Coopération consistante de bases de connaissance au premier ordre fini

L. Brisoux

CRIL-Université d'Artois
rue de l'Université SP16
F-62307 Lens Cedex, France
brisoux@cril.univ-artois.fr

Mots clés : Combinaison de bases de connaissance, inconsistance locale, agents rationnels, systèmes coopératifs.

Résumé

Dans ce papier, nous nous intéressons à la formalisation et à l'implantation de bases de connaissance coopératives et plus précisément à la préservation de la consistance logique lors de leur fusion ou de leur mise en interaction. Des bases de connaissance consistantes individuellement peuvent être inconsistantes globalement. Ceci constitue un problème d'autant plus primordial d'un point de vue déductif que l'inconsistance logique est globale en logique standard : tout (et son contraire) peut être déduit depuis tout ensemble d'informations contradictoires, même le plus petit. Dans un cadre propositionnel, des heuristiques sur la trace du travail effectué par des algorithmes de recherche locale permettent de localiser les informations contradictoires. Mais comment les méthodes de recherche locale peuvent être appliquées aux bases de connaissance, exprimées au sein de langages plus expressifs, notamment au premier ordre fini, sans entraîner une explosion combinatoire de l'espace de recherche par instanciation complète de la base, demeure un problème ouvert. Dans ce papier, nous proposons une méthode d'instanciation partielle permettant le développement d'un nouveau concept de consistance/inconsistance incrémentale, applicable aux bases de connaissance en interaction.

1 Introduction

Dans ce papier, nous nous intéressons à la formalisation et à l'implantation de bases de connaissance coopératives et plus précisément à la préservation de la consistance logique de leur fusion ou de leur mise en interaction. Le concept d'agents rationnels est le sujet de recherches variées en Intelligence Artificielle, conduisant à différentes notions de systèmes coopératifs intelligents (voir (Gasser et Huhns, 1989; Maes, 1990; Shoham, 1993; Singh, 1991; Huhns, 1987))

Ici, les agents sont assimilés à des éléments logiciels d'intelligence artificielle encodant de la connaissance et capables d'inférence. La connaissance de chaque agent est codée dans une base de connaissance (BC) exprimée au sein du langage de la logique du premier ordre fini.

Supposons que nous mettions en commun les connaissances de plusieurs de ces agents, la connaissance de chaque agent étant consistante. La connaissance résultante peut être inconsistante. Soulignons que déterminer la consistance d'une formule propositionnelle est un problème NP-complet (Cook, 1971), ce qui signifie qu'un algorithme résolvant cette question mettrait un temps exponentiel dans le pire des cas, à moins que $P = NP$. Cependant des techniques efficaces ont été développées pour des bases de connaissance exprimées en logique propositionnelle (Mazure et al., 1997a). Récemment, des méthodes de recherche locale ont en effet prouvé leur efficacité en résolvant cette question pour de larges gammes de très grandes bases de connaissance propositionnelles (Selman et al., 1992) (Mazure et al., 1997b).

Étendre ces techniques à des langages de représentation plus expressifs, notamment à des langages expressivement équivalents au premier orde fini, demeure une question ouverte. Une extension immédiate, mais impraticable, consisterait à instancier complètement la base initialement au premier ordre. Pour éviter une explosion combinatoire en espace due à pareille instanciation « totale » de la base de connaissance, nous proposons un schéma d'instanciation partielle conduisant à un concept de consistance/inconsistance incrémentale, lequel peut être formulé comme suit.

Soit BC une base de connaissance consistante et k de la connaissance (consistante) supplémentaire que nous voulons introduire dans BC. Dans la suite, nous étudions dans quelle mesure la recherche locale peut nous aider à prouver si $BC+ =_{def} BC \cup \{k\}$ est consistant ou non.

Une caractéristique essentielle de notre approche est que nous ne restreignons pas notre langage aux clauses de Horn mais traitons toutes clauses (sans fonction) de la logique du premier ordre multi-sortée, sous ses interprétations de Herbrand.

Dans la prochaine section nous ferons une brève description du cadre de travail et de ses motivations, puis exposerons comment les techniques de recherche locale peuvent être utilisées pour tester la consistance des bases de connaissance propositionnelles BCs. Ensuite nous introduirons et expliquerons le concept d'instanciation partielle et nous donnerons quelques résultats expérimentaux.

2 Rappels préliminaires

Par la suite BC sera considéré comme un ensemble fini de clauses du premier ordre sans fonction et multi-sorté. Nous nous restreindrons au domaine d'Herbrand, cette hypothèse étant naturelle dans le cadre de la formalisation logique d'agents rationnels.

```
Procédure GSAT
entrée: S ensemble de clauses, MAX-FLIPS et MAX-TRIES
sortie: un modèle de S s'il existe
Début
     pour i := 1 à MAX-TRIES
         I := affectation des variables aléatoirement
         pour j := 1 à MAX-FLIPS
             si I modèle de S alors retourne I
             x := variable propositionnelle telle que son changement de signe donne
             le meilleur score (éventuellement négatif en nombre de clauses satisfaites
             dans S par I)
             I := I avec la valeur x inversée
         fin pour
     fin pour
retourne « aucun modèle trouvé »
fin
```

FIG. 1 – *Procédure GSAT*

Soit un ensemble fini **P** de prédicats d'arité finie, **V** l'ensemble des variables et **T** l'ensemble (non vide) des constantes. Les symboles de fonctions non constantes ne sont pas considérés ici. Un prédicat P appliqué à ses arguments est appelé un *atome*. Un littéral est un atome positif ou négatif. On représentera un atome formé du prédicat P et de ses arguments par $P()$ lorsqu'il ne sera pas nécessaire de représenter ses arguments explicitement.

Une clause est une disjonction de littéraux de la forme $P_1() \vee \ldots \vee P_n() \vee \neg Q_1() \vee \ldots \vee \neg Q_m()$ où chaque variable est supposée être quantifiée universellement sur son domaine. Une clause est dite close si elle ne contient aucune variable.

Par la suite BC sera considéré comme un ensemble fini de clauses et de formules existentielles de la forme : $\exists x_1, \ldots, \exists x_n P_1() \vee \ldots \vee P_n() \vee \neg Q_1() \vee \ldots \vee \neg Q_m()$ (où $x_1, \ldots, x_n$ seront les seules variables de la formule).

En logique propositionnelle, une interprétation est l'affectation des variables propositionnelles aux valeurs *vrai* ou *faux*. Un modèle de BC est une interprétation qui associe BC à *vrai*. Dans le cas du premier ordre, une interprétation est définie, sur les domaines d'interprétations, par rapport aux fonctions d'évaluation pour les variables et fonctions d'interprétation, permettant d'interpréter alors les clauses comme *vraies* ou *fausses*. On ne considère que les interprétations d'Herbrand. L'idée intuitive est que le domaine d'interprétation de BC peut être réduit aux constantes apparaissant dans BC.

La logique est multi-sortée : chaque i$^{\text{ème}}$ argument de chaque prédicat P de **P** peut être une variable ou une constante. Dans ce cas la constante doit appartenir à un sous-ensemble $\mathbf{T}_{pi}$ de **T**. La base d'Herbrand est définie, modulo ces contraintes, comme l'ensemble de tous les atomes clos du langage. Une interprétation d'Herbrand de BC sera un sous-ensemble de cette base. Un modèle d'Herbrand de BC est une interprétation d'Herbrand qui satisfait BC (qui interprète BC comme *vrai*). Nous dirons donc que BC est consistant si et seulement s'il existe au moins une interprétation d'Herbrand qui la satisfasse.

3 La recherche locale pour la preuve de la consistance ou de l'inconsistance

Faisons quelques rappels sur l'utilisation de la recherche locale dans notre domaine. Cette technique permet de détecter, la plupart du temps, des noyaux inconsistants dans des bases de connaissance. D'abord, nous appliquons une des méthodes de recherche locale les plus représentatives comme l'algorithme de Selman et co-auteurs : GSAT (Selman et al., 1992; Selman et al., 1993) (voir fig 1). Nous utiliserons une variante de cette plate-forme : TSAT (Mazure et al., 1997b).

Il a été découvert que la trace de l'algorithme de recherche locale pouvait mettre en évidence, très souvent, un petit noyau inconsistant, lorsque cette recherche de modèle aboutissait à un échec (Mazure et al., 1996). Cette méthode est basée sur le nombre de fois qu'une clause (resp. qu'un littéral) a été falsifiée (resp. est présent dans la clause falsifiée) lors de la recherche. Très souvent lorsque la recherche locale échoue c'est par la faute d'un petit groupe de clauses, celles qui ont été le plus souvent falsifiées, qui forment un petit noyau inconsistant.

Bien sûr la validité de ce résultat dépend de la taille de l'éventuel noyau inconsistant et de la taille de BC. Lorsque BC tend à être globalement inconsistant (la taille du noyau inconsistant tend vers la taille de BC) cette technique est inefficace. Heureusement l'inconsistance est souvent due à un nombre limité de règles et de faits contradictoires.

4 « Niveau-limité » pour l'inconsistance et la consistance

Étendons cette technique dans le cadre du problème de la consistance/inconsistance incrémentale au premier ordre fini. La méthode la plus simple serait d'instancier

```
étudiant(x) ⇒ (étudie(anglais, x) ∨ étudie(français, x))
3eme_année(x) ∧ étudiant(x) ⇒ (étudie(latin, x) ∨ étudie(grec_ancien, x))
3eme_année(x) ∧ étudiant(x) ⇒ (étudie(grec_ancien, x) ∨ veut_apprendre(latin, x))
¬étudie(grec_ancien, x) ∨  étudie(français, x)
langue_moderne(x)  ⇒ langue(x)
langue_ancienne(x)  ⇒ langue(x)
étudiant(André)
étudiant(Christine)
étudiant(Pierre)
3eme_année(André)
3eme_année(Christine)
langue_moderne(anglais)
langue_moderne(français)
langue_ancienne(latin)
langue_ancienne(grec_ancien)
étudie(anglais,André) ∨ étudie(anglais,Pierre) ∨ étudie(anglais,Christine) ∨ étudie(français,André)
¬étudie(anglais,Christine)  ∨ ¬étudie(anglais, André)
¬étudie(anglais,Christine)  ∨ ¬étudie(anglais, Pierre)
¬étudie(anglais,Christine)  ∨ ¬étudie(all, André)
¬étudie(français, André)  ∨ ¬étudie(anglais, André)
étudie(anglais, André) ∨ étudie(anglais, Christine)
étudie(anglais, Pierre) ∨ étudie(anglais, Christine)
étudie(français, André) ∨ étudie(anglais, Christine)
étudie(français, André) ∨ étudie(anglais, André)
veut_apprendre(anglais,Christine) ∨ veut_apprendre(français, Christine)
veut_apprendre(français, André)
veut_apprendre(anglais, Pierre)
veut_apprendre(français, Pierre)
veut_apprendre(latin, Christine) ∨ veut_apprendre(grec_ancien, Christine)
veut_apprendre(latin, André)
```

FIG. 2 – *Exemple de base de connaissance en premier ordre fini*

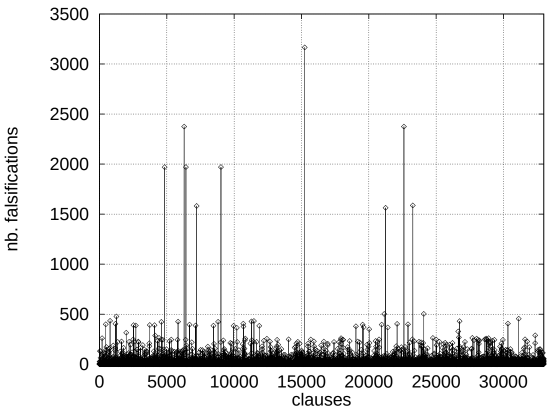

FIG. 3 – *Trace de TSAT sur BC_{end}*

complètement BC ce qui provoquerait une explosion combinatoire en espace. Nous avons donc développé une technique à *niveau-limité* qui correspond à une instanciation partielle. Soit BC une base de connaissance consistante et k une autre base consistante que l'on désire inclure dans BC, BC et k contenant au moins une constante.

Soient $BC+ =_{def} BC \cup \{k\}$ et $\mathbf{T}_0$ l'ensemble des constantes apparaissant dans k. Soit BC_0 contenant tous les termes clos de $BC+$ ainsi que tous les termes clos obtenus par instanciation de toutes les formules existentielles de $BC+$

Le niveau-i de consistance/inconsistance est défini comme suit :

Base

– soit $\mathbf{C}_1$ l'ensemble de clauses de $BC+$ contenant au moins un prédicat apparaissant dans k.
  Soit $\mathbf{T}_1 =_{def} \mathbf{T}_0 \cup \{$*constantes apparaissant dans* $\mathbf{C}_1\}$. Soit $BC_1 =_{def} BC_0 \cup \{$*les termes clos obtenus en instanciant les clauses de* $\mathbf{C}_1$ *en utilisant* $\mathbf{T}_1$ *à la place de* $\mathbf{T}$ *comme domaine d'instanciation*$\}$. Nous avons :
  BC est de niveau-1 consistant (resp. inconsistant) ssi BC_1 est consistant (resp. inconsistant)

le niveau-i de consistance ou d'inconsistance est défini de la manière suivante lorsque $i > 1$.

Induction

– soit $\mathbf{C}_i$ l'ensemble de clauses de $BC+$ contenant au moins un prédicat apparaissant dans $\mathbf{C}_{i-1}$.
  Soit $\mathbf{T}_i =_{def} \mathbf{T}_{i-1} \cup \{$*constantes apparaissant dans* $\mathbf{C}_i\}$. Soit $BC_i =_{def} BC_{i-1} \cup \{$*les termes clos obtenus en instanciant les clauses de* $\mathbf{C}_i$ *en utilisant* $\mathbf{T}_i$ *à la place de* $\mathbf{T}$ *comme domaine d'instanciation*$\}$. Nous avons :
  BC est de niveau-i consistant (resp. inconsistant) ssi BC_i est consistant (resp. inconsistant)

Lemme 1 *$\exists n$ tel que BC_n est stable, c-à-d $BC_n = BC_{n+1}$ dans ce cas BC_n est noté BC_{end}.*

Théorème 1 *Dans le cas d'une logique mono-sortée $BC+$ est consistant (resp. inconsistant) ssi BC_{end} est consistant (resp. inconsistant).*

Ce résultat peut être étendu facilement dans le cas d'un logique multi-sortée, sous la condition qu'il y ait assez de constantes dans k. Il est possible également d'introduire des fonctions non constantes mais uniquement dans le cas de domaines finis.

5 Utilisation de la recherche locale sur les bases de connaissance instanciées : résultats expérimentaux

Nous avons testé notre méthode sur une large base de connaissance $BC+$ exprimée en premier ordre. Il s'agit de la fusion entre 765 clauses du premier ordre (du problème consistant TPTP SYN427 (TPTP), utilisant 108 symboles de prédicats) avec le difficile problème « fact7h » proposé par Kautz et Selman (Kautz et Selman, 1994), celui-ci formalise un problème de planification contenant 32952 clauses propositionnelles et 2595 variables, et la base inconsistante (non Horn) exprimée en premier ordre adaptée de (Bossu et Siegel, 1981) (voir figure 2). Dès que l'instanciation converge vers un BC_{end} stable, nous obtenons une base d'environ 3300 clauses. L'exécution de TSAT (avec l'option de poids (Morris, 1993) (Selman et al., 1993)) dure 5 secondes et donne le résultat de la figure 3. Les neuf clauses les plus souvent falsifiées nous montrent le noyau inconsistant. Une remarque importante : si le problème initial partage les même prédicats et constantes, la trace de TSAT sera « ordonnée », permettant à une méthode mêlant la recherche locale et les méthodes complètes de donner une preuve rapide de l'inconsistance (Mazure et al., 1998).

6 Conclusion

Tester la consistance d'une base de connaissance est un problème crucial en Intelligence Artificielle, plus encore dans le contexte d'agents distribués et de systèmes coopératifs, puisque la connaissance globale peut être inconsistante. De bonnes méthodes ont été proposées lorsque la base est exprimée en logique propositionnelle. Adapter ces techniques aux bases du premier ordre n'est pas facile mais c'est une voie prometteuse. Dans ce papier, nous avons développé une méthode d'instanciation partielle. Elle permet une une construction de la consistance/inconsistance incrémentale du problème grâce à la méthode de consistance/inconstance à niveau-limité .

7 Remerciements

Ce travail a été financé par le projet *Ganymède II* du contrat de Plan Etat/Nord-Pas-de-Calais. Nous remercions nos collègues du CRIL pour leurs discussions enrichissantes sur ce sujet. En particulier B. Mazure pour son aide précieuse sur la partie expérimentation.

Références

BOSSU, G. et SIEGEL, P. (1981). « *La saturation au secours de la non-monotonie* ». PhD thesis, Université d'Aix-Marseille II. Thèse d'Université.

BRISOUX, L., SAÏS, L., et GRÉGOIRE, E. (1998). « Validation of Knowledge-Based Sytems by Means of Stochastic Search ». Dans *Proc. on the First International Workshop on Verification, Validation and Integrity Issue in Expert and Database Systems*, Vienna. IEEE Press.

COOK, S. (1971). « The complexity of theorem proving procedures ». Dans *Proc. of Third Annual ACM Symp. on Th. of Computing*, pages 151–158.

GASSER, L. et HUHNS, M. (1989). *Distributed Artificial Intelligence*, volume 2. Morgan Kaufmann.

HUHNS, M. (1987). *Distributed Artificial Intelligence.* Morgan Kauffman.

KAUTZ, H. et SELMAN, B. (1994). « Pushing the envelope: planning, propositional logic, and stochastic search ». Dans *Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI'94).*

MAES, P. (1990). *Designing Autonomous Agents.* MIT Press.

MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1996). « Detecting logical inconsistencies ». Dans *Symposium on AI/Math*, pages 116–121.

MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1997a). « An efficient technique to ensure the logical consistency of interacting knowledge bases ». *International Journal of Cooperative Information Systems*, 6(1):27–36.

MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1997b). « Tabu search for SAT ». Dans *Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 281–285, Rhodes Island.

MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1998). « Boosting complete technique thanks to local search methods ». Dans *Annals of Mathematics and Artificial Intelligence.*

MORRIS, P. (1993). « The breakout method for escaping from local minima ». Dans *Proc. of the Eleventh National Conference on Artificial Intelligence (AAAI'93).*

SELMAN, B., KAUTZ, H., et COHEN, B. (1993). « Local Search Strategies for Satifiability Testing ». Dans *Proc. 1993 DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability.*

SELMAN, B., MITCHELL, D., et LEVESQUE, H. (1992). « A New Method for Solving Hard Satisfiability Problems ». Dans *Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92).*

SHOHAM, Y. (1993). « Agent-oriented programming ». Dans *Journal of Artificial Intelligence*, volume 60(1), pages 51–92.

SINGH, M. P. (1991). « Toward a formal theory of communication for multi-agent systems ». Dans *Proc. of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91).*

TPTP. « TPTP: Thousands of Problems for Theorem Provers ». Benchmarks for first-order theorem proving available at `http://www.cs.jcu.edu.au/~tptp`.