer un nœud,
- ajouter une justification à un nœud,
- créer une hypothèse.

Il opère de manière incrémentale, c'est-à-dire qu'il réagit à chacune des sollicitations du système de raisonnement pour enrichir la base de nœuds.

Le système de maintenance de la vérité n'interprète pas les formules manipulées par le système de raisonnement. Les *justifications* sont fournies sous la forme

$$<\{A_1,\ldots A_n\}> : B$$

signifiant que si $A_1,\ldots A_n$ sont présents alors B est présent. Les justifications ne sont donc plus fondées sur l'absence de certaines formules de la base. Comme on le verra plus loin, cela réduit le raisonnement manipulé à être monotone. Comme au sein du système à propagation, il existe des *contraintes* qui sont représentées par l'inférence d'une formule $\perp$. Elles sont donc communiquées au système sous la forme:

$$<\{A_1,\ldots A_n\}> : \perp$$

qui signifie que $\{A_1,\ldots A_n\}$ est un ensemble de formules inconsistantes (un tel ensemble est aussi appelé “nogood”). $\perp$ n’est donc pas une formule du système de raisonnement mais permet d’exprimer sous la forme d’une justification que les nœuds antécédents de celle-ci ne doivent pas être simultanément présents dans la base. En fait, cela revient à considérer que le nœud $\perp$ ne doit pas être présent dans la base.

Une *hypothèse* est une formule propositionnelle qui est prise pour base du raisonnement, en principe, parce que sa validité est inconnue: elle n’est ni prouvée, ni niée. Les hypothèses vont servir à «paramètrer» le raisonnement. Les hypothèses peuvent être communiquées soit par le système d’inférence (bien que ce ne soit pas sa fonction première), soit par l’utilisateur, soit par un autre mécanisme.

Déclarer ainsi les inférences va permettre de les représenter, comme au sein du système à propagation, par un graphe de dépendances. À la différence de celui-ci, les inverseurs vont disparaître des justifications. Par contre, les hypothèses seront représentées par des portes OU, comme les autres nœuds, mais dont le contour est en gras.

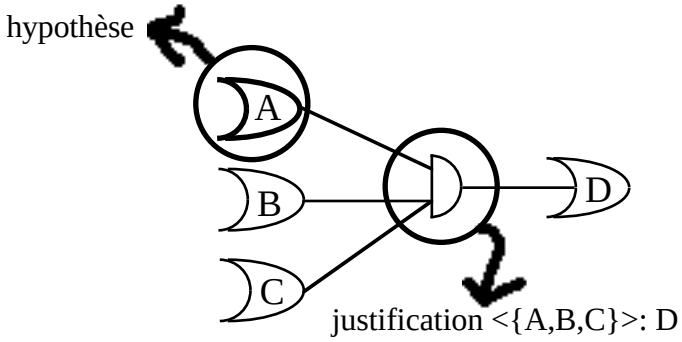


figure 17. Symbolisme utilisé dans la représentation des graphes de dépendances de l’ATMS.

À un moment donné, le module de raisonnement a fourni au système de maintenance de la vérité un certain nombre de nœuds (parfois dénotés par n_X pour le nœud associé à la formule X), un certain nombre d’hypothèses et un certain nombre de justifications qui représentent toutes les inférences produites par celui-ci.

Un *environnement* est un ensemble d’hypothèses. Un nœud sera présent ou absent sous un environnement E s’il l’est suivant l’étiquetage du graphe de dépendances où les hypothèses de E sont considérées comme présentes. À chaque nœud N manipulé par le système, est associé une *étiquette*, c’est-à-dire un ensemble d’environnements, notée L(N) ou ETIQUETTE[N]. La fonction de cette étiquette est de contenir l’ensemble des environnements sous lesquels le nœud est présent.

La construction et la maintenance des étiquettes est détaillée plus loin mais certaines configurations sont remarquables: une étiquette vide {} est attachée à un nœud qui n'est présent sous aucun environnement consistant; une étiquette contenant l'environnement vide {{}…} est attachée à un nœud présent sous tout environnement consistant (un axiome); une étiquette d'un nœud H contenant un environnement constitué par le seul nœud {{H}…} est attachée à une hypothèse.

Au niveau de la base de faits (`Déduits`), la différence entre les deux systèmes est qu'elle ne contient plus uniquement les formules considérées comme présentes dans la base à un moment mais toutes les formules communiquées auxquelles sont associés leurs étiquettes. Les nœuds ne sont donc plus étiquetés absolument (IN/OUT), mais relativement aux environnements qui leur permettent d'être IN. La représentation du graphe ne contiendra alors plus de parties hachurées représentant l'étiquetage.

Un *contexte* est l'ensemble des nœuds qui sont présents sous un environnement précis. Le contexte contient donc toutes les hypothèses de l'environnement ainsi que tous les nœuds qui doivent être présents quand les hypothèses sont présentes. Si à un environnement correspond un contexte unique, la réciproque n'est pas vraie. L'interrogation de la base, afin de savoir si une formule est valide ou non, peut donc se faire dans le cadre d'un contexte précis représenté par un ensemble d'hypothèses. Le module de raisonnement, suivant le contexte dans lequel il veut travailler (un ensemble d'hypothèses qu'il fait), peut ainsi consulter l'état de ses données.

Le travail du système de maintenance de la vérité à contextes est donc, à chaque fois qu'une nouvelle justification lui est communiquée, d'établir pour tous les nœuds manipulés leurs étiquettes, c'est-à-dire les ensembles d'hypothèses sous lesquelles ils sont présents dans la base. Si cette première phase permet de trouver un nouvel environnement dans lequel le nœud $\bot$ est présent (c'est-à-dire qu'une contrainte est violée), il faut alors purger toutes les étiquettes des environnements violant la contrainte.

2.2.Hypothèses et environnements

Dans l'ATMS, la notion importante est celle d'hypothèse. C'est un nœud comme les autres qui va avoir un statut spécial: celui d'être distingué par l'utilisateur pour essayer divers branches de raisonnements possibles. Il peut poser l'hypothèse que «Claude est un homme» et observer les conséquences que cela entraîne, par exemple pour sa situation vis-à-vis du service national. Il peut déduire que «Claude doit être dégagé des obligations militaires».

Les ensembles d'hypothèses vont donc indexer les étiquetages possibles du graphe de dépendances. Pour cela, à chaque nœud est associé l'ensemble des environnements sous lesquels le nœud est présent.

En présence d'un ensemble de justifications J, le fait qu'un environnement E permette d'établir la présence dans la base d'un nœud N est noté par[5]:

$$J,E \vdash N.$$

Si E représente l'ensemble d'hypothèses $\{H_1,\ldots H_n\}$, cela signifie qu'en connaissance de l'ensemble de justifications J et en la présence dans la base des nœuds $H_1,\ldots H_n$, le nœud N doit être présent dans la base.

Les étiquettes doivent posséder un certain nombre de propriétés. Les deux plus importantes sont celles de correction et de complétude. Une étiquette est *correcte* si chaque environnement de l'étiquette associé à l'ensemble de justifications fournies par le système de raisonnement assure la présence de la formule correspondant au nœud. Cela s'exprime par

$$\forall E,\ E \subseteq L(N) \Rightarrow J,E \vdash N.$$

Une étiquette doit par ailleurs être *complète*, c'est-à-dire que chaque environnement qui assure la présence de la formule au vu des justifications fournies par le système de raisonnement est dans l'étiquette. Cela s'exprime par

$$\forall E,\ J,E \vdash N \Rightarrow E \subseteq L(N).$$

Si les étiquettes étaient exhaustives, c'est-à-dire si elles contenaient tous les environnements permettant la présence du nœud, tout serait parfait mais cela occuperait beaucoup de place. À partir d'un ensemble de n hypothèses, il est possible de définir 2^n sous-ensembles c'est-à-dire 2^n environnements distincts. Ainsi, suivant les deux critères donnés plus haut, l'étiquette d'un nœud présent universellement devra contenir tous ces environnements.

Heureusement, la partition de l'ensemble d'hypothèses s'organise naturellement dans un treillis structuré par la relation d'inclusion (voir figure 18). Ce treillis va permettre de n'étiqueter les nœuds que sous certains environnements: les environnements les plus généraux (c'est-à-dire ceux qui sont inclus dans d'autres). Ces environnements dénotent l'ensemble des environnements qui leur sont plus spécifiques (l'ensemble des environnements dans lesquels ils sont inclus).

---

[5] Le signe $\vdash$ est utilisé dans le sens de «permet d'établir la présence de». En fait, la notation a été empruntée à la logique en raison du rapport étroit entre l'ATMS et le calcul des propositions [Reiter& 87].

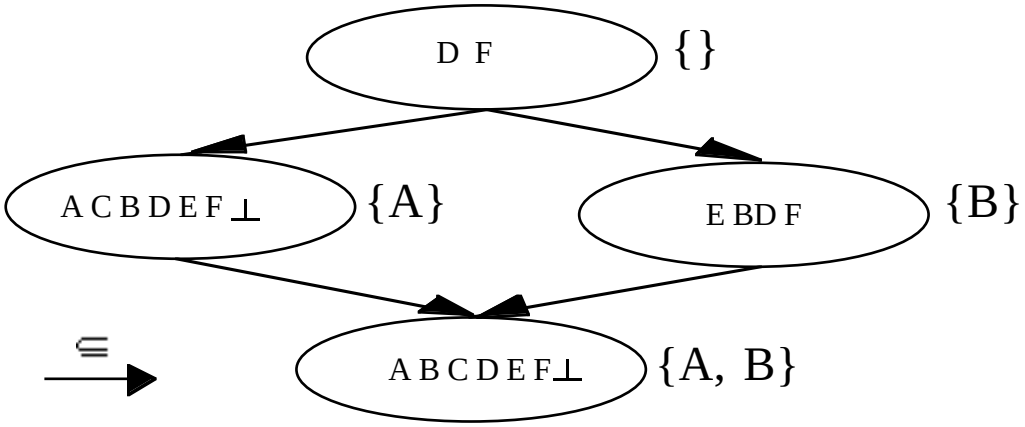


figure 18. Treillis des environnements correspondant à l'examen de deux hypothèses A et B; à côté de chaque environnement (ensemble d'hypothèses), on peut voir le contexte (représenté par un ovale) qui lui correspond.

Pour l'instant, le nœud ⊥ est un nœud comme les autres. Les justifications donnant naissance à ce treillis sont <{}>:D, <{}>:F, <{B}>:E, <{A}>:C, <{C}>: B et <{A,B}>:⊥. Les formules figurent dans les contextes dans lesquels elles doivent être présentes.

Il est donc possible de n'enregistrer dans les étiquettes que les environnements les plus généraux dans lesquels les nœuds sont présents. C'est ainsi qu'est défini le critère de minimalité d'une étiquette. Une étiquette est *minimale* si aucun environnement de l'étiquette n'est inclus dans un autre environnement de l'étiquette. Cela s'exprime par

$$\forall E,E' \in L(N),\ E \subseteq E' \Rightarrow E=E'$$

ou

$$\forall E \in L(N),\ \neg(\exists E' \in L(N);\ E \subset E').$$

Ainsi, un nœud présent universellement n'aura besoin que de l'environnement le plus général dans son étiquette. Cet environnement est le sommet du treillis: l'ensemble vide ({}). Le fait que le graphe des environnements soit un treillis garantit, de plus, que pour chaque étiquette il existe une unique représentation minimale (cela est dû au fait que chaque sous-treillis est un treillis et doit donc être remplacé par son élément maximal).

Les propriétés de correction et de complétude peuvent donc être réécrites en accord avec la propriété de minimalité. Une étiquette est *correcte* si tous les environnements plus spécifiques qu'un environnement de l'étiquette permettent la présence du nœud:

$$\forall E \in L(N),\ \forall E',\ E \subseteq E' \Rightarrow J,E' \vdash N$$

et elle est *complète* si tout environnement permettant la présence du nœud est représenté par un environnement plus général dans l'étiquette de ce nœud:

$$\forall E',\ J,E' \vdash N \Rightarrow \exists E \in L(N);\ E \subseteq E'.$$

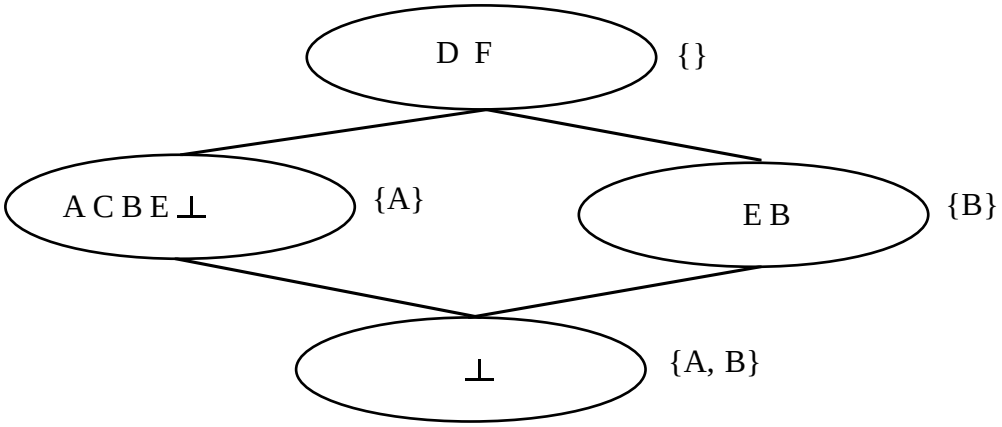


figure 19. Les formules sont maintenant placées dans le contexte le plus général dans lequel elles ont été inférées. Elles sont héritées par les contextes plus spécifiques. Après héritage, l'affectation des nœuds dans les contextes correspondra au graphe de la figure 18.

La dernière propriété des étiquettes est celle de *consistance*. Une étiquette est consistante si aucun des environnements de l'étiquette n'est inconsistant ou encore si aucun des environnements de l'étiquette ne permet la présence du nœud ⊥. Cela s'exprime par

$$\forall E \in L(N), \neg(J, E \vdash \bot).$$

Vis-à-vis du treillis des environnements, l'héritage se fait de la même façon pour les justifications de ⊥ que pour les autres. Si un environnement permet la présence de ⊥ alors tous les environnements qui l'incluent — ceux qui sont plus spécifiques — permettent aussi la présence de ⊥. Un tel environnement est barré dans le treillis (voir figure 20).

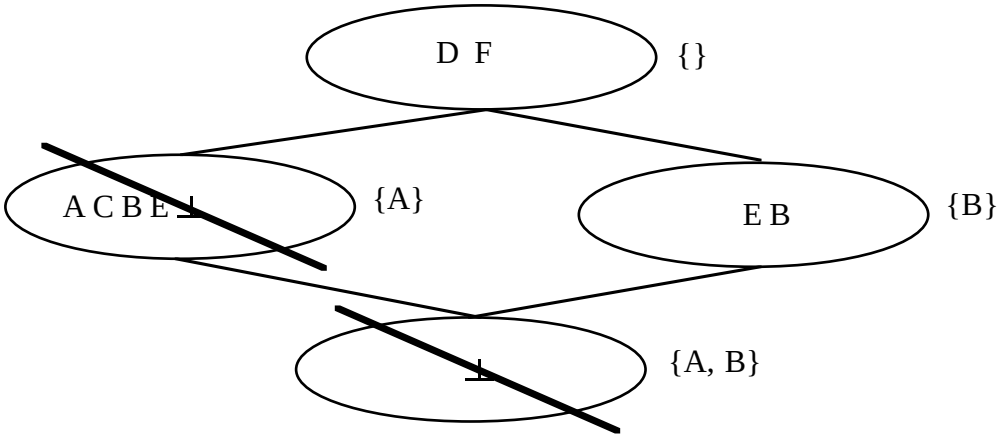


figure 20. Treillis d'environnements avec des environnements inconsistants. Cette fois-ci la théorie est enrichie de la connaissance de l'inconsistance de ⊥. Celui-ci n'est donc plus un nœud comme les autres: la fusion des contextes générés par A et B est alors interdite par l'invalidité de l'environnement {A}.

La représentation par étiquette permet de savoir très facilement, à un moment donné et pour tout contexte, si le nœud est présent dans le contexte ou non. Il suffit de comparer le contexte de requête (ou plutôt l'environnement permettant de caractériser le contexte) et les environnements de l'étiquette du nœud.

L'utilisation des données fournies par le système de maintenance de la vérité à contextes est faite par le système de raisonnement. L'exploitation des données lui permet de passer sans dommage d'un contexte d'utilisation à un autre. Le système de raisonnement — ou l'utilisateur — peut soumettre au système des requêtes sur la validité des formules. Ces requêtes sont de la forme «La formule f est-elle valide dans le contexte C?» où C est un environnement consistant. Le système est capable de classer le nœud associé à la formule dans trois catégories:

a) *nécessairement présent* si un environnement de son étiquette est dans le contexte

$$\exists E \in L(n_f) \; ; \; E \subseteq C.$$

b) *nécessairement absent* si son addition au contexte cause la présence de $\bot$

$$C \cup \{n_f\} \in L(n_\bot).$$

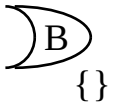
c) *couramment absent* si le nœud peut devenir présent ou absent suivant une nouvelle justification.

À partir des résultats fournis et du contexte dans lequel il se place — ou contexte courant —, le système de raisonnement peut, en connaissance de cause, déterminer quelles inférences déclencher. Il peut aussi chercher à considérer un autre contexte dans lequel les réponses aux requêtes sont plus en accord avec sa vision du monde réel.

L'étiquette représente une forme canonique minimale. Sa signification est que, dans chaque contexte où un de ses environnements est présent, la formule associée au nœud doit aussi être présente. Après avoir défini ces étiquettes, il faut être capable de les combiner et propager d'après les justifications fournies par le système d'inférence.

2.3.Propagation des étiquettes

La création d'un nœud pour une formule f consiste à créer un nœud n_f (ou f) auquel est attaché l'étiquette vide {}, puisque rien n'a permis d'inférer la présence de la formule dans un contexte. La déclaration d'une hypothèse H est tout aussi simple, il suffit de créer le nœud correspondant s'il n'existe déjà, d'ajouter à son étiquette l'environnement {H}, puis de procéder à la propagation d'étiquette comme cela est précisé ci-dessous.



21a. Création d'un nœud.

À la création du nœud B, l'étiquette vide lui est associée.

21b. Création d'une hypothèse.

À la création de l'hypothèse A, l'environnement {A} est ajouté à l'étiquette de A.

À chaque justification, le système associe une étiquette. Pour cela, il calcule les environnements minimaux sous lesquels la justification est valide, c'est-à-dire les ensembles minimaux d'hypothèses permettant l'inférence correspondante. Même si cette dernière a été déclenchée dans un contexte précis, le système de maintenance de la vérité considère sa justification dans tous les environnements sous lesquels elle est valide.

Toute justification $<\{A_1,\ldots A_n\}>$: B, fournie au système, est transposée au niveau des nœuds par l'assertion suivante: «Pour tout contexte C tel que $A_1,\ldots A_n$ soient présents dans le contexte C, alors B est présent dans le contexte C» ce qui sera aussi formulé par

$$\forall C;\ (J,C \vdash n_{A1},\ldots\ J,C \vdash n_{An}),\ J,C \vdash n_B.$$

Les environnements sous lesquels les antécédents sont présents sont bien sûr ceux des étiquettes des antécédents. L'étiquette correspondant à une justification sera donc le *produit*[6] des étiquettes des antécédents de la justification. Ainsi pour la justification $j=<\{A_1,\ldots A_n\}>$: B, l'ensemble L_{CC} (pour étiquette complète et correcte de la justification) correspond à l'ensemble

$$L_{CC}=\{E \mid E=E_1 \cup \ldots E_n\ ;\ E_1 \in L(n_{A1}),\ldots E_n \in L(n_{An})\}.$$

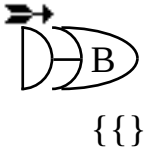
Cet ensemble est complet et correct vis-à-vis de la justification j, car il contient tous les environnements qui permettent la présence de B au vu de j.

6 On nommera «produit d'étiquettes», l'opération qui consiste à faire le produit cartésien des étiquettes, puis à appliquer l'union d'environnements à chaque élément du produit:
{{A,B},{C},{D,E}} -> {A∪C∪D, A∪C∪E, B∪C∪D, B∪C∪E}.
L'union d'environnements représente ici l'union d'ensembles, elle se compliquera plus loin (voir §III.2).

21c. Inférence quand les antécédents sont des hypothèses.

Dans le cas présent (j1=<{C,D}>: E), la nouvelle étiquette est aisée à calculer. Chaque étiquette ne contenant qu'un environnement, les deux environnements sont unis.

L'étiquette correspondant à la justification est donc {{C, D}}. Elle est propagée à son conséquent qui n'a pas d'autres justifications. C et D étant deux hypothèses, la formule E est présente dans la base si et seulement si les nœuds C et D y sont présents.



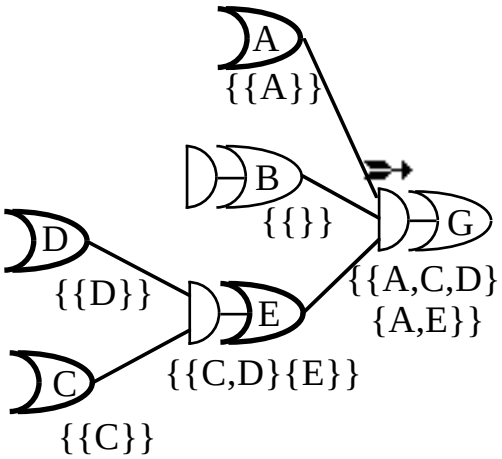
21d. Ajout d'une justification vide.

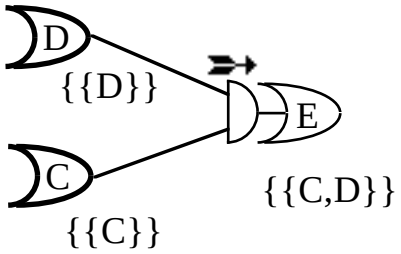
j2=<{}>:B

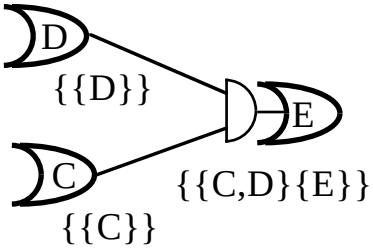
L'environnement calculé est l'élément neutre pour l'opération d'union, à savoir l'ensemble vide. L'étiquette de la justification, et donc du nœud B, contient l'environnement vide. Cette justification dénote une inférence valide inconditionnellement (elle n'a pas d'antécédents), le nœud B est donc présent même si aucune hypothèse n'est posée.

21e. Déclaration d'une hypothèse déjà inférée

La déclaration du nœud E en tant qu'hypothèse va ajouter l'environnement {E} dans son étiquette.







21f. Double propagation.

La propagation n'est pas toujours aussi simple que dans l'exemple 21c. Maintenant que l'étiquette du nœud E contient deux environnements, il faut les propager à chaque fois que E, ou un de ses conséquents, intervient dans une justification.

Ainsi, pour la justification

j3=<{A, B, E}>: G

l'étiquette de G contiendra bien le produit des étiquettes des antécédents.

L'ensemble L_{CC} n'est, par contre, ni minimal, ni consistant. Il est rendu consistant (L_{CCC}) en y supprimant tous les environnements qui contiennent un environnement de l'étiquette de $n_{\perp}$: ainsi

$$L_{CCC} = \{E \mid E \in L_{CC} \;\&\; \forall E' \in L(n_{\perp})\ E' \not\subseteq E\}.$$

L'ensemble des environnements minimaux (L_{CCCM}) d'une justification est obtenu en supprimant tous les environnements de l'ensemble L_{CCC} dont un sous-ensemble est dans L_{CCC}, ainsi

$$L_{CCCM} = \{E \mid E \in L_{CCC} \;\&\; \forall E' \in L_{CCC}\ E' \not\subset E\}.$$

C'est cette dernière étiquette qui constitue l'étiquette d'une justification. Reste à établir l'étiquette du conséquent de la justification.

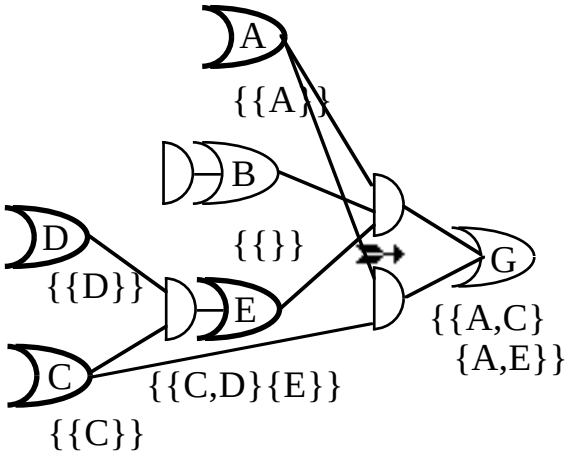
21g. Justifications multiples.

Si un nœud possède plusieurs justifications (ici G reçoit la justification
j4=<{A,C}>:G),
il faut faire l'union des étiquettes correspondant à chacune des justifications. Les étiquettes de j3 et j4 étant respectivement {{A,C,D}{A,E}} et {{A,C}}, leur union donne

{{A,C,D}{A,E}{A,C}}.

Mais cette étiquette n'est plus minimale. La minimalité est rétablie en supprimant les environnements qui sont plus spécifiques que d'autres (ici {A,C,D} est plus spécifique que {A,C}). En effet, si la présence de A et C permet d'inférer celle de G, a fortiori, celle de A, C et D le permet aussi. L'étiquette de G est donc

{{A,E}{A,C}}.



Le travail du système de maintenance de la vérité est incrémental, c'est-à-dire qu'il ne recalcule pas toutes les étiquettes à chaque nouvel ajout de justification. Il est tout d'abord capable de circonscrire l'ensemble des nœuds auxquels cette modification va être propagée (l'ensemble des nœuds conséquents du nœud justifié). Ensuite, il ne va pas recalculer toute l'étiquette des nœuds à mettre à jour, mais simplement la différence entre l'ancienne et la nouvelle étiquette. Enfin, il arrête de propager dès que la nouvelle étiquette qu'il vient d'établir est la même que celle qui la précédait. Ainsi, dès que la propagation a examiné tous les nœuds d'un cycle, l'algorithme s'arrête.

Soit une nouvelle justification pour un nœud B:

$$j=<\{A_1,\ldots A_n\}>:B.$$

La nouvelle étiquette est obtenue à partir de l'ancienne étiquette $L(n_B)$ et de L_{CCC} (l'étiquette correspondant à j) en faisant tout d'abord leur union — opération qui conserve la complétude, la correction et la consistance — puis en rendant l'union minimale de la même façon que pour obtenir L_{CCCM}, en calculant

$$\{E \mid E \in L_{CCC} \cup L(n_B) \;\&\; \forall E' \in L_{CCC} \cup L(n_B)\; E' \not\subset E\}$$

(où $L(n_B)$ est l'ancienne étiquette du nœud).

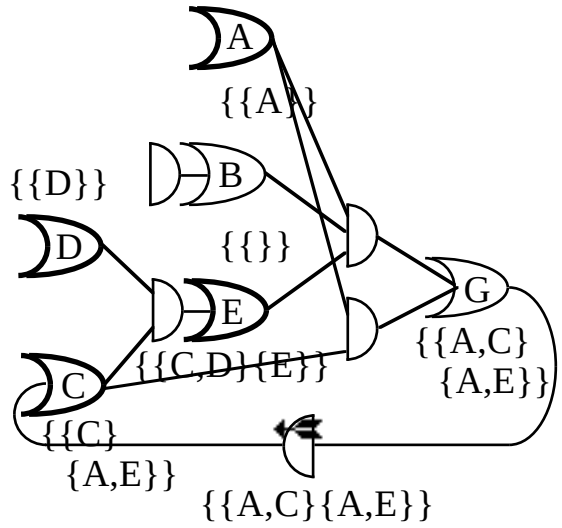
Une fois cette nouvelle étiquette trouvée, il va falloir la propager dans tout le graphe des justifications. En effet, si le nœud inféré (B) a déjà été utilisé dans une autre justification

$$j' = <\{B,A'_1,\ldots A'_m\}>: C,$$

il faut ajouter un certain nombre d'environnements à l'étiquette de C. Cette nouvelle étiquette peut être obtenue en calculant une étiquette partielle et correcte (L_{PC})

$$L_{PC} = \{E \mid E=L_{CCC} \cup E_1 \cup \ldots E_m \; ; \; E_1 \in L(n_{A'1}),\ldots E_n \in L(n_{A'm})\},$$

puis en rendant cette étiquette consistante et en l'associant à l'étiquette de n_C comme précédemment pour n_B. Cette opération se fait pour toutes les justifications dans lesquelles B apparaît et récursivement sur les conséquents des justifications.



21h. Propagation incrémentale (jusqu'à C).

Si une nouvelle justification est ajoutée au graphe de la figure 21g:

$$j5=<\{G\}>:C,$$

l'étiquette de cette nouvelle justification est calculée. C'est {{A,C}{A,E}}. Elle est ajoutée à l'étiquette de C ce qui donne {{A,C}{A,E}{C}}. Après minimisation ({C} est plus général que {A,C}), la nouvelle étiquette de C est {{C}{A,E}}.

21i. Propagation incrémentale (jusqu'à E).

Il faut alors propager cette nouvelle étiquette vers le nœud E via la justification j1. La nouvelle étiquette correspondant à j1 est le produit des étiquettes de C et D. Cela donne

{{C,D}{A,D,E}}.

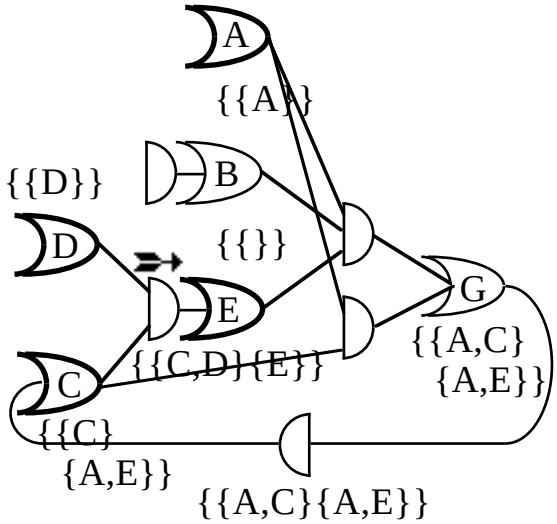
Ajouté à l'étiquette de E, celle-ci devient

{{E}{C,D}{A,D,E}}

ce qui rend après minimisation ({E} est plus général que {A,D,E}), l'étiquette

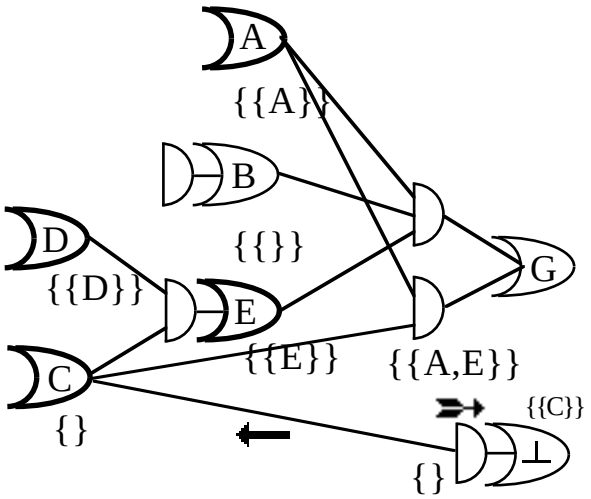
{{E}{C,D}}.

Cette dernière étiquette étant celle de E avant la propagation, celle-ci s'arrête, n'atteignant même pas G.



Une fois terminée la propagation des étiquettes, le travail du système de maintenance de la vérité à contextes n'est pas toujours terminé. En effet, si l'étiquette du nœud ⊥ a été modifiée alors la consistance des étiquettes des autres nœuds n'est plus garantie car la nouvelle étiquette de ⊥ n'a pas encore été prise en compte. Il suffit, pour rétablir la consistance de l'étiquette de chaque nœud N, d'y supprimer les environnements plus spécifiques qu'un des nouveaux environnements de l'étiquette de ⊥ (LPC). La nouvelle étiquette de N sera donc:

$$\{E \mid E \in L(N) \;\&\; \forall E' \in L_{PC}\ E' \not\subseteq E\}.$$



21j. Rétablissement de la consistance.

Si une justification représentant l'instanciation d'une contrainte est ajoutée au graphe de la figure 21g:

j6=<{C}>:⊥,

l'étiquette du nœud ⊥ est calculée. C'est {{C}}. {C} est donc un nouvel environnement inconsistant. Les étiquettes présentes dans tout le graphe ne sont plus consistantes et doivent être purgées des environnements plus spécifiques que {C}. C'est le cas des environnements {C}, {C,D} et {A,C} dans les étiquettes de C, E et G.

Il faut noter que les nœuds à visiter sont facilement circonscrits puisqu'il s'agit de l'intersection des conséquents des nœuds présents dans le nouvel environnement de ⊥.

2.4.Conclusion

Le système de maintenance de la vérité à contextes permet donc de poursuivre efficacement un raisonnement en parallèle dans différents contextes, et de pouvoir passer facilement de l'un à l'autre. Cela est très utile pour circuler dans l'espace des possibles sans perdre tout le bénéfice des inférences déjà produites.

Contrairement au système de Jon Doyle, l'ATMS bénéficie d'une caractérisation claire et facile à implémenter.

Comme le souligne lui-même De Kleer, le système à contextes n'est pas valable pour tous les usages, chaque système a ses qualités propres. Johan De Kleer [De Kleer 86a] considère successivement trois types de problèmes: trouver toutes les solutions parmi de nombreuses solutions, trouver la solution unique, trouver une solution parmi de nombreuses solutions. Il affirme que dans cet ordre les premiers problèmes sont plus efficacement résolus par les systèmes de maintenance de la vérité à contextes et les derniers par les systèmes de maintenance de la vérité à propagation.

Bien que Johan De Kleer signale que son système est capable de résoudre tous les problèmes nécessitant un raisonnement non monotone, l'ATMS est peu apte à la prise en compte des inférences non monotones. Ce système ne propose de supprimer que des faits, alors qu'un système qui gère la non monotonie doit pouvoir aussi renoncer à des inférences.

Par ailleurs, les performances du système ne sont pas sensiblement meilleures que celles du système à propagation après les améliorations apportées par James Goodwin.

Les deux classes de systèmes apparaissent donc toujours comme complémentaires et les efforts pour disposer des avantages des deux systèmes dans un seul se sont développés depuis leurs premières spécifications.

3.Systèmes de maintenance de la vérité enrichis

L'utilisation du système à contextes dans un raisonnement non monotone est, à priori, envisageable. En réalité, l'ajout de justifications non monotones est plutôt contraire à l'esprit du système de maintenance de la vérité à contextes. En effet, l'intérêt de l'ATMS est qu'il exploite complètement la structure de treillis qui lui permet d'affirmer que si la formule α est valide dans l'environnement contenant l'hypothèse H, elle sera valide dans tous les contextes contenant H. C'est précisément la définition de la propriété de monotonie. L'introduction de justifications non monotones interdirait d'utiliser le treillis qui est justement le point fort de l'ATMS (voir figure 22).

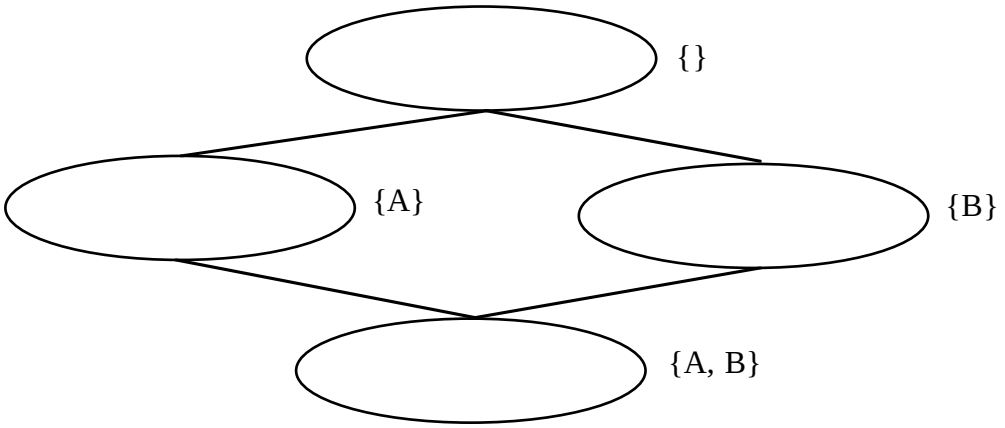


figure 22. Décider de l'environnement dans lequel est valide la justification non monotone <{A}{B}>:D est une gageure. En effet, à première vue, l'environnement {A} est adéquat puisque la formule A y est présente (IN) alors que la formule B en est absente (OUT). Mais si cela est fait, la formule D sera héritée du contexte correspondant à {A} vers celui de {A, B}, et dans celui-ci, B est maintenant IN invalidant l'inférence.

3.1.Extensions de l'ATMS

Plusieurs extensions de l'ATMS vers les inférences non monotones ont été ainsi proposées. Le problème principal est que l'ATMS est fondé sur les propriétés de monotonie de l'inclusion ensembliste. Différents stratagèmes ont donc été imaginés pour tourner cette propriété de monotonie, ils nécessitent la génération de nombreux environnements dont certains seront inconsistants.

3.1.1. 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 ($O \vee N$). Chaque justification non monotone $<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>$: c introduit deux hypothèses:

- O représentant la présence d'un des membres de la OUT-liste,
- N représentant l'absence de tous les membres de la OUT-liste.

Ces deux hypothèses sont, bien sûr, exclusives, ce qui se note par:

$$<\{\}> : O \veebar N$$

$$<\{O, N\}> : \bot.$$

La justification initiale devient alors

$$\forall j;\ 1 \leq j \leq m,\ <\{i_1,\ldots i_n, o_j\}> : O$$

$$<\{i_1,\ldots i_n, N\}> : c.$$

Cette méthode montre bien la difficulté d'introduire les inférences non monotones dans l'ATMS. La figure 23 présente le graphe de dépendances comprenant les inférences non monotones et celui ne comprenant que des inférences monotones.

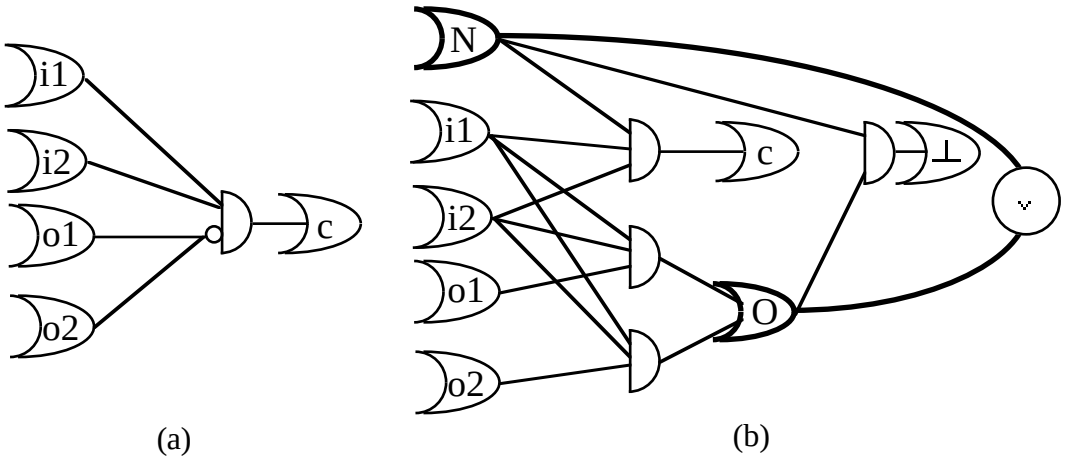


figure 23. En (a) une justification non monotone, codée en (b) pour être acceptée par l'ATMS. Le rond auquel arrivent des traits gras dans la figure (b) exprime une disjonction, inexprimable par des justifications classiques de l'ATMS.

Pour la simple justification de la figure 23 le treillis obtenu est celui de la figure 24. Le problème n'est pas que celui-ci soit important (il faut tout de même ajouter deux hypothèses par inférence non monotone), mais qu'il n'a pas beaucoup de signification par rapport à l'interprétation classique du treillis de l'ATMS.

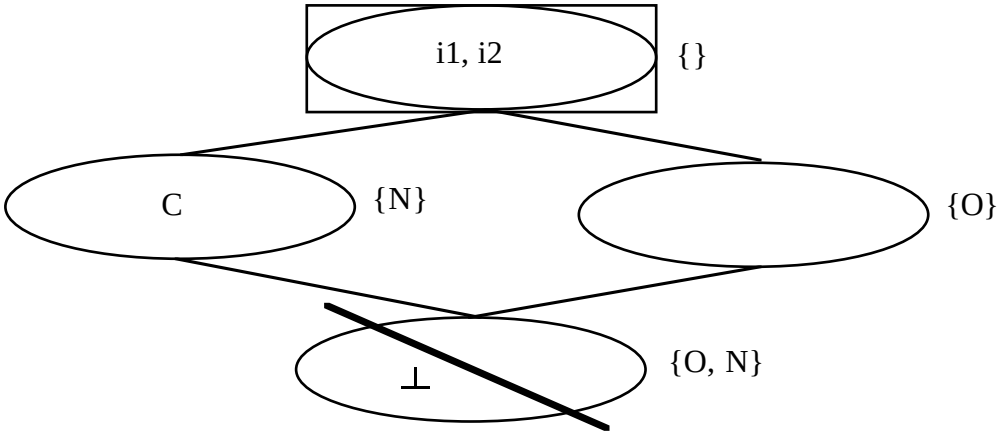


figure 24. En supposant que les antécédents monotones i1 et i2 sont toujours présents, voici le treillis d'environnements engendré par la justification <{i1,i2}{o1,o2}>:c. Le contexte correspondant à l'environnement {} est encadré car il n'est pas un contexte complet: le O∨N spécifie en effet que soit O, soit N doit être présent.

Ce mécanisme en lui-même est déjà lourd à utiliser, mais la difficulté est encore augmentée 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. Cette règle est la suivante:

$$\frac{H_1 \vee \ldots H_n \qquad \forall i \in [1,n],\ \exists E_i \in \text{ETIQUETTE}[\bot];\ H_i \in E_i \wedge \forall j \in [1,n],\ j \neq i,\ H_j \notin E}{\cup_{i \in [1,n]} [E_i \setminus \{H_i\}] \in \text{ETIQUETTE}[\bot]}$$

Elle est utilisée de la façon suivante:

$$\frac{A \vee B \qquad \{A,C\} \in \text{ETIQUETTE}[\bot] \qquad \{B,C\} \in \text{ETIQUETTE}[\bot]}{\{C\} \in \text{ETIQUETTE}[\bot]}$$

ce qui signifie que si ou A ou B doivent être présents et que chacun, conjugué avec C, est inconsistant, alors, C est toujours inconsistant.

3.1.2. Dressler 88

Oskar Dressler propose une autre implémentation [Dressler 87, 88]: pour chaque nœud N, une hypothèse Out(N) peut être créée dont l'étiquette doit capturer l'ensemble des environnements sous lesquels N n'est pas présent. Cette hypothèse ne peut jamais être conséquence d'une justification. La justification

$$<\{\text{Out(N)}, \text{N}\}> : \bot$$

est fournie au système afin que les étiquettes des deux nœuds soient bien disjointes. La complétude est obtenue en traitant l'inconsistance grâce à la règle

$$\frac{E \cup \{Out(N)\} \subseteq ETIQUETTE[\bot]}{E \subseteq ETIQUETTE[N]}$$

qui remplace l'utilisation du CHOOSE. Ainsi, si un nœud Out(N) est inconsistant avec un environnement E, alors N est présent sous cet environnement. L'étiquette de Out(N) est implicite et n'est jamais calculée: le système utilise le complémentaire de l'étiquette de N. Enfin, 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 IN.

Ceci peut permettre de prendre en compte les justifications non monotones. Ainsi, pour la justification $<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>$: c le système l'enregistre comme:

$$\forall j;\ 1 \le j \le m \ <\{o_j\}> : O$$

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

Cette méthode permet de coder une justification non monotone comme indiqué sur la figure 25.

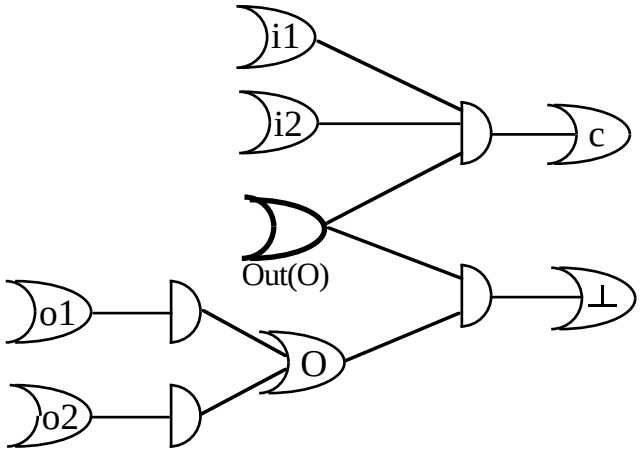


figure 25. La justification non monotone de la figure 23 est codée ici avec la méthode d'Oskar Dressler. Le graphe obtenu est beaucoup plus simple que celui obtenu avec la méthode de Johan De Kleer. Une conséquence importante est que la disjonction (difficile à traiter) a disparu et que tout est exprimé avec des justifications acceptées par l'ATMS de base.

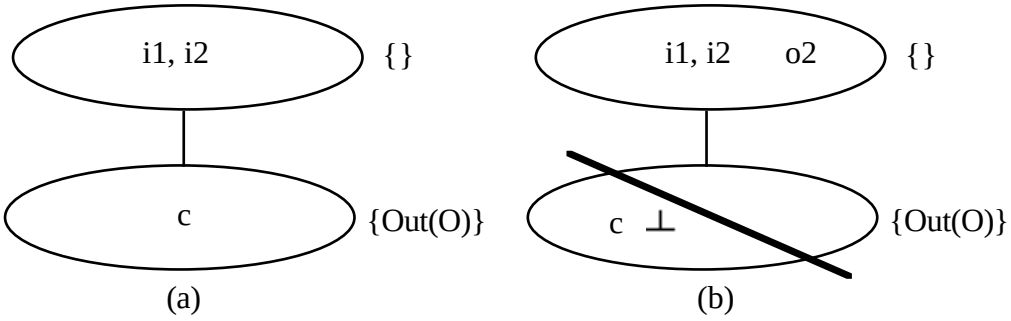


figure 26. Ici, le treillis engendré par le codage n'est pas trop pénalisant, il ne contient qu'une hypothèse supplémentaire par inférence non monotone.

En (a), il est consistant de concevoir la validité de Out(O), c'est-à-dire qu'aucun des antécédents non monotones n'est considéré comme présent. Il existe donc une extension dans laquelle c est présent.

En (b), la présence d'un des antécédents non monotones (o2) dans un contexte plus général ({}) interdit de considérer une telle extension.

Oskar Dressler propose une seconde façon de coder les inférences non monotones en ne créant plus un nœud O pour représenter la OUT-liste, mais en créant les hypothèses $Out(o_i)$ pour chaque o_i dans la OUT-liste. Une justification non monotone sera alors représentée par

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

Cela créera un plus grand nombre d'environnements (voir figure 27), mais permettra de ne pas faire intervenir un nœud O inconnu de l'utilisateur.

3.1.3. 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 d'obtenir 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.

Il permet cependant de coder les inférences non monotones comme cela est suggéré au paragraphe précédent. C'est-à-dire que pour la justification $<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>$: c la justification ajoutée sera:

$$<\{i_1,\ldots i_n,\neg o_1,\ldots \neg o_m\}>: c$$

avec pour chaque o_i:

$$<\{\neg o_i, o_i\}> : \bot.$$

Un tel codage, qu'il soit fait avec cette dernière approche ou avec la méthode d'Oskar Dressler, est cependant coûteux en hypothèses comme le souligne la figure 27.

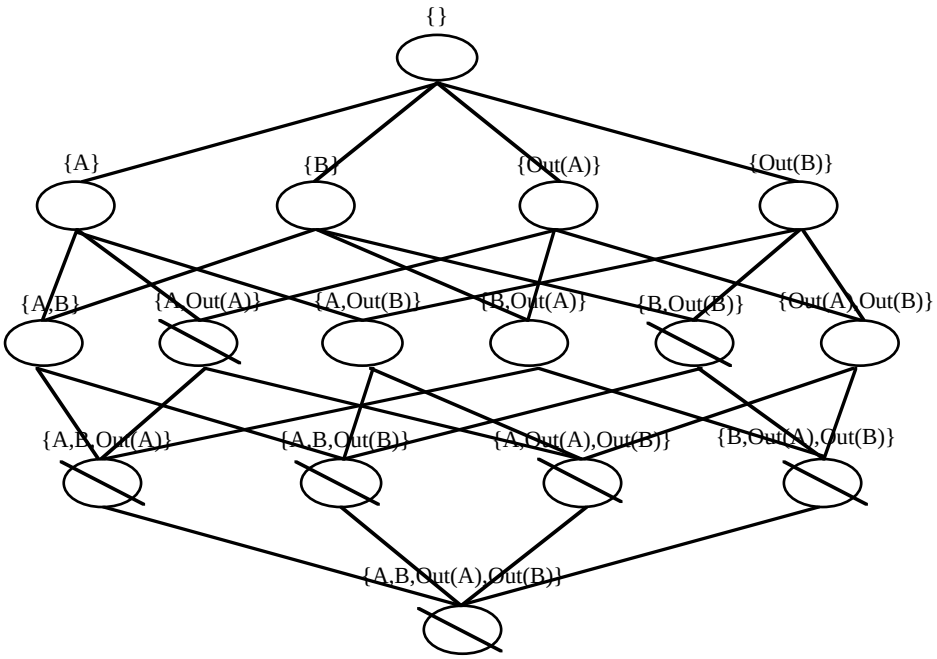


figure 27. L'ensemble des environnements engendrés par l'implémentation des inférences non monotones de Johan De Kleer (Out(N) étant ¬N) ou d'Oskar Dressler (avec les nœuds A et B déclarés comme hypothèses). Il faut noter que pour considérer 2 hypothèses, 16 environnements sont obtenus et que parmi ceux-ci, 7 sont inconsistants.

3.1.4. KEEworlds

Le logiciel KEE implémente la possibilité de masquer un objet dans un monde ("KEEworld" [Morris& 86]). Les mondes servent à représenter l'«état du monde courant» à la suite d'une action ou d'un ensemble d'actions. Ils ne correspondent pas à proprement parler à diverses complétions du monde réel mais à plusieurs mondes possibles. Cependant, si les actions produites sont regardées comme des ajouts de formules, il est alors possible de voir ces différents mondes comme des mondes de plus en plus complets.

Chaque monde est soutenu par une «hypothèse de monde» qui sera désignée par Hw. Un monde w est généré à partir de son monde d'origine W0 à l'aide de la justification suivante:

$$<\{W0,\ Hw\}> : W$$

qui engendre pour W l'étiquette ETIQUETTE[W0]$\cup\{\{Hw\}\}$. Ainsi, l'étiquette de chaque monde est composée des étiquettes des mondes précédents et de l'hypothèse du monde, ce qui constitue l'ensemble des hypothèses des mondes antécédents. L'ajout d'une formule f dans un monde W se fait avec la justification:

$$<\{W,\ DfW\}> : f$$

où DfW est appelée hypothèse de non suppression de f depuis W. Si f doit être supprimée plus tard, dans le monde W', il suffit d'ajouter la justification:

$$<\{W',\ DfW\}> : \bot.$$

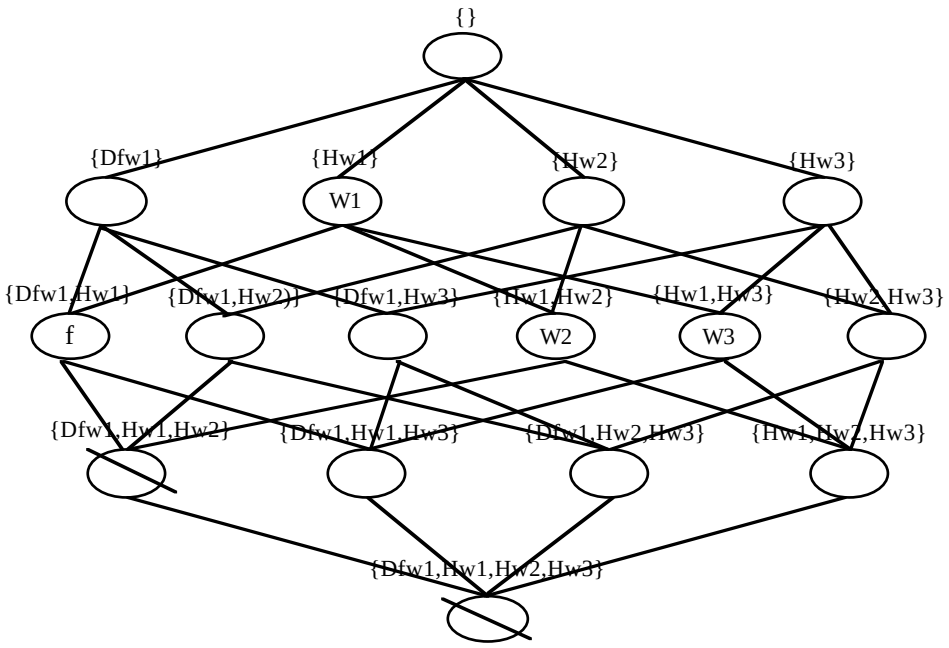


figure 28. Le graphe de l'ATMS construit pour considérer trois mondes, W1, W2 et W3 (les deux derniers étant fils du premier) et l'absence de f dans W2 alors qu'il était présent dans W1. Ce graphe s'exprime plus facilement dans la présentation utilisée par KEE (voir figure 29).

Ainsi, à partir du monde W', DfW sera invalidée et par conséquent f aussi. Ce système a l'avantage de permettre de multiples ajouts et suppressions. Cependant, la manière naturelle à l'aide de l'ATMS de faire des fusions de mondes ne semble pas satisfaire les concepteurs du système (voir figure 28).

Ce système ne permet pas à proprement parler de faire du raisonnement non monotone puisqu'il ne concerne que les actions et que la «non monotonie» est introduite par des actions de retrait. Cependant, il ressemble fortement à la dernière proposition (§3.1.3), les hypothèses DfW correspondant aux hypothèses ¬f. Il génère par ailleurs un grand nombre d'hypothèses qui permettent la gestion des mondes (qui sont différents des environnements de l'ATMS): il y a ainsi beaucoup trop d'environnements masqués à l'utilisateur par rapport aux mondes ayant une pertinence.

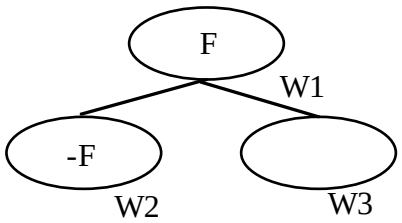


figure 29. Le graphe de la figure précédente tel qu'il est présenté à l'utilisateur de KEE. Il y a bien trois mondes et l'héritage de F dans W2 est annulé (-F).

Le système offert par KEE, même s'il est très coûteux, a l'avantage d'exploiter toutes (et seulement) les possibilités de l'ATMS et par conséquent son comportement répond à des spécifications claires.

Les approches qui ont été présentées ci-dessus tentent d'implémenter le raisonnement non monotone au sein de l'ATMS en le «monotonisant», c'est-à-dire en multipliant les entités manipulées par le système. Elles permettent de considérer les multiples étiquetages autorisés par le TMS en forçant les membres de OUT-listes à être des hypothèses et en interdisant de les inférer. Par ailleurs, certaines des entités en surnombre n'ont pas de correspondant dans le système de raisonnement et ne sont donc pas réellement traitées comme des objets normaux de l'ATMS.

La présence de nombreux nœuds additionnels oblige à supprimer des environnements générés en surnombre à l'aide de déclarations d'inconsistance. L'appel répété à la partie rétablissement de la consistance de l'ATMS (la partie la plus coûteuse) est un lourd handicap pour les systèmes proposés.

Ces approches sont donc assez inélégantes et peu efficaces. Elles modifient peu, par contre, le fonctionnement de l'ATMS et utilisent au contraire toutes ses propriétés.

3.2.Les points de vue d'ART

Le logiciel ART [Williams 84, Clayton 86] 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 vraisemblable que son fonctionnement est différent de celui de l'ATMS malgré une interface semblable.

Comme l'ATMS, ART autorise la création d'hypothèses (ou "viewpoints") et évalue les inférences au sein des contextes 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 contextes manipulés sont de la forme $C|C_1,\ldots C_n$ correspondant à l'ensemble des "viewpoints" plus spécifiques que C sauf ceux plus spécifiques que $C_1,\ldots C_n$.

Les règles sont définies par leur nom (ici `infC`) ainsi qu'une partie droite et gauche séparées par le symbole `=>`. Ainsi, la règle `infC`

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

se déclenche dans tous les contextes où A est présent et B absent, et a pour but d'ajouter C dans les contextes où elle se déclenche. Des instructions spécifiques permettent de manipuler les contextes:

- L'instruction `sprout` crée un nouveau contexte dans lequel vont s'exécuter les instructions qui lui sont passées en argument.
- L'instruction `hypothesize` crée un contexte et y exécute les instructions qui lui sont passées en argument. Ces instructions justifieront le contexte.
- L'instruction `poison` déclare le contexte courant comme inconsistant.

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 qui permet d'obtenir cette étiquette est très complexe, il doit trouver des contextes plus généraux et plus spécifiques à partir de contextes avec restrictions. La structure des contextes s'éloigne donc de celle d'environnement 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 contexte avec plusieurs restrictions. Ceci oblige à dupliquer les faits quand un contexte est éclaté. C'est pénalisant car les contextes sont bien plus souvent éclatés que les étiquettes du simple ATMS. Ce principe revient, à l'instar de MBR [Martins& 83, 88], à ne plus considérer qu'une justification par nœud. Cette duplication pose, de surcroît, le problème de l'égalité de deux faits.

Exemple 1:

Soit les règles suivantes activées l'une après l'autre (la flèche -> indique les contextes créés par les instructions de manipulations de contextes):

```
(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))) -> C1.1.1
```

A et B sont alors respectivement présents dans les contextes C1|C1.2 et C1.1|C1.1.1 (voir figure 30). 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 contextes pour étiquette.

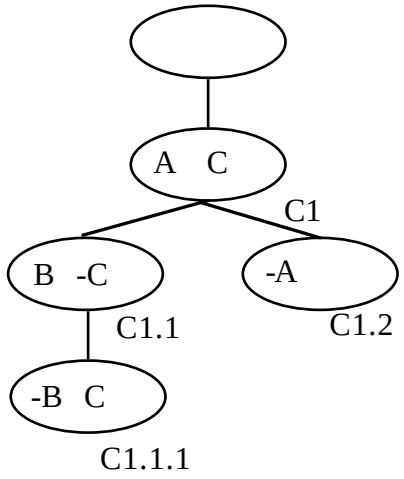


figure 30. Les contextes engendrés par les actions de l'exemple 1. Les formules sont représentées dans les contextes à partir desquels elles sont présentes. Une formule précédée d'un tiret (-) n'est plus héritée à partir du niveau où elle figure.

(2) Le système offre la possibilité de masquer dans un contexte la présence d'une formule qui doit y être présente au vu du graphe des contextes et de celui des dépendances (voir exemple 1). Plus ce genre de possibilité est utilisé, plus le sens d'un contexte diffère du sens intuitif qui pouvait se rencontrer dans les environnements de l'ATMS. Comme dans les systèmes à base de règles classiques, l'utilisation du `retract` n'a 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, en toute rigueur, être un TMS.

(4) L'héritage de l'inconsistance n'obéit pas aux mêmes règles que l'héritage des faits: ART invalide tous les contextes plus spécifiques que ceux dans lesquels l'inconsistance est dérivable, or certains peuvent ne pas être inconsistants (voir exemple 2).

Exemple 2:

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

```
(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.

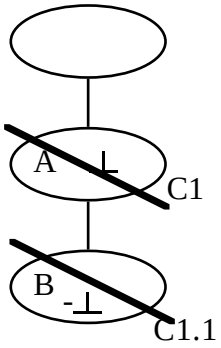


figure 31. Les contextes engendrés par les actions de l'exemple 2. Si ⊥ était un simple fait, il ne serait plus valide dans le contexte C1.1. Celui-ci n'est donc pas inconsistant. Mais la stratégie adoptée supprime tous les contextes engendrés par un contexte inconsistant.

Le problème soulevé ici est que l'instruction `poison` utilisée comme les autres instructions en partie droite des règles 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.

L'exemple de ART montre bien les difficultés qu'il y a à mêler de manière élégante les fonctionnalités des systèmes à contextes et à propagation. 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.

4.Conclusion

Au cours de ce chapitre, les deux principaux systèmes de maintenance de la vérité ont été exposés:

- Le système à propagation permet d'accepter des inférences non monotones et de maintenir un état cohérent du réseau de dépendances.
- Le système à contextes permet de maintenir parallèlement l'ensemble des états cohérents d'un graphe (sans inférences non monotones) engendrés par un ensemble d'hypothèses.

Ces deux systèmes remplissent donc bien des tâches complémentaires mais pas forcément incompatibles. Le besoin d'un système rassemblant les fonctionnalités des deux catégories transparaît dans les tentatives tant académiques que commerciales de construire des systèmes alliant raisonnement multi-contextes et raisonnement non monotone.

Les problèmes liés à ces tentatives sont soit des problèmes de spécifications incomplètes soit des problèmes de performances. En tout état de cause, il faut bien constater que l'ATMS n'est pas fait pour recevoir des inférences non monotones et que vouloir les y introduire revient à multiplier les hypothèses au détriment de l'intelligibilité du travail du système et des performances.

Le but du travail qui va être présenté dans la suite est de concevoir un système spécifiquement adapté aux inférences non monotones. Les trois chapitres suivants vont donc permettre:

1° de spécifier l'action d'un tel système.

2° de décrire une implémentation partielle qui a été faite.

3° de tirer les enseignements de cette expérience tant sur le plan de l'implémentation que sur celui du modèle proposé.

SECOND CHAPITRE

Proposition d'un système de maintenance de la vérité à propagation de contextes

Le souci d'offrir de meilleurs environnements pour le raisonnement hypothétique conduit naturellement à imaginer un système de gestion de la non monotonie complet, à savoir un système capable de supporter conjointement un raisonnement contextuel et non monotone. Ce second chapitre expose la conception d'un tel système en proposant un modèle d'environnement adapté aux inférences non monotones. Ces environnements sont définis et leur sémantique est clairement fixée afin qu'ils puissent être utilisés dans une implémentation.

Second chapitre

Proposition d'un système de maintenance de la vérité à propagation de contextes

Proposition d'un système de maintenance de la vérité à propagation de contextes

1.Objectifs

L'objectif de ce travail est d'autoriser un raisonnement non monotone tel que celui géré par les systèmes de maintenance de la vérité à propagation, qui puisse considérer simultanément plusieurs ensembles d'hypothèses comme axiomes et passer efficacement de l'un à l'autre.

Par ailleurs, le système de maintenance de la vérité à propagation doit faire face à de multiples possibilités d'étiquetage lors de la propagation. Ces possibilités sont dues aux inférences non monotones et ne se présentent pas pour le système de maintenance de la vérité à contextes. Un autre objectif est de pouvoir circuler entre les différents étiquetages qui s'offrent au système de maintenance de la vérité à propagation, comme le système de maintenance de la vérité à contextes le permet entre les différents ensembles d'hypothèses: par un changement de contexte. Ce n'est cependant pas l'objectif immédiat de ce travail.

Le plein développement d'un raisonnement hypothétique nécessite donc la possibilité, à la fois, de poser des hypothèses et de gérer des inférences non monotones. Cela représente la fusion des fonctionnalités dont dispose chacun des systèmes. Pour développer des systèmes réellement capables de cette gestion de la non monotonie, diverses voies semblent possibles:

- Prendre en compte les inférences non monotones dans un ATMS de manière monotone. C'est ce que font la plupart des systèmes présentés au §I.3.
- Linéariser les contextes, ce qui correspondrait à utiliser de concert plusieurs TMS, chacun opérant avec un ensemble d'axiomes différent.
- Ajouter, à l'un des deux systèmes, les fonctionnalités qui manquent. Deux possibilités s'offrent alors:
    - Ajouter des environnements à un système de maintenance de la vérité à propagation.
    - Introduire les inférences non monotones et des environnements appropriés dans un système de maintenance de la vérité à contextes.

  Cette alternative mène au même type de système (voir §IV.1).

A priori, la solution qui va être présentée ici peut être considérée comme un hybride des deux dernières. En effet, elle va emprunter au système à propagation, d'une part, ses inférences non monotones et la façon d'interpréter le graphe qu'il construit et au système à contexte, d'autre part, la notion d'étiquette attachée à chaque nœud dénotant les contextes dans lesquels il doit être présent. L'un n'étant pas compatible avec l'autre, nous allons définir une nouvelle forme d'environnement apte à prendre en compte les inférences non monotones. L'interprétation associée à ces inférences définit celle à associer aux étiquettes et aux environnements.

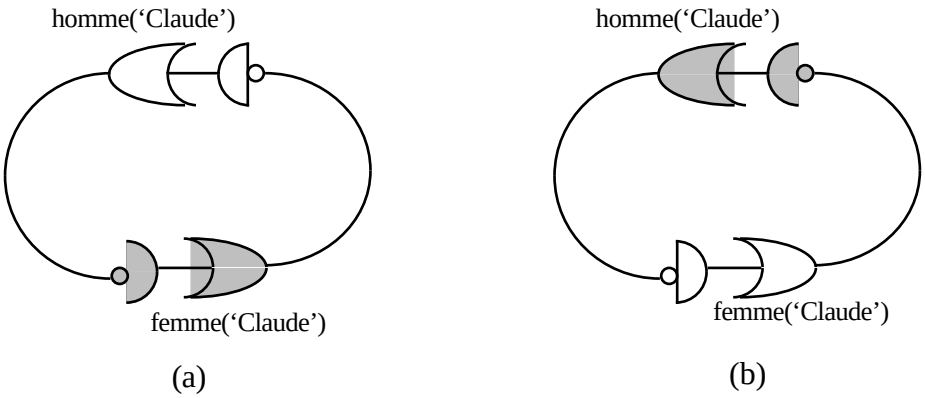


figure 32. Les graphes (repris de la figure 9) correspondent aux inférences si «Claude est une femme» est absent de la base alors poser que «Claude est un homme» et si «Claude est un homme» est absent de la base poser que «Claude est une femme». Deux étiquetages sont admis dans un environnement où aucune hypothèse n'est posée.
• Si la première option est choisie et (a) est l'étiquetage de référence, alors `homme('Claude')` est présent sous cet environnement alors que `femme('Claude')` ne l'est pas.
• Si la seconde option est prise alors ils sont tous deux absents (pour chacun, il existe un étiquetage possible dans lequel il n'est pas présent).
• Si la troisième option est prise, alors ils sont tous deux présents (pour chacun, il existe un étiquetage possible dans lequel il est présent).

L'idée centrale est d'utiliser les inférences non monotones, et donc le réseau de dépendances, et d'en faire l'étiquetage par des environnements. C'est-à-dire que chaque nœud sera étiqueté par les environnements tels qu'un étiquetage absolu (celui du système de maintenance de la vérité à propagation), en accord avec l'environnement, engendrera la présence du nœud. Le principe est donc le même que celui du système à contextes mais les inférences peuvent être non monotones. L'étiquetage absolu du TMS est multiple; il faut donc prendre position vis-à-vis de la signification d'«être présent sous un environnement» (voir figure 32). Trois options sont possibles:

1) Pour chaque environnement, le TMS choisit l'étiquetage absolu correspondant et c'est en accord avec celui-ci que les nœuds sont étiquetés.
2) Il faut que tous les étiquetages possibles à partir de l'environnement entraînent la présence du nœud.
3) Il faut qu'il existe un étiquetage possible à partir d'un des environnements qui entraîne la présence du nœud.

C’est la dernière option qui a été choisie ici: un nœud est présent sous un environnement s’il existe une interprétation de chaque complétion de cet environnement qui value le nœud à 1. Ce choix sera le seul considéré ici et sera discuté lors des critiques de ce modèle (§IV.3.1).

Ce chapitre définit un cadre de développement d’un système prenant en compte le caractère non monotone du raisonnement au sein des environnements. Il s’agit donc d’offrir une définition claire de ce que nous considérons comme un système de maintenance de la vérité permettant un raisonnement à la fois multi-contextes et non monotone. La description du système est faite ici à un niveau purement dénotationnel. Cette caractérisation permettra de développer le CP-TMS dont l’implémentation est détaillée dans le chapitre suivant (§III).

2.Définition formelle des environnements étendus

Dans la suite seront considérés un ensemble $\mathcal{H}$ d'hypothèses et un ensemble $\mathcal{N}$ de nœuds ou formules (avec, bien entendu, $\mathcal{H} \subset \mathcal{N}$). L'ensemble $\mathcal{C}$ des nœuds déclarés comme inconsistants sera réduit, pour des raisons d'exposé, à l'ensemble $\{\bot\}$ (et donc $\bot \in \mathcal{N} \setminus \mathcal{H}$).

Le but fixé semble simple: les inférences non monotones ne sont valides qu'en l'absence de certaines formules. Nous avons défini des environnements tenant compte des formules qui doivent être absentes. Les environnements étant composés d'hypothèses, les nouveaux environnements doivent contenir l'ensemble des hypothèses qui sont présentes dans la base et l'ensemble de celles qui sont absentes.

Un nouveau type d'environnement est donc créé qui sera noté par

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

ou

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

et qui représente les états du monde où les hypothèses $H_1,\ldots H_n$ sont présentes dans la base et les hypothèses $H'_1,\ldots H'_m$ en sont absentes. Le premier ensemble d'hypothèses sera nommé *ensemble d'axiomes* et le second *restriction*. Bien entendu, il faut alors imposer que l'ensemble d'axiomes et la restriction soient disjoints pour que cette définition ait un sens. L'idée n'est apparemment pas nouvelle puisque ces environnements ressemblent à ceux de ART (voir §I.3.2) ou de MBR [Martins& 83, 88].

Un *environnement complet*, qui décrit complètement l'état du monde, est un environnement où la présence (ou l'absence) de toutes les hypothèses de $\mathcal{H}$ est connue. En général, un environnement quelconque n'est pas complet, il ne peut donc représenter complètement l'état du monde. Un environnement représente l'ensemble des environnements complets plus complets que lui.

Ces environnements devront être utilisés dans les étiquettes pour représenter les états du monde dans lesquels les formules associées aux nœuds sont valides. Par conséquent, il faut définir clairement la signification des environnements et des étiquettes des nœuds pour pouvoir leur associer les propriétés de correction, complétude, minimalité et consistance. Ainsi, le travail d'un nouveau système de maintenance de la vérité sera spécifié.

Cette partie va donc tout d'abord définir formellement les nouveaux environnements et la relation qui peut exister entre divers environnements. Une fois ceci fait, l'interprétation des environnements (ne concernant que les hypothèses) sera étendue à l'ensemble des nœuds à l'aide du graphe de dépendances et permettra de définir la signification des étiquettes associées à chaque nœud. Ce travail conduit à donner, par ailleurs, diverses interprétations possibles aux requêtes. L'approche sémantique exposée ici est analogue à la sémantique couramment utilisée en logique.

2.1. Valuations

Un environnement défini comme ci-dessus correspond à une valuation. Une *valuation* est une fonction propositionnelle

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

qui fixe la présence de certaines hypothèses et l'absence d'autres. Un environnement [A/R] d'une étiquette représente une valuation fournie en extension:

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

Il faut bien sûr que A et R soient disjoints ($A \cap R = \{\}$) afin d'éviter d'avoir des valuations qui ne sont plus des fonctions. En raison de la correspondance sans ambiguïté entre une valuation et un environnement, dans la suite, un environnement pourra être remplacé par la valuation qu'il représente, et réciproquement.

Une valuation est dite *complète* si elle est définie sur l'ensemble de toutes les hypothèses ($H=\mathcal{H}$). Sur ces valuations est définie une *relation de complétion*. Une valuation ν_1 est une *complétion* d'une valuation ν_2 (ou ν_1 est *plus complète* que ν_2) si et seulement si

- ν_1 est plus définie que ν_2 ($H_2 \subseteq H_1$),
- sur le domaine de ν_2 (H_2), la valuation attribuée par ν_1 est la même que celle de ν_2 ($\forall h \in H_2, \nu_1(h)=\nu_2(h)$).

L'ensemble des *complétions* de ν est l'ensemble des environnements dont la valuation correspondante est plus complète que celle correspondant à ν. Ceci est exprimé de manière plus lisible à l'aide des environnements nouvellement définis, puisqu'un environnement [A"/R"] est plus complet qu'un environnement [A/R] si et seulement si A est inclus dans A" et R est inclus dans R". Cela permet de définir ainsi l'ensemble des complétions, ou environnements plus spécifiques, d'un environnement:

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

Théorème 1: Comp([A/R]) = {[A∪A'/R∪R'];A'∪R'⊆ℋ\(A∪R) & A'∩R'={} }

Un environnement E_1 plus complet qu'un environnement E_2 est aussi dit plus spécifique. E_2 est alors dit plus général que E_1. Cette relation de complétion, sur l'ensemble des environnements définissables à partir d'un ensemble d'hypothèses est une relation d'ordre partiel. Les propriétés de réflexivité, anti-symétrie, et transitivité sont héritées de la relation d'inclusion (⊆).

Chaque couple d'environnements ([A/R], [A'/R']) possède un environnement plus général minimal ([A∩A'/R∩R']). Toutefois, il ne possède pas forcément d'environnement plus spécifique maximal, [A∪A'/R∪R'] ne remplissant pas toujours la condition (A∪A')∩(R∪R')={}. C'est par exemple le cas (voir figure 33) pour les environnements [B|A] et [A,B|] qui ont comme élément plus général minimal [B|], mais dont [A,B|A] n'est pas un environnement acceptable. Le graphe des environnements structuré par la relation de complétion n'est donc pas un treillis complet.

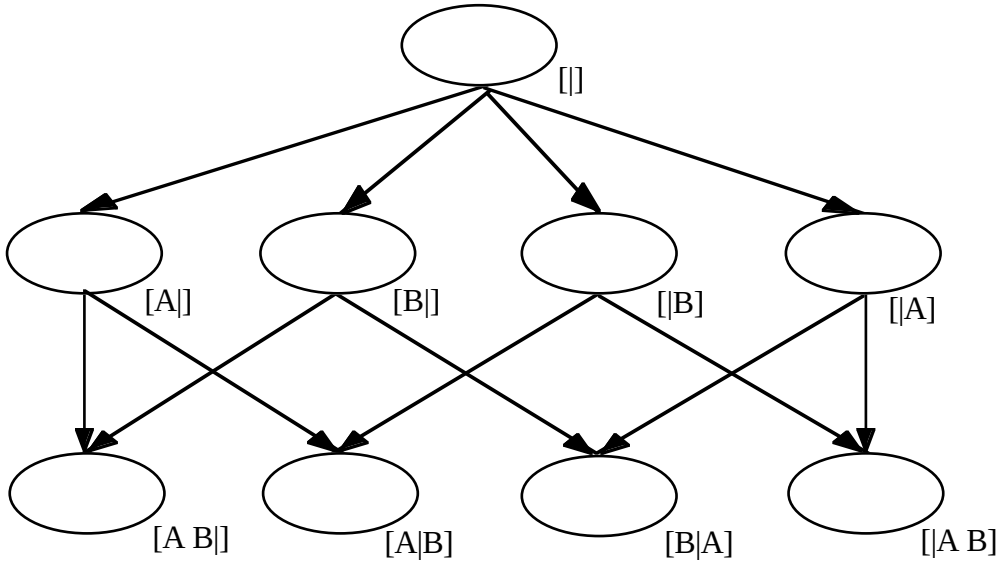


figure 33. Cette figure présente le graphe de la relation «est plus complet que» avec seulement deux hypothèses possibles: A et B. Il y a donc trois niveaux de complétude suivant les statuts accordés aux deux hypothèses. Le dernier niveau (où les deux hypothèses sont présentes dans les environnements) constitue l'ensemble des environnements complets.

D'ores et déjà, il faut constater que ce graphe de complétion semble avoir les bonnes propriétés qui manquaient au graphe des environnements de l'ATMS. En effet, si une inférence permet de dériver une formule F de l'absence de B, celle-ci pourra avoir dans son étiquette l'environnement [|B]. Mais surtout, la formule F pourra être héritée de [|B] vers ses complétions: en effet, toutes conservent l'absence de B. À [A|B] est ajouté que A est présent et à [|A B] que A est absent.

Cet exemple sera justifié quand la notion d'étiquette aura été définie.

L'ensemble des *complétions maximales* de ◇ est l'ensemble des complétions de ◇ définies sur l'ensemble de toutes les hypothèses (H=ℋ). Au niveau d'un environnement [A/R], il constitue l'ensemble des environnements complets qui sont complétions de [A/R]:

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

Ainsi, sur le graphe de la figure 33, l'ensemble des complétions maximales de [|] est {[A B|], [A|B], [B|A], [|A B]} et celui de [|B] est {[A|B], [|B A]}.

Théorème 2:

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

Les théorèmes présentés ici sont destinés à mettre en évidence que ce qui est vrai des complétions maximales d'une valuation l'est de toutes les complétions de celle-ci: les complétions maximales sont les représentations possibles du monde réel et les valuations non complètes ne font que représenter leurs complétions maximales.

2.2. Interprétations

Les environnements représentant des valuations ont été définis. Ces valuations attribuent un étiquetage ({0,1}) aux seules hypothèses. Il est donc nécessaire, pour construire un système de maintenance de la vérité, de définir l'influence de ces valuations sur les simples nœuds du graphe. La notion d'interprétation, analogue à celle présente dans les logiques classiques, va permettre d'étendre une valuation à l'ensemble des formules manipulables par le système. L'extension se fait par le biais du graphe de dépendances qui permet de propager la présence ou l'absence des nœuds à partir des éléments de l'environnement.

Une *interprétation* à partir de la valuation ◇ est une fonction propositionnelle

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

construite à l'aide des règles suivantes:

$\forall N \in \mathcal{N}$,

si $N \in H$ alors $I(N) = \diamond(N)$

sinon, si $\exists J \in$ LISTEDEJUST[N];

$\forall N' \in$ IN-liste[J] $I(N')=1$ & $\forall N' \in$ OUT-liste[J] $I(N')=0$

alors $I(N)=1$

sinon $I(N)=0$.

Cette définition correspond bien à celle qui spécifie l'étiquetage du graphe de dépendances en présence d'inférences non monotones:

- La première condition exprime la conformité de l'étiquetage à celui de l'environnement.
- La seconde condition définit la présence d'un nœud en fonction de la validité d'une de ses justifications.
- La dernière partie «sinon» permet de définir un étiquetage fondé: un nœud ne peut être étiqueté 1 (considéré comme présent) sans aucune raison.

L'exigence de conformité à la valuation (si N∈H alors I(N)=ν(N)) impose de ne pouvoir considérer comme une interprétation à partir de la valuation un étiquetage qui justifierait un nœud de la restriction. C'est-à-dire que si le graphe contient la justification <{A}{}>:B, aucune interprétation de l'environnement [A|B] ne sera admise car le seul étiquetage qui pourrait correspondre à cet environnement impose l'absence et la présence de B. Cet étiquetage se retrouve, par contre, en tant qu'interprétation de l'environnement [A|].

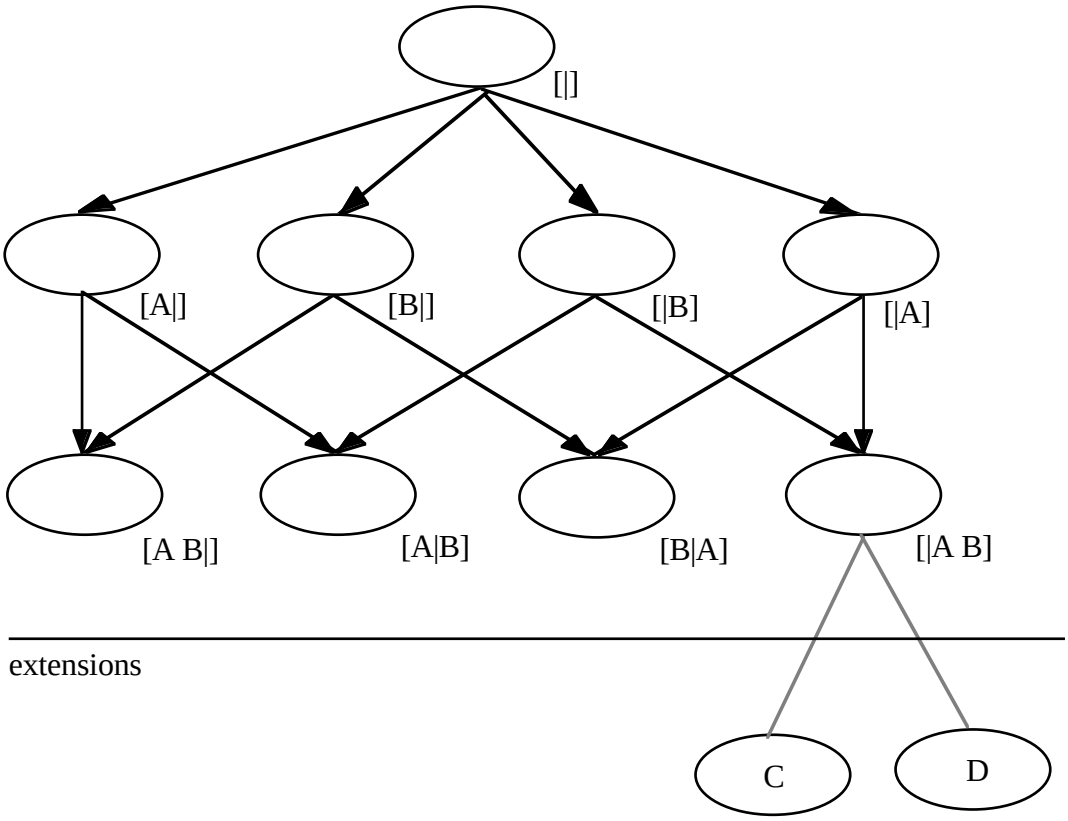


figure 34. Un exemple classique est celui où l'ensemble des justifications correspond à l'ensemble {<{}{A,B,C}>:D, <{}{A,B,D}>:C}. Sous l'environnement [|A B], il existe deux interprétations possibles: {C} et {D}: l'une où A, B et D sont absents de la base et C est présent, l'autre où A, B et C sont absents et D présent.

À une valuation peuvent correspondre plusieurs interprétations, comme le montre la définition récursive des interprétations qui peut être transformée en équation de point fixe. Ces interprétations correspondent aux différents étiquetages possibles pour le graphe du TMS. L'ensemble des interprétations construites à partir de la valuation [A/R] sera noté IntInd([A/R]). Ces interprétations multiples seront aussi appelées *extensions* par analogie avec les logiques de défaut. Elles sont représentées sur le graphe de la figure 34 au dessous de la ligne horizontale.

Pour pouvoir accéder aux extensions, il est possible de redescendre la ligne noire de la figure 34 en dessous de celles qui sont représentées ici en les faisant devenir complétions maximales, c'est-à-dire en posant C ou D comme hypothèses. Ceci a cependant le désavantage d'obliger à manipuler individuellement des hypothèses qui normalement devraient être posées par le système.

L'intérêt des environnements est de représenter les interprétations sous une forme abrégée. Cela va permettre de définir la signification des étiquettes qui sont des ensembles d'environnements.

2.3.Les étiquettes

De nouveaux environnements ont été définis. En accord avec le graphe de dépendances, ils s'étendent en des interprétations de l'ensemble des nœuds présents dans ce graphe. Il est temps de définir clairement la façon d'étiqueter les nœuds de ce graphe avec ces environnements.

Une *étiquette* d'un nœud N représente un ensemble de valuations ω dont toutes les complétions permettent de construire une interprétation I considérant N comme présent. L'étiquette est donc construite sur la définition suivante:

$$\text{ETIQUETTE}[N] = \{C\ ;\ \forall \omega \in \text{Comp}(C), \exists I \in \text{IntInd}(\omega);\ I(N)=1\ \}.$$

Cela est bien en accord avec le choix qui a été fait au début du chapitre: considérer qu'un nœud sera présent sous un environnement s'il existe une interprétation de chaque complétion de cet environnement qui value le nœud à 1.

En fait, la définition utilisée permet d'avoir une définition «minimale» de l'étiquette:

$$C \in \text{ETIQUETTE}[N] \Leftrightarrow \forall \omega \in \text{Comp}(C), \exists I \in \text{IntInd}(\omega);\ I(N)=1$$

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

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

La *correction*, quant à elle, s'exprime par

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

Ce choix dans la définition des étiquettes n'est pas sans conséquence. En reprenant l'exemple donné à la figure 34, on remarque que les nœuds C et D sont tous deux étiquetés par l'environnement [|A B]. Une mauvaise compréhension de ceci pourrait conduire à supposer qu'il existe une interprétation induite par [|A B] dans laquelle C et D sont simultanément présents, or ce n'est pas le cas.

Le problème principal posé par la définition des étiquettes est donc le problème de l'adressage individuel des extensions. Il n'est pas possible de savoir dans quelles extensions précisément un nœud N est présent. Ce problème peut être résolu en faisant descendre la ligne des extensions au dessous de celles qui existent (comme cela est indiqué au paragraphe précédent). Le graphe obtenu est plus grand et à chaque environnement complet correspond une unique interprétation. Il est alors possible de savoir exactement ce qui est présent et absent dans chaque interprétation.

Mais cette solution théorique ne semble pas très pratique, elle conduit à déclarer trop d'hypothèses et peut même exiger la génération d'une infinité d'hypothèses. Par ailleurs, elle peut amener à considérer des hypothèses dont l'utilisateur n'a que faire.

2.4. Consistance

La tâche du système de maintenance de la vérité n'est pas uniquement de trouver un étiquetage faiblement fondé du graphe. Elle consiste aussi à trouver les étiquetages consistants de celui-ci. Une définition des environnements consistants permettra de déterminer ceux qui peuvent apparaître dans les étiquettes des nœuds.

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

Cette définition correspond réellement à un étiquetage inconsistant que le TMS doit éviter par la rétrogression.

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

Une autre définition aurait pu être adoptée en considérant une valuation comme consistante s'il existe une de ses valuations qui soit soutenable. Il faut cependant remarquer que le choix est tout à fait en accord avec celui des étiquettes: une valuation inconsistante a pour complétion une valuation de l'étiquette de $\bot$.

Les étiquettes ne doivent contenir que des environnements représentant des valuations consistantes. Ainsi, tous les environnements plus spécifiques qu'un environnement inconsistant devront, bien sûr, être éliminés des étiquettes. Cette mesure est similaire à celle utilisée pour purger les étiquettes dans l'ATMS. Elle peut être définie de la façon suivante:

$$\forall N \in \mathcal{N}, \omega \in \text{ETIQUETTE}[N] \Rightarrow \forall I \in \text{IntInd}(\omega);\ I(\bot)=0$$

La figure 35 montre les valuations de l'étiquette du nœud $\bot$ en utilisant la contrainte <{B}{}>:$\bot$. Toutes leurs complétions sont inconsistantes.

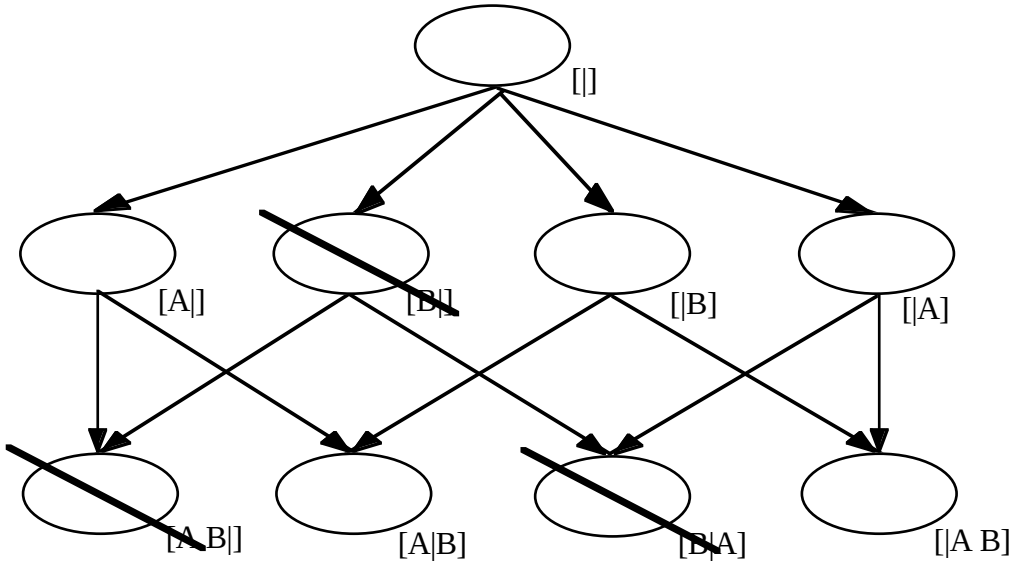


figure 35. Cette figure présente le graphe de la figure 33 où un environnement est inconsistant (celui où B est présent). Trois des environnements représentés sont donc inconsistants (ceux impliquant la présence de B) au sens où toutes leurs complétions maximales sont inconsistantes.

Par mesure d'économie dans l'expression des étiquettes, l'attitude adoptée est de considérer que toutes les complétions d'un environnement présent dans une étiquette doivent être consistantes. Du fait de la monotonie du raisonnement, cela était forcément vrai dans l'ATMS et cette attitude était suffisante. Mais dans le cadre présenté ici, il faut la remplacer par une autre définition. Un environnement peut donc faire partie d'une étiquette seulement si toutes ses complétions sont consistantes. Cela peut s'exprimer par la contrainte suivante:

$$\forall N \in \mathcal{N}, C \in \text{ETIQUETTE}[N] \Rightarrow \forall \omega \in \text{Comp}(C), \forall I \in \text{IntInd}(\omega);\ I(\bot)=0$$

La figure 36 montre, entourées de pointillés, les seuls environnements qui peuvent apparaître au sein des étiquettes en appliquant cette nouvelle définition.

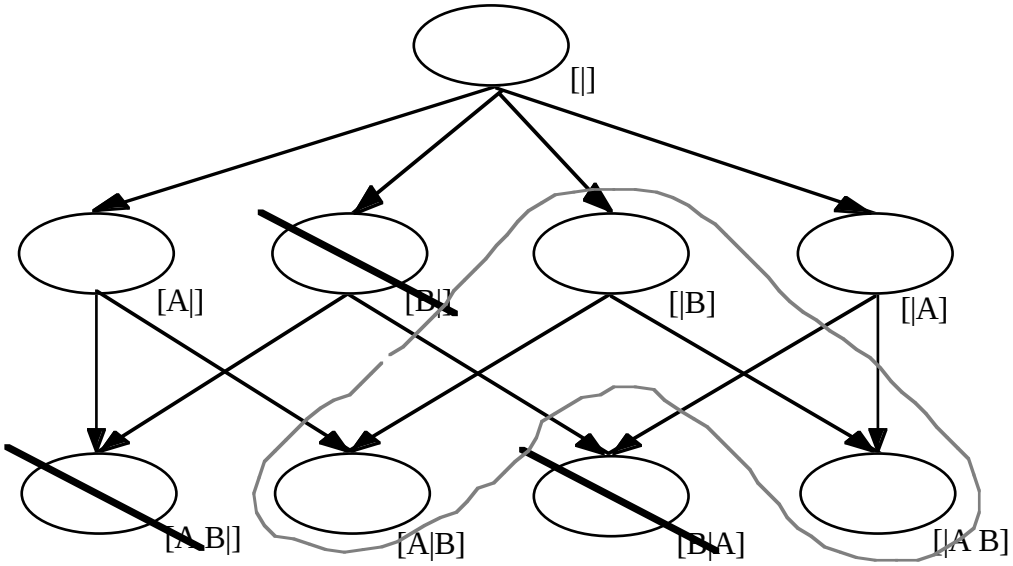


figure 36. Cette figure présente le graphe de la figure 35 où un environnement est inconsistant (celui où B est présent). Trois des environnements représentés sont donc inconsistants et seuls trois environnements peuvent apparaître au sein des étiquettes: [|B], [A|B] et [|B,A] (ceux dont aucune complétion n'est inconsistante).

Bien sûr, il ne faut pas perdre l'information des autres complétions qui sont consistantes et correctes. À cette fin, la notion de complétion consistante est définie.

Une *complétion consistante* ν' d'une valuation ν est une complétion de ν telle que toute interprétation à partir de ν' est soutenable. L'ensemble des complétions consistantes de ν est caractérisé par

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

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

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

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

Ceci est assuré par le fait que ν est l'environnement plus général minimal de ses complétions.

Une façon de caractériser l'étiquette consistante d'un nœud N est de dire que chaque valuation qui permet la présence du nœud N y est représentée par l'ensemble de ses complétions consistantes minimales. La figure 37 montre ce que deviennent ces valuations.

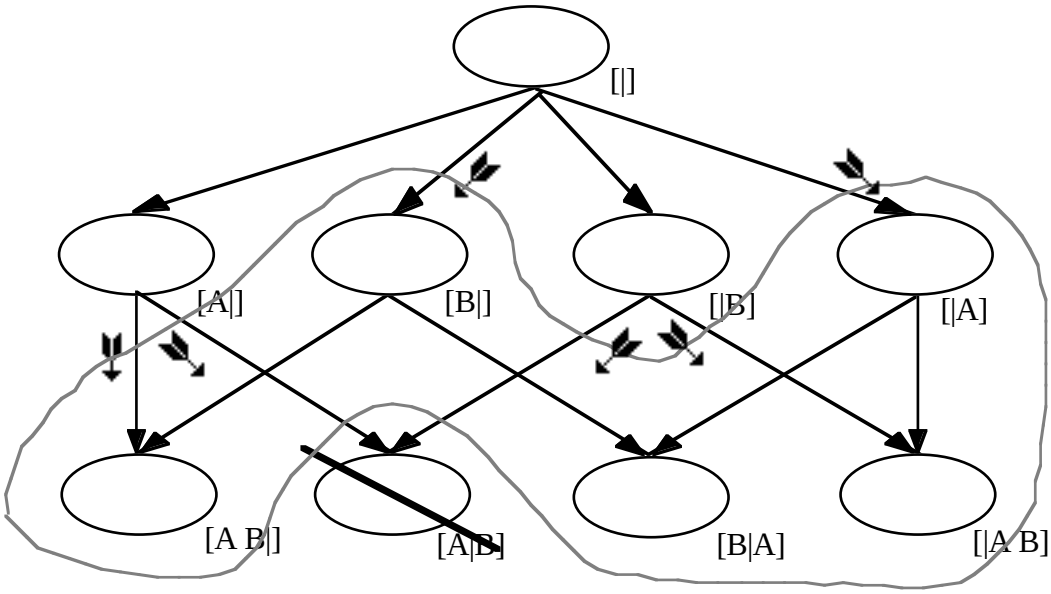


figure 37. Cette figure présente le graphe de la figure 33 où l'environnement inconsistant est celui où B est absent et A présent ([A|B]). C'est donc le seul environnement inconsistant alors qu'il existe cinq environnements qui peuvent apparaître au sein des étiquettes ([B|], [|A,B], [B|A], [|A] et [B,A|]).

Au sein des étiquettes, les environnements interdits ([A|], [|B] et [|]) sont alors remplacés respectivement par {[A B|]}, {[|A B]} et {[B|], [|A]}.

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

Le principe est donc que, dans une étiquette, tout environnement représentant une valuation ayant une complétion inconsistante est remplacé par les environnements représentant ses complétions consistantes minimales. Cette option est maximaliste; bien d'autres peuvent être adoptées, plus laxistes, mais ne mettant pas plus en danger la consistance du système [Euzenat 89a]. L'approche retenue s'éloigne de celle considérée par l'ATMS qui permet de considérer un environnement dont une complétion est inconsistante puisqu'il ne supprime que ceux dont *toutes* les complétions le sont. Cette approche rappelle plutôt celle de MBR [Martins& 83, 88] qui restreint les environnements à ne pas représenter d'environnements inconsistants à l'aide d'une liste de restrictions (voir §IV.1.2).

La figure 38 montre une classification des types de valuations. Elle permet d'observer les relations entre les différents ensembles définis lors de cette étude.

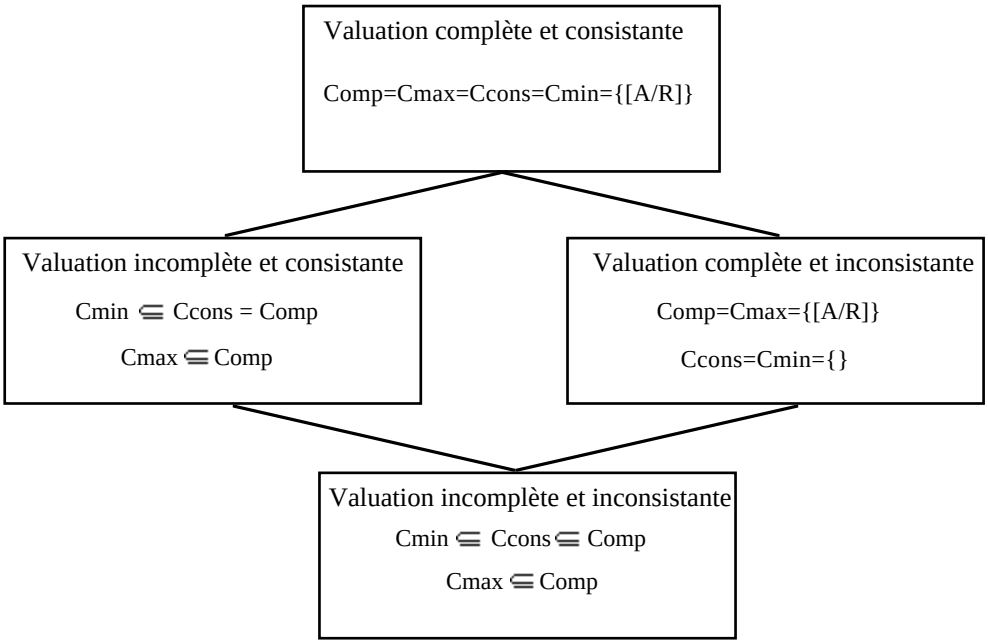


figure 38. Suivant la nature de la valuation [A/R] considérée, les rapports entre ses différents ensembles caractéristiques sont modifiés. Si la valuation considérée est complète, alors les complétions et les complétions maximales sont réduites à cette valuation. Par ailleurs, les complétions consistantes et les complétions consistantes minimales seront identiques. Si la valuation considérée est consistante, alors les complétions et les complétions consistantes sont identiques.

2.5.Les requêtes

La requête constitue le moyen d'interagir avec le système de maintenance de la vérité, que cette requête vienne de l'extérieur ou du système d'inférence (ce peut être, par exemple, l'interrogation du contenu de la base de faits). La requête est le reflet pour l'extérieur du travail du système de maintenance de la vérité. C'est elle qui rend ce travail utilisable.

Une *requête* est une formule associée à un environnement signifiant «la formule F est-elle présente dans le contexte [A/R]?» (ce qui sera noté [A/R]?F). À une telle question il doit être répondu par oui ou non. Avant de pouvoir répondre, il faut savoir ce que signifient les étiquettes des nœuds, les contextes et les requêtes.

2.5.1. Le contexte d'interrogation

Le contexte d'interrogation est, comme dans le cas des systèmes à contextes, un ensemble de nœuds dont la présence est garantie sous un environnement précis. Comme pour les systèmes à contextes, il est communiqué lors d'une requête par un environnement significatif [A/R]. Il n'y a pas, a priori, de raison pour que cet environnement détermine une valuation complète. Le problème est de savoir à quelle interprétation renvoie ce contexte d'interrogation, car il détermine plusieurs interprétations. Indépendamment de l'interprétation choisie, il est possible de mettre en évidence différents ensembles de formules à considérer comme présentes correspondant à un contexte.

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

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

Cette définition rejoint celle de Johan De Kleer [De Kleer 86a].

L'*ensemble des formules nécessaires* dans ν est l'ensemble des nœuds qui doivent être présents dans toutes les interprétations construites à partir de ν:

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

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

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

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

Il ressort de ce théorème que l'ensemble des formules nécessaires dans un environnement correspond à l'ensemble des formules présentes dans les contextes de tous les environnements complets qui sont plus complets que lui. Le théorème 5 permet d'assurer une certaine complétude — en fait la correction — entre le concept de nécessité et l'interprétation des environnements telle qu'elle a été décrite plus haut.

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

L' *ensemble des formules possibles* dans ν est l'ensemble des nœuds qui sont présents dans une interprétation construite à partir de ν:

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

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

Comme pour les formules nécessaires, ce théorème permet de faire correspondre la notion de possibilité à l'intuition vis-à-vis des contextes associés aux environnements complets:

Théorème 7: $\forall n \in \mathbb{N}$, ($n \in Pos(\omega) \Leftrightarrow \exists \omega' \in Cmax(\omega); \exists I \in IntInd(\omega'), I(n)=1$)

Il assure, ici aussi, la même complétude entre interprétations et environnements que le théorème 5.

Le théorème 8 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: $Ctxt(\omega) \subseteq Nec(\omega) \subseteq Pos(\omega)$

Il est possible que certaines complétions de l'environnement [A/R] soient inconsistantes. Il est alors commun de se rabattre sur les complétions consistantes d'un tel environnement, c'est-à-dire les complétions consistantes minimales de [A/R] dont toutes les complétions sont consistantes — c'est-à-dire encore les complétions consistantes minimales de la valuation correspondant à [A/R]. Toute la construction précédente est alors à refaire avec les complétions minimales consistantes.

2.5.2. Signification possible des requêtes

Le problème qui se pose est de savoir dans lesquels des ensembles définis ci-dessus l'utilisateur désire voir présente la formule F de la requête.

Il est clair que les requêtes sont formulées 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 présente dans toutes les interprétations induites par toutes les complétions du contexte d'interrogation[7].
2- La formule doit être présente dans au moins une interprétation induite par chaque complétion du contexte d'interrogation.
3- La formule doit être présente dans toutes les interprétations induites par au moins une complétion du contexte d'interrogation.
4- La formule doit être présente dans au moins une interprétation d'une complétion quelconque du contexte d'interrogation.

[7] Par «complétion du contexte d'interrogation», on entend «complétion de l'environnement représentant le 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 présente dans toutes les complétions consistantes minimales du contexte d'interrogation.

   2- La formule doit être présente dans une complétion consistante minimale du contexte d'interrogation.

   Ici encore, les quatre premiers choix sont possibles pour déterminer ce que signifie «être présente dans une complétion consistante» qui, d'une part, est un contexte d'interrogation comme un autre, mais, d'autre part, est forcément consistant.

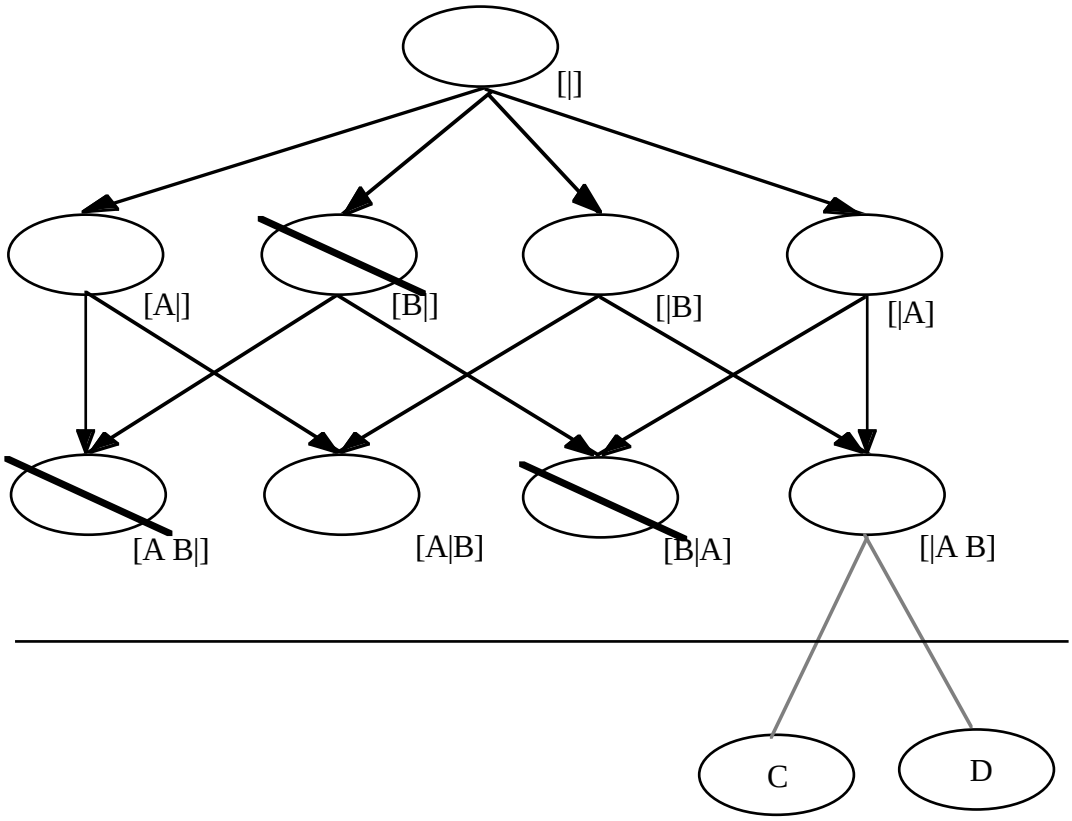


figure 39. Que peut bien signifier, sur ce graphe, la question «C est-il présent sous [|A]?». Ou «C est-il présent sous [|B]?»

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 présente dans toutes les complétions de la valuation correspondant à [A/R]?

b) F est-elle présente dans toutes les complétions consistantes minimales de la valuation correspondant à [A/R]?

c) F est-elle présente dans certaines complétions de toutes les complétions consistantes minimales de la valuation correspondant à [A/R]?

d) F est-elle présente dans une complétion consistante minimale (au sens de toutes les complétions d'une complétion consistante minimale) de la valuation correspondant à [A/R]?

e) F est-elle présente dans certaines complétions d'une complétion consistante minimale de la valuation correspondant à [A/R]?

f) F est-elle présente dans une complétion de la valuation correspondant à [A/R]?

Ici, en accord avec la définition des étiquettes (qui sont les seuls éléments disponibles pour répondre à la requête), «présente dans une complétion» signifie qu'il existe une interprétation de cette complétion pour laquelle F est présente. En ne se restreignant plus à la signification des étiquettes imposée ici, on dispose de deux fois plus de façons d'interpréter les requêtes.

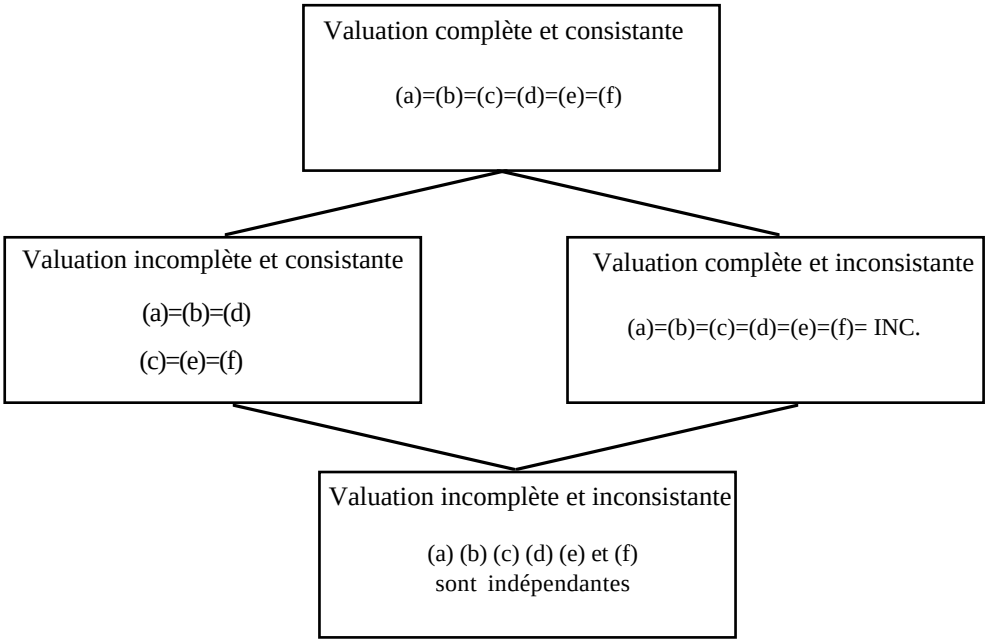


figure 40. Lien entre les diverses interprétations d'une requête suivant le contexte d'interrogation. Il est clair que la réponse positive en (a), entraîne la réponse positive dans toutes les autres questions ($F \in Ctxt([A/R])$). De même (b) entraîne (c), (d), (e) et (f). (c) entraîne (d), (e) et (f). (d) entraîne (e) et (f). Et, enfin, (e) entraîne (f). Il est clair par ailleurs, que si [A/R] est consistant, il n'a qu'une complétion consistante minimale: lui-même et par conséquent les réponses aux questions (a), (b), (d), d'une part et (c), (e) et (f) d'autre part sont identiques.

Les ensembles Nec(◇) et Pos(◇) correspondent à l'ensemble des nœuds pour lesquels la réponse aux requêtes (a) et (f) est positive. Ce résultat est obtenu grâce aux théorèmes 5 et 7 et à la complétude et correction des étiquettes des nœuds. Bien entendu, les ensembles Nec(◇) et Pos(◇) n'ont pas à être calculés pour répondre à une requête. La réponse peut être directement tirée des étiquettes. La caractérisation de Nec(◇) et Pos(◇) ne se préoccupe pas de la consistance des valuations. Il est possible d'obtenir l'équivalent de Nec(◇) et Pos(◇) pour les requêtes (b), (c), (d) et (e) en introduisant les notions de complétions consistantes minimales. Ainsi, même avec une sémantique aussi précise que celle affectée ici aux étiquettes, il est possible d'interpréter les requêtes de façon très variée.

Le problème de l'interprétation des requêtes est assez peu souvent étudié. Il est pourtant crucial en ce sens que c'est au travers des requêtes que le travail du système est perçu. Après avoir imposé, comme cela a été fait au §2.3, la signification des étiquettes et donc de ce qui est connu de la base, il est encore possible d'interpréter les requêtes de six façons différentes. Ces six types de requêtes pouvant se révéler tout aussi utiles les uns que les autres, l'attitude sage, quand la signification des requêtes le permet, est d'autoriser l'utilisateur à demander une réponse correspondant à l'une de ces interprétations.

3.Conclusion

Dans la perspective de concevoir un système de maintenance de la vérité capable de soutenir un raisonnement simultanément non monotone et multi-contextes, un nouveau type d'environnement a été défini pour prendre en compte l'effet d'inférences non monotones.

Un cadre formel a été proposé pour l'interprétation d'environnements capables de prendre en compte l'absence de formules. Cela a mené à la définition précise de ce que doivent être les étiquettes des nœuds maintenus par le système. Cela a aussi permis de signaler et détailler la multiplicité des interprétations possibles d'une requête.

Par rapport aux approches précédentes, il faut noter un certain nombre de différences:

- Une théorie simple de l'incomplétude et de la complétion possible d'un ensemble de formules est proposée. Cette approche n'est plus fondée sur la multiplication des hypothèses au sein d'un système préexistant. Elle tente, au contraire, de prendre en compte les multiples façons à travers lesquelles les mêmes nœuds et hypothèses influent les uns sur les autres.
- Cela a pour conséquence, dans l'interface avec l'extérieur, que l'ensemble des entités manipulées par le système est pertinent. Aucune hypothèse n'est ajoutée dans le but de faire fonctionner le programme.
- Une hypothèse quelconque peut donc être utilisée au sein d'une inférence, que ce soit en partie prémisse ou en partie conséquence.
- Par ailleurs, les résultats obtenus sont suffisamment riches pour pouvoir être interprétés de différentes façons. Ce qui donne lieu à un faisceau de manières d'interroger le système.

La définition proposée n'est pas la seule, des choix ont été faits et seront critiqués plus loin (voir §IV.3). La définition très précise qui a été donnée des étiquettes peut servir telle quelle de spécification à une implémentation d'un système de maintenance de la vérité permettant un raisonnement non monotone et multi-contextes. Cette implémentation est décrite au chapitre suivant.

TROISIÈME CHAPITRE

Fonctionnement du CP-TMS

Une première implémentation d'un système permettant le raisonnement multi-contextes et non monotone est présentée. Elle est fondée sur l'extension d'un système de maintenance de la vérité à propagation et propage, dans le graphe, des étiquettes dont la sémantique est celle exposée au chapitre précédent. L'implémentation des différentes façons d'interroger le système est aussi détaillée.

Troisième chapitre

Fonctionnement du CP-TMS

Fonctionnement du CP-TMS

Le système présenté ici, le CP-TMS (pour «système de maintenance de la vérité à propagation de contextes»), 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. Il implémente le système décrit au chapitre précédent. Il s'agit donc d'établir, pour chaque nœud manipulé par le système de raisonnement, l'étiquette qui lui correspond et dont la signification a été définie précédemment. L'utilisateur, ou le système d'inférence, peut alors consulter la base de faits sous un certain nombre de modes représentés par toute une collection d'opérateurs de requêtes.

L'algorithme détaillé ici n'implémente cependant pas complètement le modèle du chapitre II. En effet, il n'est correct que pour les graphes de dépendances ne contenant ni cycle impair, ni cycle pair alterné. Cette limitation importante conduit à la suppression des extensions multiples. En d'autres termes, à chaque environnement complet ne correspond qu'une interprétation unique. Cela a pour conséquence immédiate de confondre les deux sémantiques principales à accorder aux étiquettes (présent dans une interprétation et présent dans toutes les interprétations).

Les raisons de cette limitation seront exposées lors des critiques de l'algorithme dans le chapitre suivant (voir §IV.2). Les méthodes envisagées pour y remédier seront aussi proposées.

Ce chapitre expose les choix architecturaux du système en fonction de ce qui a été dit sur les différents types de systèmes de maintenance de la vérité (§1). Puis l'algorithme permettant d'étiqueter les nœuds va être présenté à l'aide d'exemples comme l'ont été les deux systèmes principaux (§2). Enfin, les méthodes utilisées pour répondre aux différents types de requêtes en fonction des étiquettes des nœuds sont brièvement exposées (§3).

Ce travail sera comparé aux tentatives antérieures et critiqué dans le chapitre suivant (§IV).

1.Architectures logicielles possibles

L'approche qui est développée ici est d'opérer une fusion entre les systèmes de maintenance de la vérité à propagation et les systèmes de maintenance de la vérité à contextes. Le cadre présenté au chapitre II ne favorise aucun des systèmes existants. Il définit simplement de nouvelles étiquettes et leur attache un sens en fonction de l'étiquetage absolu que peut trouver un système à propagation.

1.1.Architecture du TMS et de l'ATMS

Dès lors, il est légitime de s'intéresser à la structure des deux types de systèmes de maintenance de la vérité pour vérifier si cette fusion est naturelle en terme d'architecture logicielle. Les algorithmes présentés peuvent être décomposés en diverses fonctions importantes: PROPAGATION, RETROGRESSION, RETABLISSEMENT de la consistance. La réaction de ces systèmes lors de l'ajout d'une justification dans le graphe de dépendances peut être présentée sous la forme des schémas des figures 41 et 42.

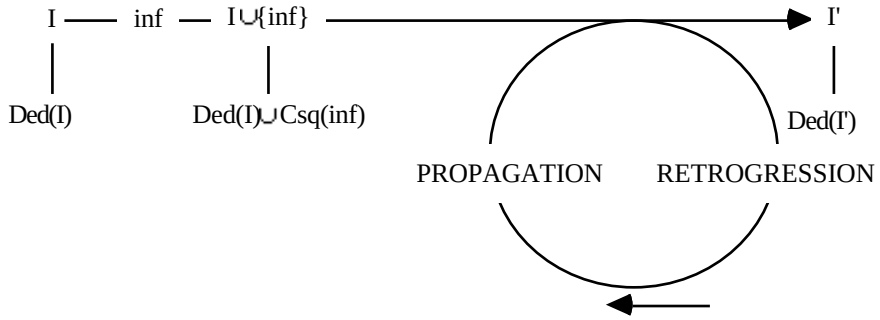


figure 41. Ajout d'une inférence (ou instance de règle) au graphe du TMS. I représente l'ensemble des inférences communiquées au système; Ded(I) est l'ensemble des formules présentes dans la base en fonction des inférences I. Après une première propagation, le rétablissement de la consistance se fait en deux phases. La première (RETROGRESSION) rétablit la consistance immédiate en neutralisant une inférence non monotone tandis que la seconde (PROPAGATION) complète de nouveau l'étiquetage du graphe.

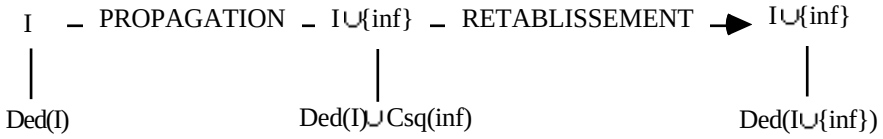


figure 42. Ajout d'une justification dans le graphe de l'ATMS. Ded(I) est maintenant l'ensemble des formules inférées munies de leurs étiquettes. À l'ajout d'une justification (inf), une première phase de propagation permet d'obtenir les nouvelles étiquettes. Elle est suivie du rétablissement de la consistance dans toutes les étiquettes (RETABLISSEMENT) qui supprime les environnements inconsistants.

Il est possible d'envisager plusieurs types de systèmes de maintenance de la vérité suivant un certain nombre de caractéristiques. Les systèmes classiques divergent en effet sur deux caractéristiques:

- La structure de I est un ensemble de justifications autorisant les inférences non monotones dans le TMS alors qu'elles sont strictement monotones dans l'ATMS.

- La structure de Ded correspond à un ensemble de théories (représenté au moyen des étiquettes) dans l'ATMS, alors que ce n'est qu'une seule théorie dans le TMS.

Le cycle PROPAGATION/RETROGRESSION du TMS est le reflet des multiples étiquetages possibles dus à la rétrogression. Le TMS, quand une inconsistance est détectée, complète la base à l'aide d'une nouvelle justification, de façon à supprimer l'inconsistance. Il faut noter que cette complétion peut, dans certains cas, amener la base du TMS dans un étiquetage qui était déjà possible avant la rétrogression mais qui n'avait pas été choisi. L'ATMS, qui maintient parallèlement tous les états consistants, n'a besoin que d'une phase de rétablissement de la consistance qui supprime ceux qui sont inconsistants.

Cette séparation est liée à la façon de rétablir la consistance en jouant sur les inférences non monotones (TMS) ou en jouant sur les hypothèses (ATMS), et à la considération d'un ou plusieurs états en parallèle. Il y a donc une dualité déterminisme/non déterminisme sur les deux façons de rétablir la consistance: la RETROGRESSION du TMS et le RETABLISSEMENT de la consistance de l'ATMS.

1.2.Rétrogression et rétablissement de la consistance

La formalisation des différents systèmes de maintenance de la vérité permet d'isoler dans chaque système les algorithmes adaptés à une caractéristique précise, et ainsi de concevoir des systèmes précis en utilisant les algorithmes adaptés à ces caractéristiques. À cette fin, les deux procédés peuvent être assemblés (voir figure 43). Il semble possible d'adopter la méthode déterministe pour un type de consistance à maintenir et la convention non déterministe pour un autre type.

- Ainsi, utiliser la vision déterministe pour la gestion monotone de la consistance immédiate et la vision non déterministe pour la gestion non monotone permettra d'implémenter facilement un système où les axiomes sont fixés, mais pas les inférences non monotones qui sont possibles à partir de l'ensemble d'axiomes. C'est le cas du système à propagation.
- Le choix du déterminisme total est celui d'un système dans lequel les axiomes considérés comme des hypothèses peuvent être abandonnés par le système lui-même.
- Le choix de l'indéterminisme total risque de compliquer la structure des ensembles de déductions et par conséquent leur traitement. Cela correspond au cas de figure où le système ne choisit jamais d'hypothèses ou d'inférences à invalider mais présente comme résultat l'ensemble des conséquences consistantes.
- Enfin, le choix du déterminisme pour la gestion de la non monotonie et de l'indéterminisme pour la gestion de l'inconsistance classique ne semble pas très justifiable: c'est celui qui correspond à fixer arbitrairement une unique extension considérée par environnement complet (voir §II.1). C'est le cas du système à contextes à ceci près que celui-ci n'a qu'une interprétation par environnement complet.

RETROG. \ RETAB.	déterminisme	non déterminisme
déterminisme	TMS à propagation	ATMS
non déterminisme	—	—

figure 43. Quatre types de systèmes de gestion de la non monotonie possibles en fonction de la façon de faire la rétrogression et le rétablissement de la consistance.

Ainsi, une fusion idéale des deux types de systèmes semble être celle dénotée par la figure 44, où le système est capable de rétrogression dans les environnements inconsistants et peut supprimer ces environnements si la rétrogression échoue. Cette idée permet de construire un système où à chaque environnement correspond une extension choisie par le système (voir §II.1). Toutefois, elle semble impossible à réaliser: la rétrogression doit ajouter des justifications dans les environnements inconsistants, mais il n'est pas normal que ceux-ci soient visibles dans les autres environnements. Le réseau de dépendances étant partagé par tous les environnements, la justification est difficile à masquer. C'est pourquoi le système, implémenté ici, ne suit pas cette voie.

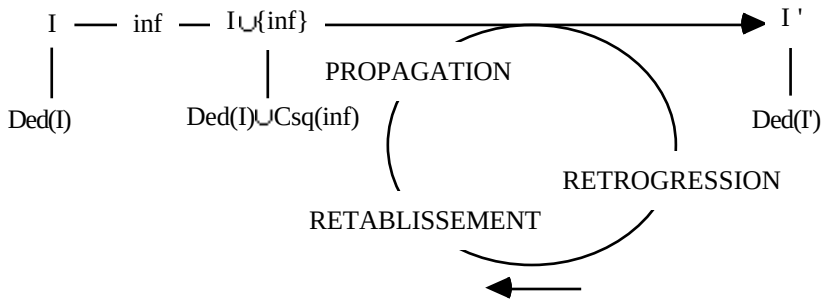


figure 44. Gestion des inférences non monotones. La consistance n'est plus seulement rétablie en neutralisant une inférence non monotone (RETROGRESSION) mais aussi au moyen du retrait d'environnements intrinsèquement inconsistants (RETABLISSEMENT) qui vient s'insérer avant la nouvelle phase de propagation (PROPAGATION).

1.3.Solution retenue

Pour élaborer de nouveaux systèmes de gestion de la non monotonie, il est préférable de ne pas s'enfoncer dans l'ajout des caractéristiques sur des mécanismes déjà existants, mais plutôt de bien considérer les problèmes afin de concevoir un système très adapté à partir des caractéristiques requises. Il n'est bien sûr pas question de rejeter le travail qui a déjà été fait, mais plutôt d'éviter la fusion bancale de deux outils qui ont été conçus dans des perspectives très différentes.

Ainsi, cette constatation ne permet pas, de manière immédiate, la fusion des systèmes de maintenance de la vérité en vue de réaliser un système de gestion de la non monotonie. Même s'ils permettent de décomposer ceux-ci en modules capables de traiter des problèmes spécifiques, ces modules sont trop spécifiques pour pouvoir être transplantés d'un système vers l'autre. Cela ne signifie pas que la fusion soit impossible mais qu'elle doit être produite à un niveau de détail plus prononcé.

Il est possible de prolonger chacun des deux systèmes séparément (voir figure 45) pour obtenir divers systèmes de gestion de la non monotonie totalement différents, mais aucun mécanisme générique n'aura été mis au point pour traiter une caractéristique précise des systèmes de maintenance de la vérité.

	Structure des étiquettes	RETABLISSEMENT	RETROGRESSION
TMS à contextes	environnements	non déterministe	NonTraité
étendu	environnements	non déterministe	non déterministe
TMS à propagation	IN/OUT	NonTraité	déterministe
étendu	IN/OUT	déterministe	déterministe

figure 45. Classification des systèmes existants et de leurs extensions possibles.

Au lieu d'étendre l'ATMS par une couche supérieure ou des mécanismes internes, afin qu'il puisse utiliser des justifications non monotones (voir §I.3.1), l'approche retenue étend le TMS pour qu'il puisse utiliser les environnements. Ce choix a été fait principalement parce qu'il semblait naturel de propager les environnements nouvellement définis au sein d'un graphe de dépendances contenant des inférences non monotones, suivant la méthode utilisée par Jon Doyle. Par ailleurs, les algorithmes utilisés par Doyle sont connus et diverses adaptations y ont été apportées. À l'opposé, l'algorithme de Johan De Kleer n'a été décrit que récemment (voir §IV.1.3).

Cette approche suppose donc de modifier le TMS, afin qu'il ne tienne plus compte de l'état (IN ou OUT) dans lequel se trouve un objet, mais qu'il se contente de propager l'étiquette de cet objet. Cette extension utilise les environnements particuliers, présentés au chapitre précédent, qui tiennent compte de la nature non monotone des inférences.

Reprendre, encore une fois, l'interface exposée plus haut, va permettre de se rendre compte des fonctionnalités qu'apportera un tel système.

Le protocole de communication entre le système d'inférence et le système de maintenance de la vérité rassemble les traits communs aux deux types de systèmes de maintenance de la vérité:

- L'utilisateur peut définir des hypothèses.
- Les inférences communiquées peuvent être non monotones.

Par ailleurs, la richesse des étiquettes affectées aux nœuds permet au système d'inférence de consulter la base de faits selon différentes modalités.

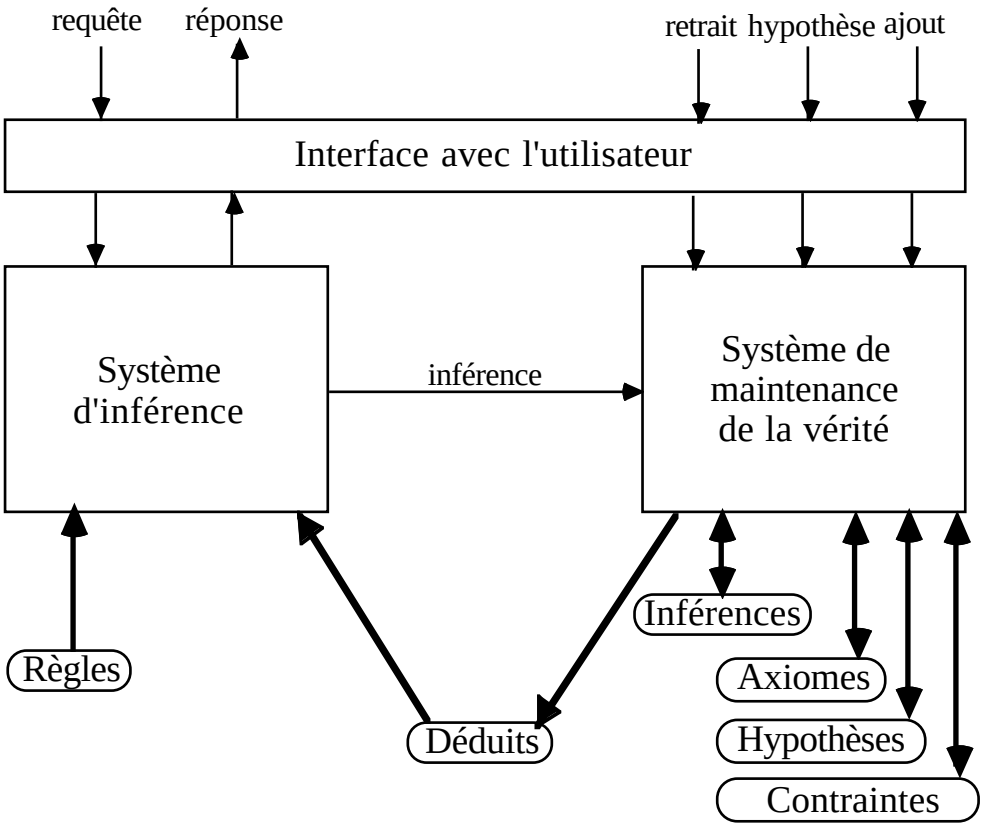


figure 46. Représentation fonctionnelle d'un système capable de produire un raisonnement non monotone. Ce schéma est celui prévu par la figure 1, il assemble ceux concernant TMS et ATMS: les hypothèses et les inférences non monotones sont toutes deux prises en compte.

Comme auparavant, le système d'inférence va communiquer toutes les inférences qu'il produit au système de maintenance de la vérité. Comme au sein du système à propagation, ces inférences peuvent être non monotones. Le système de maintenance de la vérité conservera les inférences (`Inférences`) afin de maintenir l'ensemble des formules inférables. Le système de maintenance de la vérité à propagation propageait la présence et l'absence des nœuds à travers le graphe. Le CP-TMS, lui, ne s'intéresse plus à la présence absolue des formules, mais, comme au sein du système à contextes, aux ensembles d'hypothèses sous lesquelles ces formules sont présentes. Le système étiquettera les nœuds correspondant aux formules en fonction des ensembles d'hypothèses qui permettent de les inférer.

La propagation du système de Jon Doyle agit en deux phases: la première phase cherche des supports fortement fondés pour les nœuds et la seconde, en désespoir de cause, cherche des supports faiblement fondés. Ces deux phases étant liées à la présence de boucles et vu les limitations proposées, elles n'ont plus de raison d'être dans le système propageant les étiquettes. Elles seront donc réduites à une phase unique décrite dans la suite (§2).

Le système de Jon Doyle contient aussi une phase de rétrogression utilisée quand le nœud $\perp$ devient IN. Ce nœud n'étant plus présent absolument, la phase de rétrogression est remplacée par une phase de rétablissement de la consistance du graphe de dépendances. En effet, au lieu de garantir que le nœud $\perp$ n'est pas présent dans l'environnement courant, le système se charge de garantir que l'utilisateur ne pourra pas se placer dans un quelconque environnement permettant la présence de $\perp$.

Une requête permet au système de raisonnement de consulter une partie du graphe, c'est-à-dire de demander si une formule est présente sous un certain nombre d'hypothèses. Cet aspect du système est discuté plus loin (§3).

2.Algorithme

L'interface entre le système de raisonnement et le système de maintenance de la vérité a gardé toute sa simplicité: il ne s'agit jamais que de fournir une inférence, éventuellement non monotone, au système. Cette inférence est transformée en justification, avec les antécédents dans les IN- et OUT-listes:

$$<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>: f$$

Cette justification est insérée dans le graphe de dépendances comme pour le système de maintenance de la vérité à propagation. Le graphe de dépendance est exposé en utilisant le même symbolisme qu'au chapitre I.

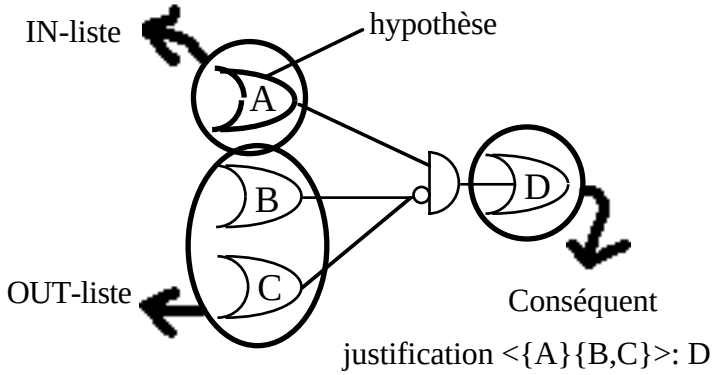


figure 47. Une inférence est composée d'une IN- et d'une OUT-liste, comme pour le système de maintenance de la vérité à propagation. Mais le CP-TMS accepte aussi les hypothèses comme le système à contextes.

L'insertion de la justification donne lieu à la propagation des étiquettes à travers le graphe. Les étiquettes des nœuds, définies au chapitre précédent, répondent aux propriétés de correction, complétude et consistance. Une propriété de minimalité faible leur est adjointe ici (voir §2.3). Elles sont stockées par le système sous une forme compactée. Au lieu d'être des ensembles d'environnements de la forme:

$$\{[a_{11},\ldots a_{1p_1}|b_{111},\ldots b_{11q_{11}}],\ldots\ [a_{11},\ldots a_{1p_1}|b_{1m_11},\ldots b_{1m_1q_{1m_1}}],$$
$$\ldots$$
$$[a_{n1},\ldots a_{np_n}|b_{n11},\ldots b_{n1q_{n1}}],\ldots[a_{n1},\ldots a_{np_n}|b_{nm_n1},\ldots b_{nm_nq_{nm_n}}]\}$$

elles sont en quelque sorte factorisées en:

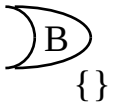
$$\{[a_{11},\ldots a_{1p_1}|\{(b_{111},\ldots b_{11q_{11}}),\ldots\ (b_{1m_11},\ldots b_{1m_1q_{1m_1}})\}],$$
$$\ldots$$
$$[a_{n1},\ldots a_{np_n}|\{(b_{n11},\ldots b_{n1q_{n1}}),\ldots(b_{nm_n1},\ldots b_{nm_nq_{nm_n}})\}]\}.$$

Un élément de cet ensemble est appelé un *environnement global*; il est composé d'un ensemble d'axiomes (qui est un ensemble d'hypothèses) et d'un ensemble de restrictions (qui est un ensemble d'ensembles d'hypothèses). Si cette forme permet de réduire en mémoire la taille des environnements, elle ne constitue pas en elle-même une forme minimale. Il est cependant possible, pour les implémentations, d'adopter diverses formes internes classiques pour représenter les formules logiques, comme la forme canonique disjonctive (qui a le défaut d'être extrêmement redondante: les algorithmes seraient alors très simples mais les environnements contiendraient au pire $2^{|\mathfrak{H}|}-1$ environnements) ou des arbres de décision [Billon 87, Madre& 88].

L'algorithme va être présenté comme lors du chapitre 1, à l'aide d'exemples; un rapport le décrit plus en détail ([Euzenat 89a]).

2.1.Déclaration des items

La déclaration de nœuds et d'hypothèses ne diffère pas du principe de l'ATMS. Le nouveau nœud est doté de l'étiquette vide (rien ne permet de connaître une interprétation dans laquelle il est présent), alors que la nouvelle hypothèse reçoit l'environnement où elle est présente dans l'ensemble d'axiomes.



48a. Création d'un nœud.

À la création du nœud B, l'étiquette vide lui est associée. B n'est présent dans aucun environnement.

48b. Création d'une hypothèse.

À la création de l'hypothèse A, l'étiquette contenant pour seul environnement [A|] est ajoutée à son étiquette.

Cela signifie que dans toutes les interprétations bâties à partir de la présence de l'hypothèse, cette hypothèse est présente.

2.2.Propagation

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.

Si ce principe est simple, il n'est pas directement applicable pour trois raisons: des circularités peuvent être présentes dans le graphe de dépendances qui feraient boucler le programme, les environnements ainsi obtenus ne seraient ni minimaux ni consistants et enfin l'exécution prendrait un temps considérable. L'algorithme proposé tente de remédier à ces trois problèmes:

- Les cycles pairs stratifiés dans le graphe sont correctement traités de la même façon que par le système à contextes. Par contre, l'algorithme ne supporte pas les cycles pairs alternés (ses résultats sont incorrects), ni les cycles impairs (il lui arrive de boucler).
- Les étiquettes des justifications et des nœuds ne sont recalculées que si cela est nécessaire.
- Les étiquettes obtenues sont vérifiées comme consistantes et faiblement minimales.

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:

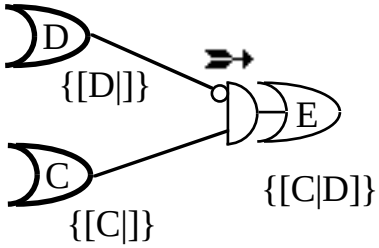
- La stratégie du support est abandonnée. Cette stratégie permettait de ne recalculer l'étiquette d'un nœud que si un antécédent précis responsable de l'état actuel du nœud voit son état modifié. Les états d'un nœud n'existant plus, il n'est plus question de maintenir une telle stratégie, et l'étiquette du nœud doit être 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 tous les nœuds conséquents. Cette modification de l'incrémentalité permet de résoudre le problème (3) de ART (voir §I.3.2).
- La décomposition du graphe de dépendances en composantes fortement connexes utilisée par James Goodwin [Goodwin 87] est conservée. Elle permet de remettre à jour les étiquettes des nœuds d'une composante seulement si tous les nœuds des composantes précédentes ont déjà été examinés. Suite à l'abandon de la stratégie du support, les composantes sont réellement celles du graphe de dépendances.
- 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.

L'amorce de cette propagation (l'ajout de justifications) est décrite ci-dessous.

48c. Inférence si les deux antécédents sont des hypothèses.

Dans le cas présent (j1=<{C}{D}>: E), la nouvelle étiquette est aisée à calculer. Chaque étiquette ne contenant qu'un environnement, ceux-ci sont réunis en un seul.

L'étiquette correspondant à la justification est donc {[C| D]}: C et D étant deux hypothèses, la justification est valide quand C est présent dans la base et D en est absent. Conformément à l'interprétation du réseau par Jon Doyle, D est absent partout où il n'est pas présent. L'étiquette est propagée à son conséquent qui n'a pas d'autres justifications.



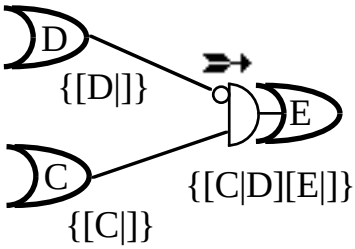
48d. Ajout d'une justification vide.

j2=<{}{}>:B

L'environnement obtenu est le sommet du graphe d'environnement ([|]). Cette justification dénote une inférence valide inconditionnellement (elle n'a pas d'antécédents), le nœud B est donc présent sans condition sur la présence ou l'absence d'autres nœuds.

48e. Déclaration d'une hypothèse déjà inférée

La déclaration du nœud E en tant qu'hypothèse va ajouter l'environnement où il est présent ([E|]) à son étiquette. Elle devient donc {[C|D][E|]}



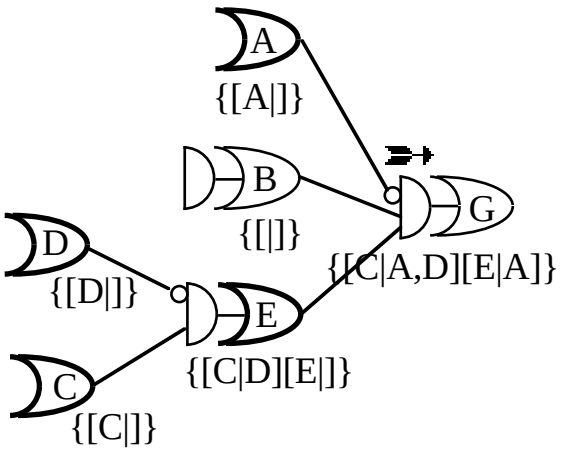
L'algorithme de propagation présenté ici agit donc comme si l'étiquette était une formule logique exprimée en forme normale disjonctive:

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

Cette approche ne pose pas de problème en ce qui concerne la conjonction et la disjonction. Par contre, la négation ne se trouve pas forcément à sa place car l'absence d'un item dans une base ne correspond pas à sa négation (bien que ce soit effectivement sur elle qu'est fondée l'inférence). Cette interprétation se trouve cependant correcte dans le cas où les cycles alternés ne sont pas acceptés. En effet, l'ensemble des interprétations étant connues, et les environnements maximaux n'ayant qu'une interprétation, les items sont bien présents dans toutes les interprétations de tous les environnements de leur étiquette (et leurs complétions) *et absents de toutes les interprétations des environnements qui ne sont pas complétions d'un environnement de leur étiquette*. Ces derniers correspondent justement à ce qui est obtenu en appliquant la négation à l'équivalent logique de leur étiquette.

La propagation se conforme alors exactement à la signification des éléments de la représentation graphique du graphe de dépendances:

- l'étiquette d'un nœud est la disjonction des étiquettes de ses antécédents (les justifications).
- l'étiquette d'une justification est la conjonction des étiquettes de ses antécédents (OUT-liste et nœuds de la IN-liste).
- l'étiquette d'une OUT-liste est la conjonction des négations des étiquettes de ses antécédents (nœuds de la OUT-liste).



48f. Double propagation.

La propagation n'est pas toujours aussi simple que dans l'exemple 48d. Maintenant que l'étiquette du nœud E contient deux environnements, ce sont ces deux environnements qu'il faut propager à chaque fois que E ou l'un de ses conséquents intervient dans une inférence.

Ainsi, pour l'inférence

j3=<{B, E}{A}>: G

l'étiquette contiendra bien le produit des étiquettes des antécédents. Il faut remarquer à cette occasion que {[|]} est bien l'élément neutre pour l'opération de conjonction d'étiquettes

2.3.Minimalité

Concernant la minimalité, le principe retenu a été celui dit de *minimalité faible*: une étiquette est minimale s'il n'existe pas dans cette étiquette deux environnements comparables l'un à l'autre (l'un est plus complet que l'autre). 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 manipulées par le CP-TMS.

Le problème est qu'il existe plusieurs formes normales disjonctives pour une formule quelconque comme celles considérées ici. Qui plus est, il existe plusieurs formes normales «minimales» possibles. Garantir la minimalité des formules en forme normale est très complexe. Toutes les vérifications doivent être faites à chaque étape de la propagation: en supposant que les étiquettes des nœuds soient faiblement minimales, si la conjonction de deux étiquettes minimales rend une étiquette minimale, il est possible de conserver cette minimalité. L'algorithme de disjonction doit être transformé pour garantir la minimalité faible. Ainsi, à chaque disjonction, les environnements sont comparés entre eux afin de supprimer ceux qui sont plus complets que d'autres.

Cet algorithme rend des étiquettes faiblement minimales dans le sens où il ne rajoute pas d'environnements comparables à d'autres environnements. Par contre, il ne fait jamais de fusion d'environnements globaux. 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 peut sembler plus minimale. Il serait souhaitable d'ajouter à cet algorithme un autre algorithme, activé à chaque fois que l'étiquette d'un nœud est déterminée, qui maintienne une minimalité plus forte opérant la fusion d'environnements globaux.

Il est tentant de poser le principe de minimalité forte: une étiquette est fortement minimale s'il n'existe pas d'étiquette logiquement équivalente comportant un nombre moindre d'environnements. Mais ce principe ne garantit toujours pas un minimum unique (par exemple {[a,b|],[|a]} $\Leftrightarrow$ {[b|],[|a]}) et n'est pas très opérationnel.

Il faut noter, par ailleurs, que le raisonnement n'étant plus monotone, la croissance des étiquettes ne l'est plus non plus. En effet, la présence de certains nœuds entraînant l'absence d'autres, si les premiers sont présents sous plus d'environnements, les seconds le seront sous moins d'environnements (voir figure 49).

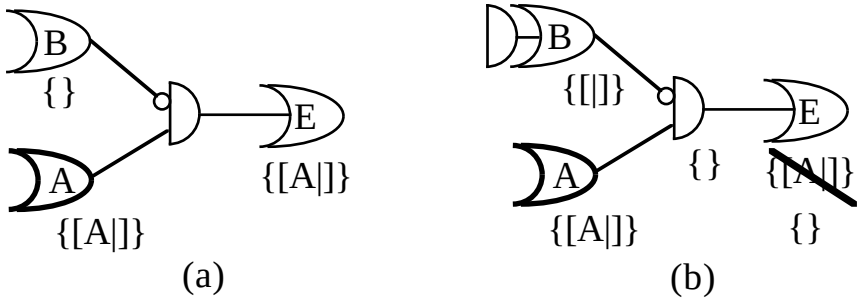


figure 49. En l'absence d'autre connaissance, la formule E est présente dans toutes les complétions de l'environnement [A|], puisque la formule B qui pourrait l'empêcher n'est présente nulle part. Mais quand il y a plus d'information permettant d'établir la présence de B, et donc plus d'environnements dans lesquels B est présent, il y a moins d'environnements dans lesquels E est présent. Il n'est donc pas possible de propager la nouvelle étiquette ({}) et de l'unir à celles déjà présentes dans le nœud E ({[A|]}) car le résultat ne serait pas correct ({[A|]}≠{}).

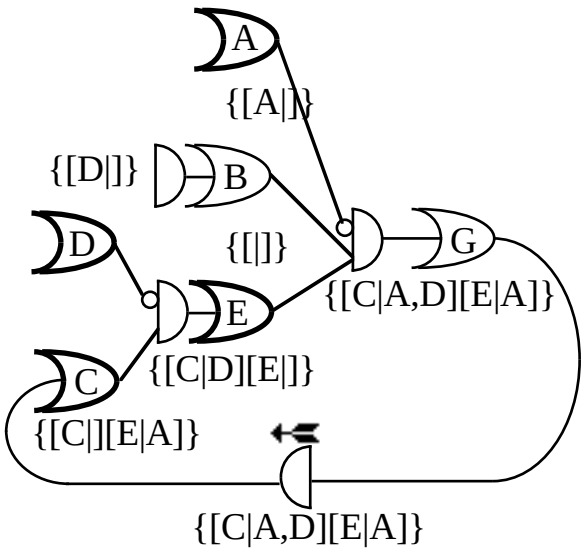
La solution à ce problème consiste à conserver avec chaque justification l'étiquette qui y correspond, et, quand une justification change d'étiquette, de recalculer l'étiquette de tous ses conséquents en s'appuyant sur la nouvelle étiquette. Toute la propagation se fait alors ainsi: une étiquette de nœud n'est jamais recalculée en fonction de son ancienne étiquette mais toujours à partir des étiquettes de ses justifications qui sont de nouveau associées. Par contre, si la nouvelle étiquette obtenue est la même que la précédente, l'algorithme arrête de propager. Cette pratique permet d'éviter de boucler lors de cycles pairs et de ne pas perdre toute l'incrémentalité des systèmes de maintenance de la vérité initiaux:

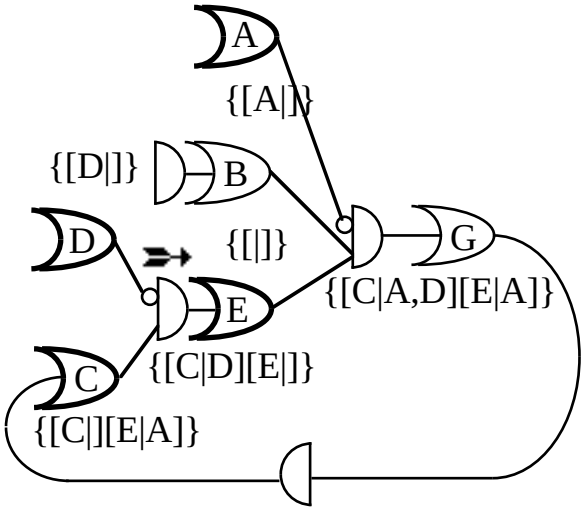
48g. Propagation incrémentale (jusqu'à C).

Si une nouvelle justification est ajoutée au graphe de la figure 48f:

j4=<{G}{}>:C,

l'étiquette de cette nouvelle justification est calculée. C'est la même que celle de G: {[C|A,D][E|A]}. Elle est ajoutée à l'étiquette de C ce qui donne {[C|A,D][E|A][C|]}.
Après minimisation ([C|] est plus général que [C|A,D]), la nouvelle étiquette de C est {[C|][E|A]}.





48h. Propagation incrémentale (jusqu'à E).

Il faut alors propager cette nouvelle étiquette vers le nœud E via la justification j1. La nouvelle étiquette correspondant à j1 est le produit des étiquettes de C et D:

{[C|D][E|A,D]}.

Ajoutée à l'étiquette de E, celle-ci devient

{[E|][C|D][E|A,D]}

ce qui rend après minimisation ([E|] est plus général que [E|A,D]), l'étiquette

{[E|][C|D]}.

Cette dernière étiquette étant celle de E avant la propagation, celle-ci s'arrête, n'atteignant même pas G.

48i. Justifications multiples.

Si, reprenant la figure 48f, G reçoit la justification

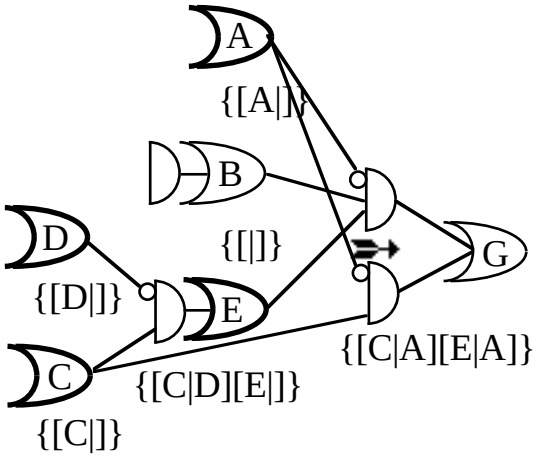
j5=<{C}{A}>:G,

il faut faire la disjonction des étiquettes correspondant à chacune des justifications. Les étiquettes de j3 et j5 étant respectivement {[E|A][C|D,A]} et {[C|A]}, leur union donne

{[E|A][C|D,A][C|A]}.

Mais cette étiquette n'est plus minimale. La minimalité est rétablie en supprimant les environnements qui sont plus spécifiques que d'autres (ici [C|D,A] est plus spécifique que [C|A]). L'étiquette de G est donc

{[E|A][C|A]}.



2.4.Consistance interne

La *consistance interne* est ainsi nommée car elle ne dépend pas des contraintes présentes dans le système. Elle se borne à cerner des étiquettes bien formées, sans considérer si ces étiquettes contiennent des environnements caractérisant un environnement inconsistant. Plutôt que les configurations inconsistantes, ce sont les configurations impossibles — ou paradoxales — qui sont éliminées.

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[H]={…, […|…H…], …}.

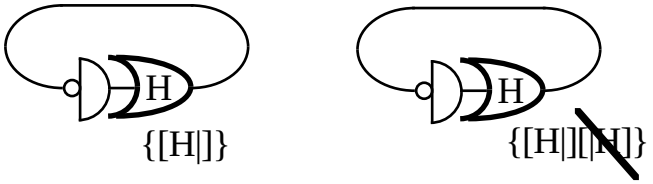


figure 50. Un cycle impair peut engendrer des environnements facilement reconnaissables quand l'un de ses membres est une hypothèse. Il est alors facile de dépister ces cycles et d'annuler leur effet. Malheureusement ce type d'environnement n'est pas systématique.

Par ailleurs, il faut aussi vérifier que le système ne propage pas d'environnements insensés ("meaningless"), c'est-à-dire d'environnements comprenant dans leur restriction un nœud de leurs axiomes. C'est le cas par exemple de l'environnement

[…H…|…H…].

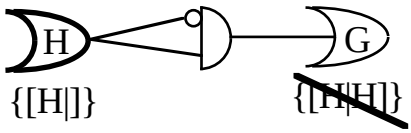


figure 51. Un type bien particulier d'inférence qui ne peut jamais fonctionner. Il a l'avantage d'être facilement détectable. Son effet est supprimé du graphe sans autre conséquence. Il faut noter la facilité avec laquelle ceci est fait: il n'est pas besoin de déclencher de nouvelles contraintes pour interdire ces environnements.

Pour rétablir la consistance interne, tous les environnements inconsistants sont ôtés des étiquettes dans lesquelles ils apparaissent. En principe, l'algorithme ne traite pas les cycles impairs, mais il est capable, à l'aide de ce mécanisme, d'en éliminer certains. Ces deux causes d'inconsistances n'apparaissaient pas dans le système de Johan De Kleer; elles sont toutes deux conséquences de la présence de restrictions dans les environnements.

2.5.Inconsistance externe

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, et en fin de propagation, quand $\perp$ a son étiquette modifiée, en supprimant du graphe tous les environnements inconsistants.

Comme spécifié dans le chapitre précédent, chaque environnement ayant une complétion commune avec un environnement de l'étiquette de $\perp$ est remplacé par l'ensemble de ses complétions consistantes minimales. La méthode utilisée consiste donc à disloquer tous les environnements plus généraux qu'un environnement de l'étiquette de $\perp$, de façon à ne conserver que la partie disjointe de l'étiquette de $\perp$.

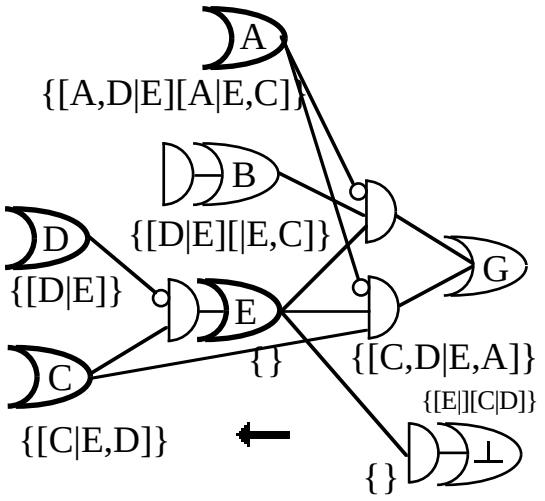
Vis-à-vis de la considération des étiquettes comme des formules logiques, cette opération est très simple, puisqu'il suffit d'exclure de tout environnement les environnements de l'étiquette de ⊥. Cela revient simplement à remplacer tout environnement E par

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

Lors de la découverte de nouveaux environnements dans l'étiquette du nœud ⊥, il est possible que le réseau ne soit plus consistant. Il faut alors rétablir la consistance externe en supprimant ces nouveaux environnements de toutes les étiquettes des nœuds où ils se trouvent. Comme cela a été vu plus haut, la présence d'inférences non monotones fait que la nouvelle étiquette de ⊥ n'est pas forcément plus générale que la précédente. Cela signifie que des environnements, qui étaient considérés précédemment comme inconsistants, peuvent ne plus l'être suite à l'ajout d'une inférence (ces environnements étaient inconsistants par défaut).

Ainsi, les étiquettes contenues dans le graphe peuvent être bien trop restreintes par rapport à ce qu'exige la consistance: elles ne sont plus à jour. Il faut alors relancer toute la propagation avec la nouvelle étiquette de ⊥. Il est donc souhaitable que les contraintes soient communiquées au système de maintenance de la vérité, avant toute autre justification, ou qu'elles ne soient inférées que grâce à des inférences monotones.

À la propagation, il faut s'assurer que toutes les étiquettes présentes dans le système sont consistantes. Les fonctions utilisées lors de la propagation sont modifiées de façon très simple: chaque étiquette calculée est vérifiée comme consistante en s'assurant qu'aucun de ses environnements ne permet la présence de ⊥.



48j. Rétablissement de la consistance.

Suite à l'ajout de la justification j6=<{}{E}>: ⊥, l'étiquette obtenue pour ⊥ est celle de E:{[E|][C|D]}.

Il faut rétablir la consistance de toutes les étiquettes. Pour cela, le complémentaire de l'étiquette de ⊥ est calculé, c'est

{[|E,C],[D|E]}.

Elle va être ajoutée à toutes les étiquettes contenues dans le graphe, ce qui donne:

pour A: {[A,D|E],[A|E,C]}

pour B: {[D|E],[E,C]}

pour C: {[C|E,D][C|E,C]} ce qui, une fois rétablie la consistance interne, donne {[C|E,D]}

pour D: {[D|E],[D|E,C]} ce qui, une fois minimisé, donne {[D|E]}

pour E: {}

pour G: {[E|E,C,A], [D,E|A,E], [C|A,C,E], [C,D|A,E]} ce qui, une fois la consistance interne rétablie, donne {[C,D|E,A]}

3.Traitement des requêtes

Après avoir montré comment le CP-TMS établit les étiquettes, il faut montrer comment il permet d'y accéder. Le cadre formel défini au chapitre précédent a permis de distinguer diverses façons de demander la présence d'une formule dans la base en fonction d'un contexte d'interrogation.

Une requête est une question de la forme «La formule F est-elle présente dans le contexte [A/R]?». [A/R] est alors nommé le contexte d'interrogation. Pour mémoire, les différents types de requêtes, référencés de (a) à (f), sont rappelés ici:

a) F est-elle présente dans toutes les complétions de la valuation correspondant à [A/R]?
b) F est-elle présente dans toutes les complétions consistantes minimales de la valuation correspondant à [A/R]?
c) F est-elle présente dans certaines complétions de toutes les complétions consistantes minimales de la valuation correspondant à [A/R]?
d) F est-elle présente dans une complétion consistante minimale (au sens de toutes les complétions d'une complétion consistante minimale) de la valuation correspondant à [A/R]?
e) F est-elle présente dans certaines complétions d'une complétion consistante minimale de la valuation correspondant à [A/R]?
f) F est-elle présente dans une complétion de la valuation correspondant à [A/R]?

où «présente dans une complétion» signifie qu'il existe une interprétation de cette complétion pour laquelle F est présente.

Les différentes fonctions décrites ici permettent d'obtenir des renseignements sur la requête. Elles sont suffisamment puissantes pour permettre de répondre aux différentes interprétations d'une requête.

Il n'est pas difficile de répondre à une requête si la valuation correspondant au contexte de requête est complète. Il suffit alors de répondre en la comparant avec les environnements de l'étiquette de F. Cela n'est pas non plus difficile si cette valuation est seulement complète vis-à-vis de l'étiquette, ce qui est défini comme suit: un environnement [A/R] est *complet vis-à-vis d'une étiquette* L s'il existe un environnement [A'/R'] de L tel que $A' \subseteq A$ et $R' \subseteq R$, ou si pour tout environnement [A'/R'] de L $A \cap R' \cup R \cap A' \neq \{\}$. La première possibilité correspond à la catégorie «nécessairement présent» de Johan De Kleer (voir §I.2), la seconde à «nécessairement inconsistant», c'est-à-dire à la présence nécessaire de $\perp$. Dans le premier cas, la réponse à la requête (a) est affirmative, dans le second elle est négative. Si le contexte d'interrogation n'est pas complet vis-à-vis de l'étiquette de F, alors les problèmes posés par l'information incomplète resurgissent, et c'est là que les autres types de requêtes sont pertinents.

Lors d'une requête [A/R]:F?, il est possible de fournir tout simplement à l'utilisateur l'ensemble des conditions de la présence de F, c'est-à-dire l'ensemble des hypothèses à ajouter à [A/R] pour que F soit présente. Le résultat sera alors le même que celui que donne le système SHERLOCK [Cordier 87], à savoir une réponse conditionnelle: «F est présent si telles hypothèses sont présentes et telles hypothèses sont absentes». Cette opération est appelée la *différence entre une étiquette et un contexte* (elle est noté [A/R] Δ ETIQUETTE[F]). Cet ensemble se présente comme une étiquette, telle que pour chaque environnement [A'/R'] de celle-ci, F soit présente sous [A$\cup$A'/R$\cup$R'] et (A$\cap$A'$\cup$R$\cap$R')$\cup$(A$\cap$R'$\cup$A'$\cap$R)={}. Cette opération de différence s'implémente de manière simple à partir de sa définition. Il est aisé de vérifier qu'elle conserve la minimalité et la consistance de l'étiquette initiale.

Dès lors, répondre à la requête (a) est simple puisqu'il suffit de vérifier que:

$$[|] \Subset [A/R]\Delta\text{ETIQUETTE}[F].$$

La réponse à la requête (f) est tout aussi simple puisqu'il suffit que:

$$[A/R]\Delta\text{ETIQUETTE}[F] \neq \{\}.$$

Rechercher si une formule est présente dans le contexte proposé par l'utilisateur correspond à l'attitude à adopter. L'utilisateur est alors considéré comme connaissant la nature du contexte dans lequel il se place. A priori, une étiquette ne contient aucune complétion inconsistante (à cause de la propriété de consistance). Mais le contexte d'interrogation peut être inconsistant, auquel cas il est bon de se rabattre sur une de ses complétions consistantes minimales, c'est l'objet des requêtes référencées (b)-(e).

Ces requêtes font appel à la notion de complétions consistantes minimales. La fonction calculant les complétions consistantes minimales est donc largement utilisée (il est clair que si [A/R] est un environnement consistant, il n'a qu'une complétion consistante minimale: lui-même). La procédure permettant d'obtenir les complétions consistantes minimales est celle utilisée dans l'algorithme présenté plus haut pour rétablir la consistance. Elle utilise la formulation logique des étiquettes présentée avec l'algorithme.

Pour répondre aux autres requêtes, les deux premières requêtes doivent donc être itérées sur les complétions consistantes minimales, soit pour en trouver une les satisfaisant, soit pour qu'elles les satisfassent toutes.

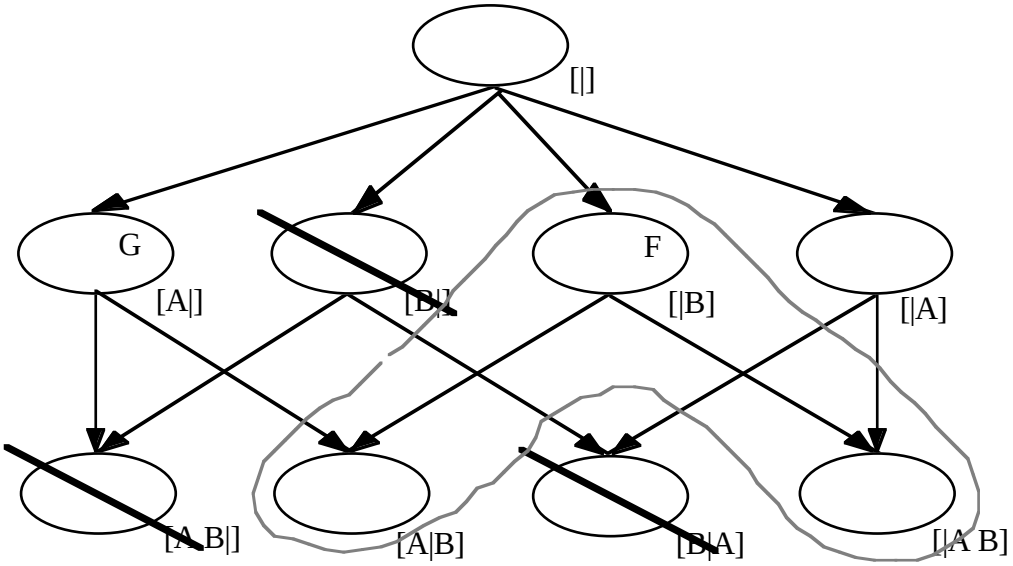


figure 52. Cette figure reprend la figure 37, où G est inféré dans l'environnement [A|] et F dans l'environnement [|B]. L'environnement [B|] étant inconsistant, les seuls environnements possibles dans les étiquettes sont donc [|B], [A|B] et [|A,B]. L'étiquette de G est donc {[A|B]} et celle de F {[|B]}.

Concernant l'exemple de la figure 52, la complétion consistante minimale de [A|] est [A|B], celle de [|] est [|B] et celle de [|B], [|B] elle-même. Les réponses aux requêtes seront les suivantes:

requête	[A|]:G	[|]:G	[|]:F	[|B]:F
(a)	*	N	N	O
(b)	O	N	O	O
(c)	O	O	O	O
(d)	O	N	O	O
(e)	O	O	O	O
(f)	O	O	O	O

(*) signifie que la réponse à la requête est positive en considérant qu'un environnement inconsistant permet la présence de n'importe quelle formule, ou négative s'il ne doit pas exister de complétion inconsistante du contexte d'interrogation.

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 voulue de la requête. Il est à remarquer que tous les algorithmes utilisés par le système pour obtenir les étiquettes sont indépendants du critère de requête de l'utilisateur, ce qui apporte une très grande flexibilité au système. Cela peut, par contre, être un handicap en terme d'efficacité des algorithmes (par exemple, si toutes les requêtes posées ne dépendent pas de l'inconsistance, alors tous les efforts produits par l'algorithme pour rendre les étiquettes consistantes sont inutiles). Il est possible de les simplifier pour ne prendre en compte qu'une partie des requêtes proposées ici.

4.Conclusion

L'extension du système de maintenance de la vérité à propagation qui a été présentée 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 le système implémenté est soumis à une limitation importante: en effet, la présence de cycles impairs et surtout de cycles pairs alternés ne permet pas d'obtenir les résultats attendus par le modèle. Le type de raisonnement non monotone admissible par le système implémenté est donc particulier.

Quoi qu'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. Les ensembles d'hypothèses sont maintenant implémentés à l'aide de vecteurs de bits et les algorithmes sont optimisés en ordonnant les environnements globaux dans les étiquettes et les vecteurs dans les ensembles de restriction pour faciliter leur accès. Il permet de déclarer hypothèses et inférences et répond à différents types de requêtes. Cette implémentation est détaillée par ailleurs [Euzenat 89a, b].

Le chapitre suivant va permettre de comparer l'implémentation qui vient d'être décrite avec les systèmes déjà existants. Cette comparaison conduira à une critique de l'implémentation qui a été faite, permettant de proposer des solutions nouvelles.

QUATRIÈME CHAPITRE

Critiques et perspectives

Le CP-TMS décrit au chapitre précédent est comparé ici avec les autres systèmes. Cela permettra ensuite de discuter les options utilisées pour son implémentation. Cette discussion mènera à réexaminer à son tour le modèle présenté au chapitre II.

Quatrième chapitre

Critiques et perspectives

Critiques et perspectives

En réponse aux problèmes liés à la trop grande spécialisation des systèmes de maintenance de la vérité, une fusion des deux systèmes existants a été proposée. Elle permet de produire un raisonnement simultanément non monotone et multi-contextes. Ce système a été défini formellement et a été implémenté comme une extension du système de maintenance de la vérité à propagation.

Les relations entre le système implémenté et d'autres vont être explicitées (§1), avant de critiquer l'implémentation qui a été faite (§2). Enfin, la formalisation qui a donné lieu au système sera, à son tour, discutée (§3).

1.Relation avec les systèmes existants

La conception du CP-TMS n'est pas un acte isolé. Ce système va donc être replacé parmi les nombreuses recherches sur les systèmes de maintenance de la vérité dont il est inspiré et les autres.

1.1.Le TMS

Si toutes les étiquettes sont {} (OUT) ou {[|]} (IN) le travail de propagation du CP-TMS est le même que celui fait par le TMS. Même en cas de cycles pairs alternés, les résultats obtenus seront les mêmes que ceux du système de Jon Doyle. En effet, tout comme ce dernier, le CP-TMS choisit arbitrairement les extensions dans lesquelles il se place. En cas de cycles impairs, le système présenté bouclera contrairement à l'implémentation de James Goodwin. Toutefois, l'efficacité du CP-TMS sera moindre car il est nécessaire de tenir compte des opérations sur les environnements. 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.

La différence principale est dans la rétrogression. Il n'est plus possible de faire de la rétrogression comme le faisait le système de Jon Doyle. En particulier, le système présenté ici est incapable de rétablir la consistance après qu'une inconsistance a 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 89a].

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

1.2.MBR

MBR ("Multiple Belief Reasoner"[Martins& 83, 88] sera simplement évoqué ici afin de le comparer au CP-TMS. C'est un ancêtre de l'ATMS destiné à raisonner dans plusieurs contextes. Ce système, comme l'ATMS, n'accepte pas d'inférences non monotones et n'accepte en théorie qu'une inférence par nœud. Les environnements qu'il génère sont, contrairement à l'ATMS, composés

- d'un ensemble d'axiomes (appelé *ensemble d'origine*) qui constitue l'environnement initial dans lequel le nœud a été dérivé,
- d'un *ensemble de restrictions* qui constituent les hypothèses qui ne doivent pas être ajoutées à l'ensemble d'origine sous peine d'inconsistance.

Dans leur structure, ces environnements sont donc similaires aux environnements globaux du CP-TMS, mais la signification en est différente. En effet, les restrictions de MBR ne sont pas liées à l'inférence de la formule, mais à l'ensemble d'axiomes en lui-même (tous les nœuds ayant même ensemble d'origine auront même ensemble de restrictions). Ces ensembles de restrictions sont engendrés lors de la phase de rétablissement de la consistance qui est faite exactement comme dans le CP-TMS, c'est-à-dire que le système travaille pour éviter d'avoir à considérer des complétions inconsistantes (voir §II.2.4).

Le principal intérêt de MBR est d'être construit sur un système logique dénotant son action. Au vu des caractéristiques du système, il est raisonnable de penser que compte tenu des restrictions concernant MBR, le résultat obtenu en utilisant le CP-TMS est similaire. L'avantage du CP-TMS est donc de pouvoir prendre en compte des inférences non monotones et plusieurs justifications pour un même nœud.

1.3.L'ATMS

Le système présenté perd en simplicité 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 une étiquette mais plusieurs formes, ce qui ne facilite pas les algorithmes de recherche. Pour déterminer si une formule est présente, le treillis perd alors beaucoup de son intérêt puisqu'il ne suffit plus de comparer les environnements (ou plutôt leurs ensembles d'axiomes) de l'étiquette d'un nœud au contexte courant; il faut ensuite comparer les restrictions.

Il est possible d'utiliser le CP-TMS à la place de l'ATMS. Pour obtenir le même résultat, il faut atténuer la définition de la consistance d'une étiquette en exigeant simplement qu'un environnement ne soit pas plus spécifique qu'un environnement inconsistant. Alors, tous les environnements seront sans restriction. Le CP-TMS sera moins efficace, car il devra tenir compte des opérations sur les restrictions et ne profitera pas de la minimalité.

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

- PROPAGATE correspond à l'algorithme général de propagation d'étiquette.
- NOGOOD correspond au rétablissement de la consistance 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.

Bien sûr, l'ATMS n'a pas à considérer la consistance interne.

La différence avec l'algorithme de l'ATMS est que la partie propagation d'étiquettes est bien plus complexe à cause de la présence des justifications non monotones. Les problèmes posés par ce type d'environnement ont entraîné:

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

L'avantage principal du CP-TMS est qu'il permet d'exprimer simplement des inférences s'appuyant sur l'absence de formules dans la base.

1.4.L'ATMS étendu d'Oskar Dressler

L'une des méthodes utilisée par Oskar Dressler consiste simplement à forcer tous les membres des OUT-listes à être des hypothèses et à interdire d'inférer ces hypothèses. Quand cette option est choisie pour le CP-TMS, il y a alors équivalence entre les environnements du CP-TMS:

$$[i_1,\ldots i_n|o_1,\ldots o_m]$$

et les environnements de l'extension de l'ATMS:

$$\{i_1,\ldots i_n, \mathrm{Out}(o_1),\ldots \mathrm{Out}(o_m)\}.$$

Il faut cependant restreindre le système d'Oskar Dressler à ne pas définir de cycles alternés pour obtenir l'équivalence avec le CP-TMS.

Il faut noter, sous cette option, que si les deux systèmes sont formellement équivalents, le travail fait par celui d'Oskar Dressler est beaucoup plus lourd. Il n'est qu'à comparer les figures 27 et 36. Elles ont exactement la même signification. Le problème est que l'espace considéré par l'ATMS contient 16 environnements alors que celui du CP-TMS n'en contient que 9. Par ailleurs, les 7 environnements supplémentaires dans l'ATMS doivent être invalidés à l'aide de règles d'inconsistances (dont le travail est coûteux) alors que le CP-TMS opère, lors de la propagation, sans aucune difficulté à l'aide de la notion d'inconsistance interne, non présente dans l'ATMS.

Le problème le plus important de cette approche est conceptuel. Le système d'Oskar Dressler pose comme hypothèses l'ensemble des entités pouvant donner lieu à plusieurs extensions en déclarant comme hypothèses tous les membres des OUT-listes, ce qui est également proposé pour le CP-TMS. Cette solution n'est pas complètement satisfaisante, car elle détourne le concept d'hypothèse, très clair et très conforme à l'intuition, tel qu'il est utilisé par Jon Doyle ou Johan De Kleer. En effet, les inférences non monotones sont des règles qui servent à poser des hypothèses, alors qu'ici les hypothèses sont utilisées pour déclencher ces mêmes inférences.

Le CP-TMS bien qu'il soit, dans ce cas particulier, équivalent au système de Dressler présente certains avantages sur celui-ci dans le cas général:

- Le système reste homogène; tous les nœuds sont traités de la même façon ce qui n'est pas le cas des nœuds Out(N), introduits par Oskar Dressler, qui ne peuvent être justifiés.
- En conséquence le caractère propositionnel du système a été préservé: il n'est pas besoin de se préoccuper de la structure des faits associés aux nœuds pour savoir, par exemple, si ce sont des nœuds de type Out(N).
- Il évite la multiplication des nœuds et hypothèses supplémentaires qui n'ont pas été soumis par l'utilisateur ou le système de raisonnement.
- Enfin, le CP-TMS est plus souple, il n'oblige aucun nœud (pas même les nœuds présents dans les OUT-listes) à être déclaré comme hypothèse.

1.5.Formalisation en algèbre de Boole

Un autre travail a permis d'exprimer les systèmes de maintenance de la vérité dans le cadre de l'algèbre de Boole [Brown& 87]. Ce travail est plus proche des algorithmes de maintenance de la vérité. Pour Allen Brown, un ensemble de nœuds peut être représenté sous forme de variables booléennes. À chaque variable est associée une équation booléenne, fonction des autres variables, sous forme normale disjonctive. Cette équation correspond à la disjonction des justifications du nœud. Un nœud N sans justification ou un nœud appartenant à $\mathcal{E}$ étant représenté par l'équation N=0. À un ensemble de justifications et de nœuds correspond donc un système à $|\mathcal{N}|$ équations.

Il existe des algorithmes permettant d'obtenir les solutions d'un tel système — ne contenant pas de cycle impair — grâce aux opérations de substitution et de normalisation. À l'ensemble des solutions d'un tel système — une solution étant représentée par l'ensemble des nœuds dont la variable est égale à 1 — correspond l'ensemble $\Gamma(\mathcal{N},\mathcal{J},\mathcal{C})$, mais le modèle va plus loin puisqu'il permet d'exprimer la notion de solution fortement fondée pour un nœud. De plus la spécification algébrique rend compte aisément d'optimisations du type de l'algorithme de James Goodwin [Goodwin 87]. En effet, le système d'équations peut se diviser en sous-systèmes «fortement connexes» et ordonnés. L'algorithme pour la résolution des équations est alors fortement similaire à l'algorithme de James Goodwin.

Par exemple, en reprenant le graphe de la figure 7 (composé par les justifications <{}{}>:A, <{}{D}>:B et <{A}{B,C}>:D), le système d'équations est le suivant:

$$\begin{cases} A=1 \\ C=0 \\ B=\sim D \\ D=A\&\sim B\&\sim C, \end{cases}$$

ce qui donne la liste de systèmes ordonnés et connexes:

$$\{A=1, \{C=0, \begin{cases} B=\sim D \\ D=A\&\sim B\&\sim C, \end{cases}$$

et l'ensemble de solutions $\Gamma(\mathcal{N},\mathcal{J},\mathcal{C})=\{\{A,B\},\{A,D\}\}$.

L'intérêt du système algébrique est qu'il permet d'exprimer aussi le comportement du système de maintenance de la vérité à contextes. En effet, un ensemble d'hypothèses peut être considéré comme des nœuds n'apparaissant pas dans la partie gauche d'une équation. Ces hypothèses combinées par l'opération de conjonction forment un treillis ordonné par la relation d'implication.

Les solutions des systèmes d'équations ne seront donc plus réduites à 1 ou 0 mais à une formule booléenne fonction des hypothèses. Les hypothèses sont distinguées par des capitales et les nœuds simples par des minuscules, ainsi la résolution d'une équation consiste en une égalité entre un nœud (simple ou hypothèse) et une formule booléenne fonction uniquement d'hypothèses.

Appelons S(N) la formule booléenne correspondant au nœud N, cette fonction peut être mise en forme normale disjonctive (sans négations); ainsi, déterminer la présence du nœud N dans un contexte C revient à savoir si l'environnement correspondant au contexte permet de déduire la valeur 1 pour la solution de l'équation en N:

$$\text{N présent dans le contexte C} \Leftrightarrow C \vdash S(N).$$

Par exemple, en reprenant le graphe de la figure 21g (composé par les justifications <{D,C}>:E, <{}>:b, <{A,b,E}>:g et <{A,C}>:g et des hypothèses A, C, D et E), le système d'équations est le suivant:

$$\begin{cases} A=A \\ b=1 \\ C=C \\ D=D \\ E=E|(D\&C) \\ g=(A\&b\&E)|(A\&C), \end{cases}$$

ce qui donne le système résolu suivant:

$$\begin{cases} A=A \\ b=1 \\ C=C \\ E=E|(D\&C) \\ g=(A\&E)|(A\&D\&C)|(A\&C)=(A\&E)|(A\&C), \end{cases}$$

Cette formalisation n'apporte rien non plus par rapport aux algorithmes utilisés mais permet, en réutilisant des résultats déjà connus en algèbre, de prouver la correction des algorithmes existants. Son originalité est qu'elle permet de rapprocher les deux types de systèmes de maintenance de la vérité. Elle permet aussi de se rendre compte aisément de la difficulté à utiliser le treillis des systèmes de maintenance de la vérité à contextes dans un cadre où des justifications non monotones sont permises. En effet, celles-ci, en introduisant des négations dans les formes normales, introduisent des conditions complexes sur le treillis.

Le système rend donc compte de manière indépendante du TMS et de l'ATMS. L'idée du CP-TMS étant de fusionner les deux systèmes, il semble intéressant de pouvoir représenter celui-ci dans le cadre des systèmes d'équations booléennes. Le travail présenté ici est issu de ces travaux destinés à trouver le cadre formel permettant d'exprimer à la fois TMS et ATMS. Le rapport entre les deux travaux est évident, c'est ce qui a conduit à justifier les algorithmes présentés ici en tant que manipulations de formules propositionnelles.

La solution est en effet relativement simple, elle considère, comme dans la modélisation du TMS, des équations avec négations mais, comme dans celle de l'ATMS, ces équations ne sont résolues qu'en fonction des hypothèses. Cependant, les problèmes se posant avec le CP-TMS se reproduisent dans la résolution de ces équations. En effet, en cas de cycles pairs alternés, il n'est pas possible de trouver une solution (au moins unique) en fonction d'hypothèses. Par contre, dans le cas de systèmes ne présentant pas cette particularité, la solution obtenue pour chaque équation est une formule exprimable sous forme normale disjonctive — correspondant à l'étiquette du nœud. Les algorithmes pour obtenir de telles formules en formes normales sont exactement les mêmes, aux optimisations près, que ceux utilisés par le CP-TMS.

De même que précédemment, pour le graphe de la figure 48i (composé par les justifications <{C}{D}>:E, <{}{}>:b, <{b,E}{A}>:g et <{C}{A}>:g et des hypothèses A, C, D et E), le système d'équations est le suivant:

$$\begin{cases} A=A \\ b=1 \\ C=C \\ D=D \\ E=E|(\sim D\&C) \\ g=(\sim A\&b\&E)|(\sim A\&C), \end{cases}$$

ce qui donne le système résolu suivant:

$$\begin{cases} A=A \\ b=1 \\ C=C \\ E=E|(\sim D\&C) \\ g=(\sim A\&E)|(\sim A\&\sim D\&C)|(\sim A\&C)=(\sim A\&E)|(\sim A\&C), \end{cases}$$

ce qui est bien l'étiquetage rendu par le CP-TMS.

1.6.Programmes logiques stratifiés

La stratification d'un programme logique avec négation P consiste en une partition $P_1,\ldots P_n$ des symboles de prédicats qu'il définit. Chaque classe P_i est appelée une strate. Cette partition est telle que pour tout i ($1\leq i\leq n$) et pour chaque prédicat p dans P_i, les clauses définissant p ne font référence:

- qu'à des littéraux utilisant des prédicats des strates P_1 à P_i,
- qu'à des littéraux niés utilisant des prédicats des strates P_1 à P_{i-1}.

Cela permet de classer les définitions de prédicats de telle sorte qu'un prédicat p faisant appel dans sa définition à un prédicat q dans un littéral nié est forcément dans une strate supérieure à celle de q. Un programme logique stratifié interdit donc la définition récursive d'un prédicat à travers la négation. C'est-à-dire qu'en considérant le graphe de dépendances entre les prédicats dont les arcs signifient «… apparaît dans un littéral positif d'une des clauses définissant…» et «… apparaît dans un littéral négatif d'une des clauses définissant…» il n'y a pas de cycles contenant des arcs de la deuxième sorte dans ce graphe. Ce faisant, un programme logique stratifié a un et un seul modèle minimal «privilégiable» appelé modèle standard.

La notion de graphe de dépendance stratifié a été introduite pour les systèmes de maintenance de la vérité [Fujiwara& 89]. C'est exactement un graphe ne contenant ni cycle impair, ni cycle pair alterné. Pour un tel graphe, on montre qu'il existe un et un seul étiquetage possible.

Jean-Marc Pugin et Krzysztof Apt [Apt& 87, Pugin 88] ont quant à eux développé un système de maintenance de la vérité fondé sur le graphe de dépendances entre prédicats pour maintenir une base de données déductive stratifiée (SGBDD). Ces travaux ont plusieurs points communs avec ceux exposés ici:

- Les systèmes construits agissent sur le même type de données (graphe de dépendances).
- Le SGBDD tire parti des strates maximales pour optimiser la propagation. Le CP-TMS utilise les composantes fortement connexes ce qui, en l'absence de cycles impairs ou pairs alternés, coïncide exactement [Pugin 88].

Mais il existe plusieurs différences:

- Les programmes du SGBDD vérifient a priori la stratification du programme logique alors que le CP-TMS ne vérifie pas l'absence de cycles impairs ou pairs alternés.
- Le SGBDD peut manipuler des variables alors que le CP-TMS est purement propositionnel. S'il est aisé de réduire le premier au second (en ne considérant que des prédicats sans arguments), il n'est pas immédiat d'étendre le second vers le premier, mais le SGBDD pourrait servir de modèle.
- Le SGBDD conserve continuellement les strates alors que le CP-TMS recalcule les composantes concernées à chaque propagation. Ainsi, si la mise à jour de la base est incrémentale, c'est aussi le cas de la stratification. C'est vers ce genre de programme (conservant les composantes) que s'orientent les futurs travaux autour du CP-TMS (voir §2.3, [Euzenat 89c]). Les algorithmes développés pour le SGBDD devraient donc être profitables à ce dernier.
- Enfin, le CP-TMS a l'avantage de traiter plusieurs contextes simultanément. L'ajout de cette fonctionnalité au SGBDD semble possible en adaptant les programmes existants (la partie concernant la stratification reste alors la même).

1.7.Interprétations «multivers» des bases de données incomplètes

Le modèle proposé au chapitre II pourrait se révéler être une extension des interprétations de l'incomplétude dans les bases de données [Lipski 81]. En particulier, la vision d'un environnement comme représentant l'ensemble des environnements complets plus complets que lui est très proche de la notion d'interprétation «multivers» [Ostermann 88]. À une base de donnée incomplète est associée l'ensemble de ses modèles (son multivers). Les multivers sont reliés entre eux par une relation appelée ordre d'information. À chaque multivers correspondant à une base de donnée complète ne correspond qu'un unique modèle. Ainsi, une base ne contenant rien sur la validité de A ni de B est présenté par la figure 53.

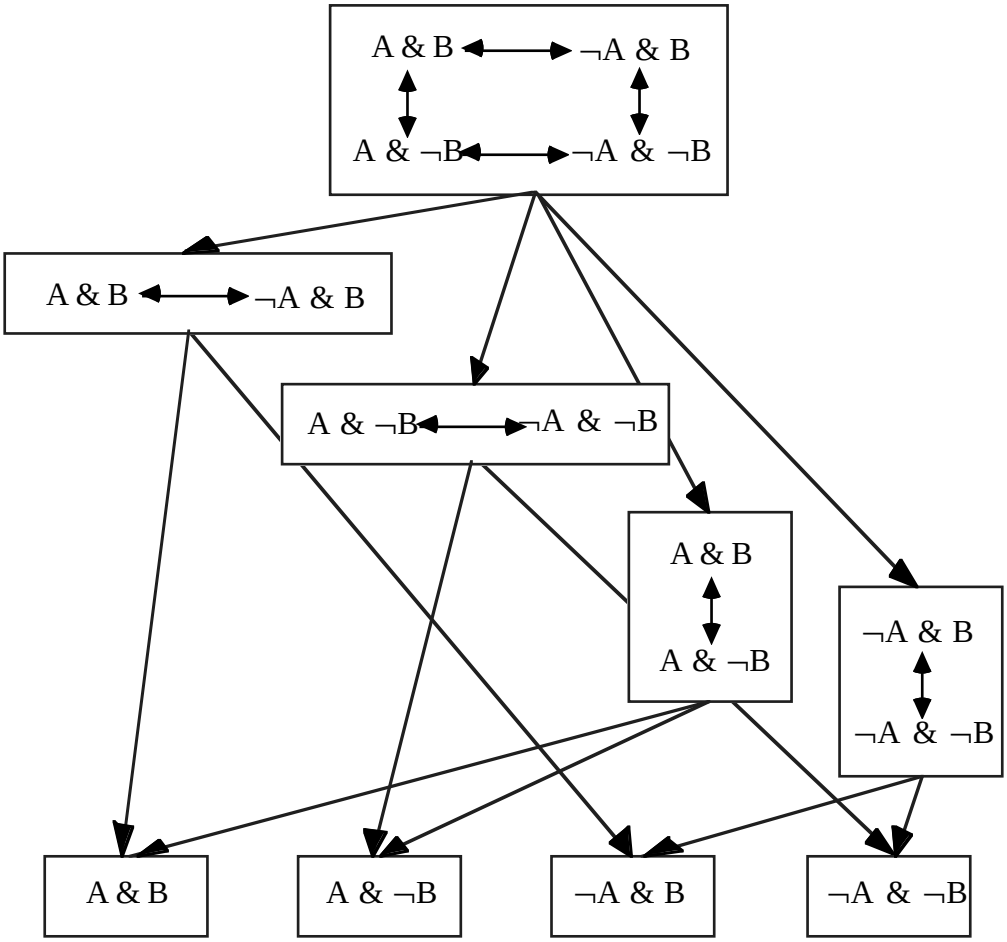


figure 53. Interprétation d'une base de données incomplète dans laquelle les assertions A et B n'ont pas de valeur de vérité. Chaque multivers est représenté par un rectangle pointillé. Les multivers de la seconde couche représentent les états de la base dans lesquels la validité (ou l'invalidité) d'une des deux assertions est connue, et les multivers de la dernière couche représentent des bases complètes. Il faut remarquer que ce graphe est exactement celui de la figure 34.

Cette construction peut s'interpréter de deux façons:

- À l'intérieur de chaque multivers, il est possible de considérer chaque modèle comme un monde de logique modale et de considérer comme relation d'accessibilité la relation universelle (clôture transitive et réflexive des flèches à l'intérieur du multivers). Chaque multivers est alors un modèle de la logique S5.
- Entre les multivers, il est possible de les considérer comme des mondes et d'adopter l'ordre d'information (clôture transitive et réflexive des flèches en pointillé) comme relation d'accessibilité. On obtient alors un modèle de la logique IDL [Ostermann 88].

Cela permet de définir les notions de «nécessairement présent» et «nécessairement absent» à l'aide des modalités classiques de logiques modales combinées entre elles ($\Box$ et $\Diamond$ dans l'un quelconque des mondes pour la première interprétation et $\Box\Diamond$ et $\Diamond\Box$ dans le monde désiré pour la seconde). Ces modalités permettent d'interpréter les requêtes exactement comme cela a été défini dans le modèle.

Si l'ensemble des multivers et l'ordre d'information correspondent exactement à l'ensemble des environnements du modèle du chapitre II, l'interprétation en logique modale des bases de données ne prend pas explicitement en compte les inférences et a fortiori les inférences non monotones. C'est en ce sens que le modèle utilisé dans ce travail peut être une extension du travail fait sur les bases de données [Euzenat 89d].

2.Critique de l'algorithme

Comme cela a été déjà signalé, le CP-TMS souffre, quant à lui, de certaines limitations. Elles sont reprises et illustrées ici afin de tenter de leur trouver une solution. Certaines solutions ne sont valables qu'en reconsidérant le modèle d'environnements qui sera donc discuté plus loin (§3).

2.1.Incrémentalité

L'algorithme du CP-TMS présenté n'est 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.

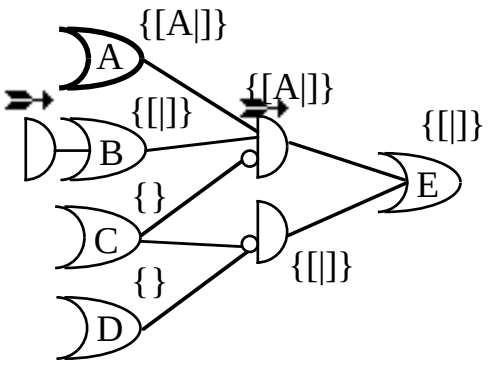


figure 54. Propagation incrémentale. Quand le nœud B est justifié en tant qu'axiome, sa nouvelle étiquette est obtenue ({[|]}). Comme elle est différente de la précédente, la propagation s'attaque à la justification suivante (<{A,B}{C}>:E), sa nouvelle étiquette est calculée ({[A|]}), elle doit aussi être propagée. Au lieu de reconsidérer les anciennes étiquettes, le système fait l'union de l'ancienne étiquette de E ({[|]}) et de l'étiquette de la justification, ce qui après minimisation donne {[|]}. Comme rien n'est changé, il n'y a pas de propagation ultérieure.

Malheureusement, à cause des inférences non monotones, ce dernier système est incorrect dans le cas général. En effet, si la propagation incrémentale était correcte dans l'ATMS, c'était grâce à la monotonie du raisonnement produit: une formule présente sous un environnement ne peut pas être considérée comme absente en considérant plus d'information. Comme le montre la figure 55, cela est possible en cas d'inférences non monotones. Le système incrémental est cependant correct s'il est utilisé de concert avec un système de raisonnement produisant un raisonnement linéaire (une fois établi une justification, plus aucun de ses antécédents ne peut être justifié). Un tel système a de meilleures performances et l'avantage de détecter les cycles impairs.

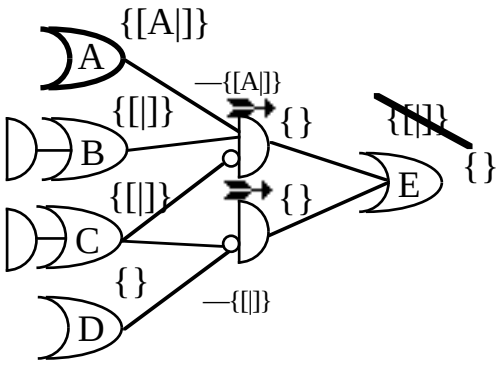


figure 55. Si C est justifié à son tour en tant qu'axiome, sa nouvelle étiquette va être {[|]}. Propager cela aux deux justifications suivantes affectera à toutes deux la nouvelle étiquette {}. Faire l'union de cette nouvelle étiquette avec l'ancienne {[A|]} n'est pas une bonne solution, car cela laisserait l'ancienne étiquette au nœud E ({[|]}). Il faut donc, comme avant, propager des différences négatives, c'est-à-dire que la différence, entre l'ancienne étiquette de chaque justification et la nouvelle, doit être ôtée à l'étiquette du conséquent: {[|]}\{[|]} donnera la nouvelle étiquette {} pour le nœud E.

Retrouver l'incrémentalité doit passer par l'implémentation de différences négatives (voir figure 55). Toutefois, cette solution ne peut éviter de refaire certains calculs: en cas de propagation de différence négative, il faut impérativement recalculer l'étiquette complète des conséquents (voir figure 56).

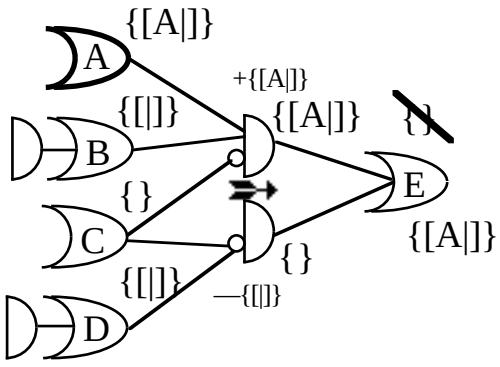


figure 56. Si, à la place de C, D est justifié comme axiome, alors sa nouvelle étiquette va être {[|]}. Si, comme cela a été indiqué plus haut, le système propage la différence négative, il aurait obtenu le même résultat que précédemment ({} pour E). Or ce résultat n'est pas correct. Pour qu'il le soit, il faut refaire l'union de toutes les étiquettes de la justification ce qui donnera l'étiquette de E:{[A|]}.

Le seul algorithme possible est donc incrémental tant que la nouvelle étiquette est plus générale que la précédente (propagation de différences positives) et correspond exactement à l'algorithme décrit au chapitre précédent quand ce n'est pas le cas.

2.2.Cycles impairs

Comme le montre la figure 57, le système présenté au chapitre III peut boucler en présence de cycles impairs.

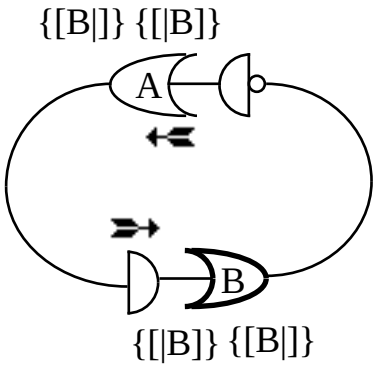


figure 57. Soient un nœud A et une hypothèse B, lorsque sont ajoutées les deux justifications j1=<{A}{}>:B et j2=<{}{B}>:A, la propagation se fait ainsi:
B:{[B|]} j1:{[|B]} A:{[|B]} j2:{[|B]}
B:{[B|],[|B]}={[|]} j1:{} A:{} j2:{}
B:{[B|]}…

Il est possible d'adapter l'algorithme à la détection des cycles impairs 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. Un tel algorithme permettra d'éviter de boucler mais ne garantit pas que les cycles seront détectés s'ils existent. En effet, il se peut que l'algorithme s'arrête avant de revenir sur le nœud, si la nouvelle étiquette d'un des nœuds du cycle est identique à l'ancienne.

2.3.Interprétations multiples

La principale limitation de l'algorithme est de ne pouvoir considérer de graphe acceptant plusieurs étiquetages dans un même environnement complet. Cela est reflété par la restriction de ne pas avoir de cycles pairs alternés, qui garantit l'unicité de l'étiquetage dans chaque environnement complet.

La figure 58 montre bien ce que fait l'algorithme dans ce cas précis.

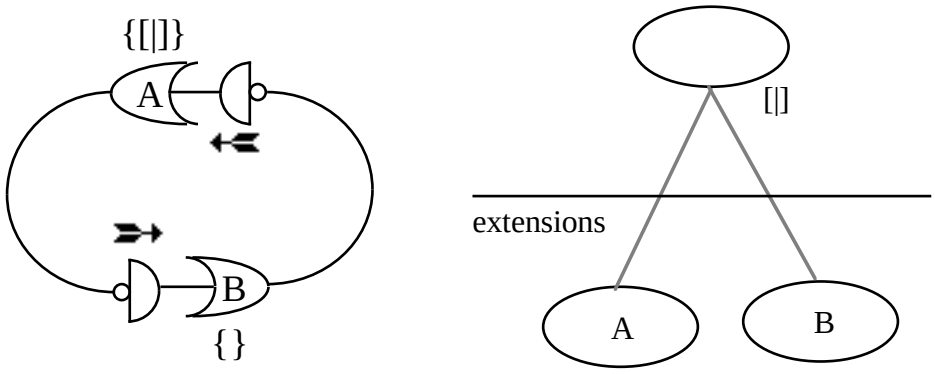


figure 58. Soient deux nœuds A et B, sans justifications, ils ont tous deux l'étiquette {}. Lorsque la justification j1=<{}{B}>:A est ajoutée, la propagation trouve pour A l'étiquette {[|]}. Cela est parfaitement correct parce que par défaut, B est absent de la base, et quand B est absent, A est présent. Maintenant, quand la justification j2=<{}{A}>:B est ajoutée, le CP-TMS trouve pour B l'étiquette {} qu'il avait déjà. En effet, le nœud A étant présent dans tous les environnements, son absence ne pourra jamais valider B. En fait, cet étiquetage est incorrect: vis-à-vis de la spécification du chapitre II, les deux nœuds doivent être étiquetés {[|]}.

Il est possible de penser que l'algorithme est plus proche de l'interprétation des étiquettes de manière universelle: un environnement est dans une étiquette si la formule correspondante doit être présente dans toutes les extensions de cet environnement. C'est en effet ce que semble dénoter la réaction de l'algorithme à l'ajout de la seconde justification (j2). Cela est faux car l'étiquetage ne correspond pas non plus à cette définition puisqu'il devrait être {} pour les deux nœuds (A et B).

En fait, l'explication est plus simple: l'algorithme a hérité des propriétés du système de Jon Doyle. Tout se passe comme s'il choisissait l'extension dans laquelle il se place et qu'il étiquetait les nœuds en fonction de cette extension. Ainsi, la troisième branche du choix qui a été posée au §II.1 semble la plus naturelle à implémenter avec un système dérivé du TMS. Il n'en reste pas moins que les arguments utilisés pour le choix tiennent toujours et que la meilleure solution n'est certainement pas de se satisfaire du CP-TMS tel qu'il est actuellement.

L'implémentation utilisée n'est donc pas conforme aux spécifications en ce sens qu'elle ne peut tenir compte des extensions multiples. La manière de l'améliorer consiste à essayer de détecter, au cours de la propagation (ou avant), les configurations de graphe qui engendrent ces extensions multiples. Ces configurations sont appelées des *générateurs d'extensions*.

La solution adoptée par le TMS est si insatisfaisante que des recherches sont déjà en cours pour détecter ces générateurs d'extensions. C'est pourquoi la restriction d'utilisation liée au CP-TMS (pas de cycles alternés) a pu être définie. Il est en effet clair que sans cycles pairs alternés, il n'y a pas de multiples extensions. Vu leur définition, il est normal que les trois possibilités soient conformes à l'étiquetage quand il n'existe qu'une extension par environnement maximal.

Pour l'instant le travail de caractérisation de ces générateurs se poursuit sur les systèmes de maintenance de la vérité à propagation [Euzenat 89c]. Les cycles n'engendrant des extensions que dans certains environnements précis, il est plus facile de les caractériser lors d'un étiquetage absolu et d'étendre ensuite la notion aux étiquetages contextuels.

Le travail déjà fait s'appuie sur un découpage du graphe de dépendances en *composantes fortement connexes*, en fonction des supports minimaux des nœuds. L'idée principale de ce mécanisme est de considérer les cycles pairs alternés non contraints comme les ensembles de nœuds qui engendrent les extensions multiples. Exiger que ces cycles ne soient pas contraints c'est exiger qu'ils puissent commuter librement et que des justifications externes au cycle ne viennent pas contraindre le système à ne considérer qu'une extension unique. L'intérêt est que ces cycles, quand ils sont réellement capables d'engendrer des extensions, peuvent se détecter dans une composante fortement connexe.

Néanmoins, cette caractérisation n'est pas suffisante. Tout d'abord parce que suivant le choix des supports pour les nœuds, des cycles pairs non contraints peuvent ne pas être pris en compte dans une composante. Mais aussi parce qu'il existe des *composantes artificielles*, c'est-à-dire des composantes dont les nœuds ne peuvent réellement commuter.

Le travail se poursuit et tend à isoler les générateurs d'extensions dans les composantes fortement connexes du graphe de supports complets et non plus minimaux.

Dans leur utilisation actuelle, ces résultats sont destinés à rapprocher beaucoup plus les résultats du TMS de Doyle de ce qu'attend l'utilisateur. En effet, en cas de découverte d'inconsistance, celui-ci ajoute dans la base un nœud qui permet de lever cette inconsistance. Cela est fait lors de la rétrogression en ajoutant une justification au nœud choisi (élu, voir §I.1.3). Lorsque le système a le choix entre plusieurs étiquetages, il est possible qu'il n'y en ait qu'un d'inconsistant. Il serait alors judicieux d'obtenir les autres étiquetages, ce que le système de Jon Doyle est incapable de faire.

L'algorithme proposé [Euzenat 89c] est fortement fondé sur les composantes du graphe qu'il considère comme des entités pertinentes du raisonnement. Il tente de faire une bonne rétrogression en:

- éliminant, lors de la propagation, les composantes artificielles.
- se plaçant, lors de la rétrogression, dans un étiquetage du graphe initial en recherchant les générateurs d'extensions dans les composantes.

Lorsque ce programme sera achevé, il faudra transposer ces résultats au CP-TMS de façon à dépister les générateurs d'extensions et étiqueter correctement les nœuds du graphe en fonction de la sémantique réelle des étiquettes.

2.4.Correction

En effet, le mode de propagation (en interprétant les étiquettes comme des formules logiques, voir §III.3) n'est pas adapté à la signification assignée aux étiquettes: les deux figures suivantes le montrent pour les deux principales sémantiques des étiquettes.

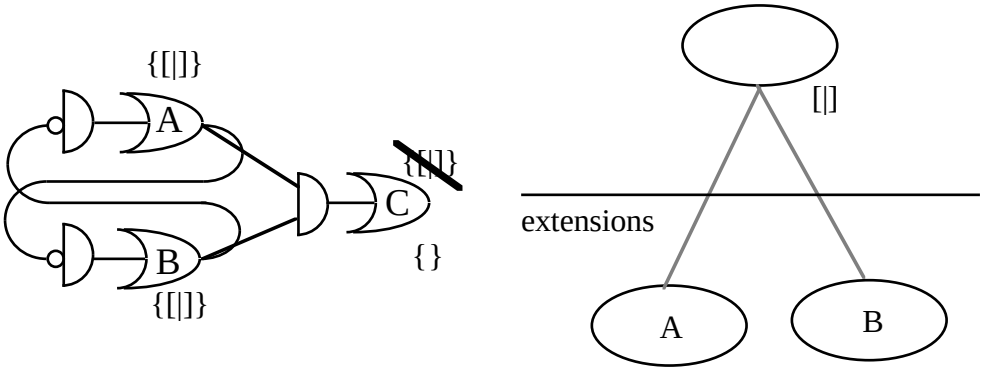


figure 59. Les nœuds A et B sont pris dans un cycle pair alterné: l'absence de l'un justifie la présence de l'autre. Il existe donc deux extensions, l'une dans laquelle B est absent et A présent et l'autre dans laquelle A est absent et B présent. Les deux nœuds sont donc étiquetés par l'environnement [|] dans lequel il existe les deux extensions. La propagation actuelle du CP-TMS étiquette le nœud C avec cet environnement. Or, au vu de la justification <{B,A}{}>:C, C n'est présent dans aucune des deux extensions de cet environnement.

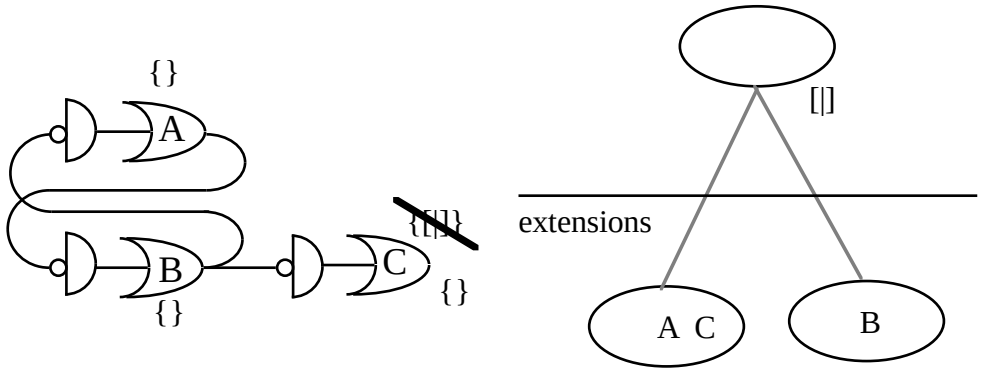


figure 60. La situation est la même que dans la figure 59, mais les environnements des étiquettes signifient que le nœud est présent dans toutes les extensions de cet environnement. Les deux nœuds A et B sont donc étiquetés par {} car le seul environnement autorisé ([|]) contient deux extensions et ni A ni B ne font partie des deux. La propagation actuelle du CP-TMS étiquettera alors le nœud C avec l'environnement [|]. Or, au vu de la justification <{}{B}>:C, C n'est présent que dans une des deux extensions de cet environnement et devrait donc être étiqueté {}.

Le problème dans le CP-TMS n'est donc pas «seulement» de détecter les générateurs d'extensions, il faut aussi modifier complètement le mode de propagation pour tenir compte des extensions. La solution actuellement envisagée est de propager, avec les environnements, les identificateurs des extensions (au sens d'une expression permettant de les identifier) dans lesquels les nœuds sont présents. Cette solution n'est plus dans la ligne du modèle proposé au chapitre II, ou plutôt, elle permettrait de l'étendre considérablement à peu de frais. Aussi, avant de développer cette solution, est-il préférable de passer directement aux critiques du modèle.

3.Critique du modèle

Le modèle qui a été proposé au chapitre II n'est pas exempt de reproche. S'il est assez simple, il ne permet pas de considérer individuellement chaque interprétation du graphe de dépendances, ce qui serait très utile.

3.1.Autre choix sémantique pour les environnements

Le problème de l'interprétation des résultats du CP-TMS en cas de multiples extensions a fait resurgir le choix fait au début du chapitre II. Au vu des critiques de l'implémentation, il semble que l'attitude qui consiste à privilégier une extension par environnement complet soit plus facile à implémenter. Ce n'est pas une raison suffisante pour revenir sur le choix qui a été fait. Par contre, les deux attitudes qui consistent à considérer qu'un environnement doit être dans une étiquette s'il existe une extension dans laquelle la formule est présente, ou si elle est présente dans toutes ses extensions, sont également raisonnables.

Le choix de la sémantique des étiquettes a été dicté par une option minimaliste qui consiste à dire qu'un environnement doit continuer à être considéré tant qu'il est possible qu'une de ses extensions corresponde au monde réel. Le choix maximaliste consiste à ne considérer que les environnements dont toutes les extensions sont conformes au monde réel.

La seconde option à l'avantage de donner au modèle un aspect plus homogène et élégant. Elle serait, par contre, en désaccord avec la définition de l'inconsistance, ce qui n'est pas le cas de l'option adoptée. Elle devrait par ailleurs être bien plus facilement modélisable que la première; l'essentiel du travail a été fait.

L'un ou l'autre de ces deux choix sont également utiles, mais au delà de ces simples choix, la solution qui s'est dessinée lors de la critique de l'implémentation est de ne plus considérer les environnements mais aussi leurs extensions. Ceci nécessite d'importants aménagements dans le modèle du chapitre II, mais il faut noter que, si les relations entre les environnements d'une part, et entre environnements et extensions d'autre part, sont conservées, il devrait être possible de disposer simultanément des deux interprétations.

3.2.Nommer les extensions

Bien sûr, au niveau du modèle, les problèmes d'extensions multiples peuvent être résolus comme cela a été indiqué au §II (descendre la barre des extensions en ajoutant des hypothèses). Cette solution est analogue à celle de Dressler qui déclare comme hypothèses les antécédents non monotones des inférences. Si elle est simple sur un plan conceptuel, elle ne l'est pas sur le plan de l'implémentation: les problèmes de prise en compte des extensions subsistent.

Mais ce type de solution n'est pas judicieux: il faut laisser à l'utilisateur le loisir de décider ce qui doit ou ne doit pas être hypothèse. Il risque, sinon, de se perdre dans un immense treillis d'environnements dont un grand nombre est sans signification.

L'approche qui consiste à pouvoir accéder aux extensions individuellement est le but à long terme du travail amorcé ici. À l'heure actuelle, il est uniquement possible de poser des jalons signalant les grandes étapes de ce travail.

Le problème principal à résoudre est celui du *nommage des extensions*, problème difficile et de longue haleine. L'ATMS était capable de nommer directement et de manière univoque toutes les interprétations du graphe à l'aide des environnements considérés. Cela est possible parce qu'il n'existe, dans son système, qu'une interprétation par environnement. Avec les inférences non monotones, apparaissent les extensions multiples et l'insuffisance des environnements pour nommer les interprétations.

Bien entendu, une solution possible est de construire toutes ces extensions et de les identifier par leur contenu (l'ensemble des formules qui sont présentes et de celles qui sont absentes). Mais l'idée des systèmes tels que ceux examinés ici est de considérer des bases de connaissance de taille conséquente. Cela conduit à prendre en compte des extensions de grande taille et c'est pourquoi ces systèmes ne font pratiquement jamais de saturation. Il n'y a, par conséquent, pas de possibilité de manipuler des extensions complètes. Les systèmes informatiques doivent donc trouver un moyen de *nommer* les extensions s'ils veulent les manipuler.

Le principe qui a présidé à la définition du modèle du chapitre II ne considère que des environnements fondés sur les hypothèses. Ces environnements sont incapables de rendre compte des extensions de manière individualisée. La sémantique du modèle reflète donc les extensions collectivement. Malheureusement, l'implémentation du système est assez ardue et une solution proposée pour l'améliorer consiste à mettre à jour les générateurs d'extensions et de manipuler celles-ci individuellement. Cela oblige à sortir du modèle.

Une solution envisageable pour nommer les extensions est donc celle qui consiste à leur donner le nom de ce qui leur a donné naissance. D'où l'importance du travail décrit plus haut pour isoler les générateurs d'extensions. Il suffirait de nommer les extensions par leurs générateurs et la façon dont ils sont configurés. Ainsi, à chaque choix s'offrant au système, celui-ci pourrait donner des noms aux différentes possibilités qui s'offrent à lui. Ces noms, conjugués aux environnements, seraient propagés dans le système, permettant ainsi d'individualiser les extensions. Les environnements devront donc être étendus pour pouvoir intégrer ces générateurs et leurs configurations.

Cela peut être comparé au travail mené en logique des défauts pour obtenir les extensions d'une théorie [Bonté& 89]. La théorie étant unique, il n'est pas besoin de noter l'ensemble des axiomes (ou hypothèses) considérés. Les «environnements» caractérisants les extensions (appelés univers) sont les ensembles de défauts (ou règles de défauts instanciées) qui doivent être utilisés pour obtenir l'extension. L'idée consiste à trouver les univers qui donnent lieu à des extensions. Ces univers sont alors de bons «noms» pour les extensions.

Par analogie, les noms des extensions pourront donc être les ensembles de justifications qui sont valides dans celles-ci. Il est possible de diminuer la taille des univers associés aux environnements en ne retenant que les justifications présentes dans les générateurs (définis en fonction de l'environnement considéré), c'est-à-dire celles susceptibles d'engendrer des extensions. En effet, nombre de justifications s'avèrent ne pas faire partie des générateurs ou faire partie de générateurs contraints par l'environnement.

Enfin, une autre approche peut être envisagée: basée sur le concept d'héritage, elle consiste à représenter explicitement contextes et extensions comme des entités, semblables aux objets dans les systèmes à base d'objets, qui héritent les unes des autres. Ce genre d'implémentation abandonnerait l'élégance de l'ATMS pour prendre beaucoup plus de place. C'est pourquoi il est à réserver pour un système de raisonnement ayant une stratégie claire de sélection des contextes dans lesquels il travaille. Cependant, cela permettrait de hiérarchiser la génération d'hypothèses comme le demandent de nombreuses applications, ce qui limite le nombre de contextes, et que l'ATMS ne pouvait faire jusqu'à présent.

Nommer les extensions permet de manipuler celles-ci individuellement ou collectivement. C'est donc un travail important qui devra être résolu par les travaux à venir.

4.Conclusion

La comparaison du CP-TMS avec les autres systèmes a permis de mettre en évidence l'étroite parenté qui existe entre tous les systèmes. Cependant, le CP-TMS souffre des limitations héritées du TMS dont il s'inspire. Résoudre ces problèmes conduit à tenter d'isoler dans les graphes de dépendances les configurations engendrant les extensions. Ce travail devra tout d'abord être mené au sein des systèmes de maintenance de la vérité à propagation avant d'être étendu. L'approche proposée remet en cause la structure des environnements tels qu'ils ont été définis dans le modèle du chapitre II. Il faudra étudier de nouveaux types d'environnements permettant de nommer les extensions multiples.

Conclusion

Conclusion

L'approche qui a été présentée dans ce travail consiste à rechercher des mécanismes capables de rendre service aux utilisateurs de systèmes de raisonnement. Le but est de fournir des algorithmes permettant à un système de produire un raisonnement hypothétique. C'est possible soit en autorisant les utilisateurs à poser eux-même les hypothèses soit en leur permettant de soumettre au système la connaissance sur la façon de poser des hypothèses. Cette dernière option conduit à un raisonnement non monotone.

Les programmes traditionnellement utilisés pour gérer les hypothèses sont des systèmes de maintenance de la vérité. Les systèmes proposés jusqu'alors excluaient l'une des deux options. Le travail présenté a permis de proposer un modèle, puis un programme dans lesquels elles coexistent

À partir du constat que les environnements proposés par les systèmes de maintenance de la vérité à contextes ne pouvaient prendre en compte des inférences non monotones, de nouveaux environnements capables le faire en considérant l'absence d'hypothèses ont été définis. Ces environnements se sont vu associer une interprétation qui a été étendue à l'ensemble des nœuds du graphe de dépendances, en accord avec l'interprétation des systèmes de maintenance de la vérité à propagation. Cela a permis d'établir la signification des étiquettes à associer aux nœuds du graphe, et de proposer de multiples possibilités d'interrogation d'un système de maintenance de la vérité correspondant.

Le CP-TMS a été implémenté en étendant un système de maintenance de la vérité à propagation de façon à ce qu'il puisse propager des environnements. La rétrogression est remplacée par une phase de rétablissement de la consistance substituant à chaque environnement présent dans une étiquette par les environnements plus spécifiques consistants. Le système propose l'ensemble des requêtes définies par le modèle. Cet algorithme souffre cependant du défaut de ne pas accepter les extensions multiples d'un environnement complet.

Quoi qu'il en soit, le CP-TMS, s'il n'implémente pas tout le modèle défini, est utilisable dès que le raisonnement considéré n'engendre pas d'extensions multiples. C'est le cas, par exemple, de la représentation de connaissance par objet SHIRKA [Rechenmann 85, &89] où les inférences se font par défaut et sont strictement ordonnées, et où aucune contrainte inter-attributs ne vient poser de problèmes de priorité. Un système à propagation, spécifiquement conçu, a déjà été introduit dans SHIRKA [Euzenat& 87]; il est utilisé dans une application d'étude de sites avalancheux. Dans le cadre du projet SHERPA, concernant la dynamique des grandes bases de connaissance, deux utilisations du CP-TMS sont envisagées:

- un système de gestion de versions, implémenté par le CP-TMS a été proposé [Euzenat 89d]. Il contraint à la déclaration des hypothèses sous-jacentes à la création d'une version.
- l'utilisation du CP-TMS pour la gestion des hypothèses concernant la classe d'appartenance d'un objet, les méthodes utilisées pour connaître les valeurs de ses attributs et les valeurs de ses attributs elles-mêmes est en cours d'étude.

Si un cadre formel a été défini, il n'a pas vocation universelle. Il serait donc très intéressant de jeter des ponts entre lui et les formalismes d'usage courant (logique des défauts, programmation en logique, logique auto-épistémique…). Cela devrait permettre de voir les implémentations de ce modèle comme des démonstrateurs de certaines classes de formalismes d'usage courant.

Pour résoudre les problèmes posés par une implémentation complète du modèle, il faudra poursuivre les recherches menées pour isoler, dans le graphe de dépendances, les configurations génératrices d'extensions multiples. Pouvoir référencer ces configurations devrait permettre de nommer les extensions multiples, de les manipuler et les interroger individuellement. Ainsi, le but aura été atteint permettant, à la fois, au système de raisonnement et à son utilisateur de poser des hypothèses et de pouvoir considérer individuellement chaque complétion de l'information disponible engendrée.

ANNEXES

Références et preuves

Références

Sigles

AAAI	American Association for Artificial Intelligence conference
ECAI	European Conference on Artificial Intelligence
IJCAI	International Joint Conference on Artificial Intelligence
RFIA	Congrès Reconnaissance des Formes et Intelligence Artificielle

[Apt& 87]
Krzysztof Apt, Jean-Marc Pugin
Management of stratified databases
Rapport technique CS-R-8761, CWI, Amsterdam (NL), 1987

[Billon 87]
Jean-Paul Billon
Perfect normal forms for discrete functions
Rapport de recherche DSG/CRG/87014, Centre de recherche du groupe Bull, Louveciennes (FR), 1987

[Bonté& 89]
Eric Bonté, François Lévy
Une procédure complète de calcul des extensions pour les théories de défauts en réseau
Actes 7ième RFIA, pp1315-1325, Paris (FR), 1989

[Brown& 87]
Allen Brown, Dale Gaucas, Dan Benanav
An algebraic foundation for truth maintenance
Actes 10ième IJCAI, pp973-980, Milan (IT), 1987

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

[Cordier 87]
Marie-Odile Cordier
Unification contextuelle et raisonnement hypothétique: le moteur d'inférence SHERLOCK.
Actes 6ième RFIA, pp787-797, Antibes (FR), 1987

[De Kleer 84]
Johan De Kleer
Choices without backtracking
Actes 4ième AAAI, pp79-85, Austin (TX US), 1984

[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 86c]
Johan De Kleer
Problem solving with the ATMS
Artificial intelligence 28-II, pp197-224, 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 78]
Jon Doyle
Truth maintenance systems for problem solving
Rapport technique AI-TR-419, MIT, Cambridge (MA US), 1978

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

[Dressler 87]
Oskar Dressler
An extended basic ATMS
Rapport de recherche INF-2 ARM-3-87, Siemens, München (DE) (rev. Extending the basic ATMS, actes 8ième ECAI, pp535-540, München (DE), 1988), 1987

[Dressler 88]
Oskar Dressler
An extended basic ATMS
Lecture notes on computer science (Lecture notes on artificial intelligence) 346 (Michael Reinfrank, Johan De Kleer, Matthew Ginsberg, Erik Sandewall (éds.), Non monotonic reasoning (actes 2nd international workshop, Grassau, DE, 1988)), pp144-163, 1989

[Euzenat& 87b]
Jérôme Euzenat, François Rechenmann
Maintenance de la vérité dans les systèmes à base de connaissance centrée-objet
Actes 6ième RFIA, pp1095-1109, Antibes (FR), 1987

[Euzenat 88]
Jérôme Euzenat
Un nouvel algorithme de maintenance de la vérité
Rapport technique, Cognitech, Paris (FR), 1988

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

[Euzenat 89b]
Jérôme Euzenat
Étendre le TMS (vers les contextes)
Actes 7ième RFIA, pp581-596, Paris (FR), 1989

[Euzenat 89c]
Jérôme Euzenat
Un algorithme de maintenance de la vérité tirant parti des composantes fortement connexes
Rapport interne, Laboratoire ARTEMIS, Grenoble (FR), 1989

[Euzenat 89d]
Jérôme Euzenat
Consistance du cache dans les grandes bases de connaissance par objets
Rapport interne, Laboratoire ARTEMIS, Grenoble (FR), 1989

[Fujiwara& 89]
Yasushi Fujiwara, Shinichi Honiden
Relating the TMS to autoepistemic logic
Actes 11ième IJCAI, pp1199-1205, Detroit (MI US), 1989

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

[Lipski 81]
Witold Lipski
On databases with incomplete information
Journal of the ACM 28-I, pp41-70, 1981

[Madre& 88]
Jean-Christophe Madre, Jean-Paul Billlon
Proving circuit correctness using formal comparison between expected and extracted behaviour
Actes 25ième design automation conference, pp205-210, Anaheim (CA US), 1988

[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

[Morris& 86]
Paul Morris, Robert Nado
Representing actions with a truth maintenance system
Actes 5th AAAI, pp13-17, Philadelphie (PA US), 1986

[Ostermann 88]
Pascal Ostermann
Interprétation de l'information incomplète en logique modale
Actes 4ièmes journées bases de données avancées, PRC BD3/INRIA, pp-279-296, Bénodet (FR), 1988

[Petrie 87]
Charles Petrie
Revised dependency directed backtracking for default reasoning
Actes 6ième AAAI, pp167-172, Seatle (WA US), 1987

[Pugin 88]
Jean-Marc Pugin-Marien
Contribution à l'étude des raisonnements non monotones: le tableur logique
Thèse, université Paris 7, Paris (FR), 1988

[Quintero 89]
Jose Alejandro Quintero-Garcia
Parallélisation de la maintenance de la vérité tirant parti des composantes fortement connexes
Rapport de DEA, INPG, Grenoble (FR), 1989

[Rechenmann 85]
François Rechenmann
SHIRKA : mécanismes d'inférence sur une base de connaissance centrée-objet
Actes 5ième RFIA, pp1243-1254, Grenoble (FR), 1985

[Rechenmann& 89]
François Rechenmann, Patrice Uvietta, Pierre Fontanille
SHIRKA, manuel d'utilisation
Rapport interne, Laboratoire ARTEMIS, Grenoble (FR), 1989

[Reiter& 87]
Raymond Reiter, Johan De Kleer
Foundations of assumption-based truth maintenance systems: preliminary report
Actes 6th AAAI, pp183-188, Seattle (WA US), 1987

[Úrbanski 87]
Andrej Úrbanski
Truth maintenance systems and their algorithms
Actes 2ième Cognitiva, pp262-266, Paris (FR), 1987

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

Preuves

Toutes les preuves de théorèmes sont regroupées ici. La plupart d'entre elles sont triviales et ne nécessitent que quelques lignes mais figurent ici par souci d'homogénéité.

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

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

Définition 2. Une valuation est dite une *valuation complète* si $H=\mathcal{H}$.

Définition 3. Une valuation ν est dite *plus complète* qu'une valuation ν' ssi $H' \subseteq H$ et $\forall h \in H, \nu'(h)=\nu(h)$. L'ensemble des complétions de ν est l'ensemble des environnements dont la valuation correspondante est plus complète que celle correspondant à ν:

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

Théorème 1:

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

preuve. immédiate

Définition 4. L'ensemble des complétions maximales de [A/R] est l'ensemble des complétions complètes de [A/R] :

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

Théorème 2:

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

preuve.

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

Lemme 1: $\exists C, \nu \in Cmax(C) \Rightarrow Cmax(\nu)=\{\nu\}$

preuve.

$[A/R] \in Cmax(C) \Rightarrow A \cup R = \mathcal{H}$ (définition 4)
$\Rightarrow \mathcal{H} \setminus (A \cup R) = \{\}$
$\Leftrightarrow Cmax([A/R]) = \{[A \cup A'/R \cup R']; A' \cup R' = \{\}\}$ (Théorème 2)
$\Leftrightarrow Cmax([A/R]) = \{[A/R]\}$

Lemme 2: $\nu \in Cmax(\nu) \Leftrightarrow Cmax(\nu)=\{\nu\}$

preuve.

$\Rightarrow$ (Lemme 1)
$\Leftarrow$ immédiat

Définition 5. Une valuation ν' est une *complétion de degré i* d'une valuation ν ssi $\nu' \in Comp(\nu)$ et $|H'|=i$. L'ensemble des complétions de degré i d'une valuation ν est définie par

$$Comp^i(\nu)=\{\nu' \in Comp(\nu); |A' \cup R'|=i\}$$

Conséquences: $\forall \nu$ valuation, $Comp^{|H|}(\nu)=\{\nu\}$ & $Comp^{|\mathcal{H}|}(\nu)=Cmax(\nu)$

Lemme 3: $\forall C \in Comp^i(\nu), Comp^j(C) \subseteq Comp^j(\nu)$

preuve.
$[A'/R'] \in Comp^i([A/R]) \Rightarrow [A'/R'] = [A \cup A''/R \cup R'']$
et donc
$Comp^j([A'/R']) = Comp^j([A \cup A''/R \cup R''])$
$= \{[A \cup A'' \cup A'''/R \cup R'' \cup R''']; |A \cup A'' \cup A'''/R \cup R'' \cup R'''| = j \}$
$\subseteq \{[A \cup A''''/R \cup R'''']; |A \cup A''''/R \cup R''''| = j \}$
$= Comp^j([A/R])$

Définition 6. L'ensemble des complétions de degré supérieur ou égal à i d'une valuation ν est dénoté par

$$Comp^{*i}(\nu) = \cup_{i \leq j \leq |\mathcal{H}|} Comp^j(\nu)$$

Lemme 4: $\forall C \in Comp^i(\nu), Comp(C) \subseteq Comp^{*i}(\nu)$

preuve.
$\forall C \in Comp^i(\nu)$,
$Comp(C) = Comp^i(C) \cup Comp^{*i+1}(C)$
$\subseteq Comp^i(C) \cup Comp^{*i+1}(\nu)$ (Lemme 3)
$= \{C\} \cup Comp^{*i+1}(\nu)$
$\subseteq Comp^{*i}(\nu)$

Définition 7. 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 $n \in H$ alors $I(n) = \nu(n)$
sinon, si $\exists j \in$ LISTEDEJUST[n]; $\forall h \in$ INLISTE[j] I(h)=1 & $\forall h \in$ OUTLISTE[j] I(h)=0
alors I(n)=1
sinon I(n)=0.

Définition 8. L'ensemble des interprétations construites à partir de la valuation [A/R] sera noté IntInd([A/R]).

Définition 9. À une interprétation correspond une *valuation interprétée* $\nu_I: \mathcal{H} \longrightarrow \{1,0\}$; $\nu_I(h) = I(h)$.

Conséquence: $\forall$ I interprétation, $I \in IntInd(\nu_I)$

Lemme 5: $IntInd(\nu) \subseteq \cup_{C \in Cmax(\nu)} IntInd(C)$

preuve.
$\forall I \in IntInd([A/R])\ \exists \nu_I = [A_I/R_I]$
$\forall a \in A$, I(a)=1 & $\forall r \in R$, I(r)=0 (définition 9)
$\Rightarrow$ $\forall a \in A$, $\nu_I(a)=1$ & $\forall r \in R$, $\nu_I(r)=0$ (définition 7)
$\Rightarrow$ $A \subseteq A_I$ & $R \subseteq R_I$
et $I \in IntInd(\nu_I)$

Définition 10. Une interprétation I est dite *soutenable* (respectivement *insoutenable*) si et seulement si $I(\bot)=0$ (respectivement $I(\bot)=1$).

Définition 11. Une valuation ν est dite *consistante* si toutes les interprétations construites à partir de ν sont soutenables. Si une valuation n'est pas consistante elle est dite *inconsistante*.

Définition 12. Une interprétation construite à partir d'une valuation [A/R] est dite *compatible* si $\forall h \in A$, $I(h)=1$ et $\forall h \in R$, $I(h)=0$.

Conséquence: Toute interprétation construite à partir d'une valuation [A/{}] est compatible.

preuve. Par la définition de l'interprétation (définition 7).

Définition 13. Une *complétion consistante* ν' d'une valuation ν est une valuation telle que $\forall h \in H$, $\nu'(h)=\nu(h)$, et $\forall I'$, interprétation à partir de ν', I' soit soutenable. L'ensemble des complétions consistantes de ν est caractérisé par

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

Définition 14. 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. Une complétion consistante minimale de ν est appelée une complétion consistante minimale de ν. L'ensemble des complétions consistantes minimales de ν est caractérisé par

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

Théorème 3: Si $\forall I$, interprétation à partir de ν, I est soutenable, alors l'ensemble des complétions consistantes minimales de ν est restreint à $\{\nu\}$. Il peut être formulé ainsi:

$$\forall I \in IntInd(\nu),\ I(\bot)=0 \Rightarrow Cmin(\nu)=\{\nu\}$$

preuve.

1) $\nu \in Comp(\nu)$ et $\forall I \in IntInd(\nu)$, $I(\bot)=0$
$\Rightarrow$ $\nu \in Ccons(\nu)$
2) $Cmin(\nu)$
$=$ $\{\nu' \in Ccons(\nu);\ \forall \nu'' \in Ccons(\nu),\ \nu' \in Ccons(\nu'') \Rightarrow \nu''=\nu'\}$
$=$ $\{\nu\} \cup \{\nu' \in Ccons(\nu) \setminus \{\nu\};\ \forall \nu'' \in Ccons(\nu),\ \nu' \notin Ccons(\nu'')\}$ (par 1)
or $\forall \nu' \in Ccons(\nu) \setminus \{\nu\}, \exists \nu''[=\nu] \in Ccons(\nu);\ \nu' \in Ccons(\nu'')$
$\Rightarrow$ $\{\nu' \in Ccons(\nu) \setminus \{\nu\};\ \forall \nu'' \in Ccons(\nu),\ \nu' \notin Ccons(\nu'')\} = \{\}$
$\Rightarrow$ $Cmin(\nu) = \{\nu\}$

Définition 15. Une étiquette d'un nœud n est définie comme l'ensemble des valuations dont toutes les complétions sont soutenables et ont au moins une interprétation qui considère n comme valide:

$$ETIQUETTE[n] = \{C\ ;\ \forall \nu \in Comp(C), \exists I \in IntInd(\nu);\ I(n)=1\ \&\ \forall I \in IntInd(\nu),\ I(\bot)=0\}$$

Définition 16. L'ensemble des formules dérivées de [A/R], c'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:

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

Lemme 6: $Ctxt(\nu) \subseteq \cap_{C \in Cmax(\nu)} Ctxt(C)$

preuve.

$n \in Ctxt(\nu) \Leftrightarrow \exists C;\ \nu \in Comp(C);\ C \in ETIQUETTE[n]$
or $\forall C;\ \nu \in Comp(C)$ et $\forall C' \in Cmax(\nu)$ on a $C' \in Comp(C)$
et donc
$\forall C' \in Cmax(\nu)\ \exists C;\ C' \in Cmax(C);\ C \in ETIQUETTE[n]$
$\Leftrightarrow \forall C' \in Cmax(\nu),\ n \in Ctxt(C')$

Définition 17. L'ensemble des formules nécessaires dans [A/R] est l'ensemble des nœuds qui doivent être valides dans toutes les interprétations construites à partir de [A/R]:

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

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

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

preuve.

i) $Nec(\diamond) \subseteq \cap_{C \in Cmax(\diamond)} Ctxt(C)$:

a) $Cmax(\diamond)=\{\diamond\}$

$n \in Nec(\diamond) \Leftrightarrow n \in Ctxt(\diamond) \Leftrightarrow n \in \cap_{C \in Cmax(\diamond)} Ctxt(C)$

b) $Cmax(\diamond) \neq \{\diamond\} \Rightarrow Cmax(\diamond) \subseteq Comp(\diamond) \setminus \{\diamond\}$ (contraposée du lemme 2)

. $n \in Ctxt(\diamond) \Rightarrow n \in \cap_{C \in Cmax(\diamond)} Ctxt(C)$ (lemme 6)

. $n \in \cap_{C \in Comp(\diamond) \setminus \{\diamond\}} Nec(C)$

$\Rightarrow n \in \cap_{C \in Cmax(\diamond)} Nec(C)$

or $C \in Cmax(\diamond) \Rightarrow C \in Cmax(C)$ (par lemme 2) et donc $Nec(C)=Ctxt(C)$

$\Rightarrow n \in \cap_{C \in Cmax(\diamond)} Ctxt(C)$

ii) $Nec(\diamond) \supseteq \cap_{C \in Cmax(\diamond)} Ctxt(C)$:[8]

pas 0)

$\forall C \in Comp^{|\mathfrak{R}|}(\diamond)$,

$Cmax(C)=\{C\}=Comp(C)$

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

$= Ctxt(C)$

$= \cap_{C' \in Cmax(C)} Ctxt(C')$

pas d'induction)

$\forall C \in Comp^{i}(\diamond)$,

$\forall C' \in Comp^{*i+1}(\diamond)$, $Nec(C')=\cap_{C'' \in Cmax(C')} Ctxt(C'')$ (hypothèse d'induction)

alors $Nec(C)$

$= Ctxt(C) \cup \cap_{C' \in Comp(C) \setminus \{C\}} Nec(C')$ (définition 17)

$= Ctxt(C) \cup \cap_{C' \in Comp^{*i+1}(C)} Nec(C')$ (définition 6)

$= Ctxt(C) \cup \cap_{C' \in Comp^{*i+1}(C)} \cap_{C'' \in Cmax(C')} Ctxt(C'')$ (hypothèse d'induction)

$= Ctxt(C) \cup \cap_{C' \in Cmax(C)} \cap_{C'' \in Cmax(C')} Ctxt(C'')$

car $\forall C' \in Comp^{*i+1}(C)$, $Cmax(C') \subseteq Cmax(C)$ (lemme 3, $j=|\mathfrak{R}|$)

$= Ctxt(C) \cup \cap_{C' \in Cmax(C)} Ctxt(C')$

$= \cap_{C' \in Cmax(C)} Ctxt(C')$ (lemme 6)

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

preuve.

$n \in Nec(\diamond)$

$\Leftrightarrow$ $\forall C \in Cmax(\diamond)$, $n \in Ctxt(C)$ (théorème 4)

$\Leftrightarrow$ $\forall C \in Cmax(\diamond)$, $\exists C'$; $C \in Comp(C')$; $C' \in$ ETIQUETTE[n] (définition 16)

$\Leftrightarrow$ $\forall [A/R] \in Cmax(\diamond)$, $\exists A' \subseteq A$, $R' \subseteq R$; $[A'/R'] \in$ ETIQUETTE[n] (définition 3)

$\Leftrightarrow$ $\forall C \in Cmax(\diamond)$, $\exists I \in IntInd(C)$, $I(n)=1$

Définition 18. 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: $Pos(\diamond) = \cup_{C \in Cmax(\diamond)} Ctxt(C)$

preuve.

$Pos(\diamond)$

$= \cup_{C \in Comp(\diamond)} Ctxt(C)$ (définition 18)

$= \cup_{C \in Comp(\diamond)} \cup_{C'; C \in Comp(C')} \{n \in \mathbb{N}; C' \in$ ETIQUETTE[n]$\}$ (définition 16)

$= \cup_{C \in Cmax(\diamond)} \cup_{C'; C \in Comp(C')} \{n \in \mathbb{N}; C' \in$ ETIQUETTE[n]$\}$

Car $\forall C \in Comp(\diamond) \setminus Cmax(\diamond)$ $\forall C'' \in Cmax(C)$

$C'' \in Cmax(\diamond)$ et $\{C'; C \in Comp(C')\} \subseteq \{C'; C'' \in Comp(C')\}$

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

---

[8] Recherche désespérément démonstration plus élégante (ne faisant pas intervenir les $Comp^i$). À noter que cette démonstration de la seconde partie du théorème est suffisante pour démontrer tout le théorème.

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

preuve.

$n \in Pos(\diamond)$
$\Leftrightarrow$ $\exists \diamond' \in Cmax(\diamond); n \in Ctxt(\diamond')$ (théorème 6)
$\Leftrightarrow$ $\exists \diamond' \in Cmax(\diamond); \exists \diamond'' \in Cmax(\diamond'); \diamond'' \in ETIQUETTE[n]$ (définition 16)
$\Leftrightarrow$ $\exists \diamond' \in Cmax(\diamond); \diamond' \in ETIQUETTE[n]$
$\Leftrightarrow$ $\exists \diamond' \in Cmax(\diamond); \forall \diamond''' \in Comp(\diamond'); \exists I \in IntInd(\diamond'''), I(n)=1$ (définition 15)
$\Leftrightarrow$ $\exists \diamond' \in Cmax(\diamond); \exists I \in IntInd(\diamond'), I(n)=1$

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

preuve.

$Ctxt(\diamond) \subseteq Nec(\diamond)$: par définition.
$Nec(\diamond) \subseteq Pos(\diamond)$: s'appuie sur le fait que si $\forall x \in X, \exists y \in Y; x \subseteq y$ alors $\cap Y \subseteq \cup X$.

Théorème 9:
$Ccons([A/R]) =$
$\{[A'/R']; A \subseteq A', R \subseteq R', A' \cap R' = \{\}$
$\& \forall [A''/R'] \in ETIQUETTE[\bot]; A \subseteq A'' \& R \subseteq R'' \Rightarrow (A' \cap R'') \cup (A'' \cap R') \neq \{\}\}$

preuve.

$\{[A'/R']; A \subseteq A', R \subseteq R', A' \cap R' = \{\}$
$\& \forall [A''/R''] \in ETIQUETTE[\bot]; A \subseteq A'' \& R \subseteq R'' \Rightarrow (A' \cap R'') \cup (A'' \cap R') \neq \{\}\}$
$= \{[A'/R'] \in Comp([A/R]);$
$\forall [A''/R''] \in ETIQUETTE[\bot], [A'/R'] \notin Comp([A''/R'']) \}$ (définition 1)
$= \{[A'/R'] \in Comp([A/R]);$
$\neg(\exists I \in IntInd([A'/R']); I(\bot)=1) \}$ (définition 15)
$= \{[A'/R'] \in Comp([A/R]); \forall I \in IntInd([A'/R']), I(\bot)=0 \}$ (définition 7)
$= Ccons([A/R])$ (définition 13)

Index

Les numéros de page associés aux entrées sont ceux correspondant à la définition de celles-ci.