Résolution de formules booléennes quantifiées : problèmes et algorithmes

Sylvie Coste-Marquis[1] Hélène Fargier[2] Jérôme Lang[2] Daniel Le Berre[1*] Pierre Marquis[1]

[1] Centre de Recherche en Informatique de Lens
SP 16 - Rue de l'Université - Université d'Artois - 62300 Lens
{coste,leberre,marquis}@cril.univ-artois.fr

[2] Institut de Recherche en Informatique de Toulouse
118 route de Narbonne - Université Paul Sabatier - 31062 Toulouse
{fargier,lang}@irit.fr

Résumé

Nous étudions les formules booléennes quantifiées (QBF) sous différents angles. Tout d'abord, nous montrons qu'en pratique, la résolution du simple problème de décision associé à une QBF ne suffit pas, et nous proposons deux problèmes de recherche associés à une QBF, dont les sorties attendues sont dans le premier cas une politique solution, dans le second cas une politique saine maximale. Puis nous nous occupons de la représentation de ces politiques : puisque celles-ci peuvent être exponentiellement grandes, il est important d'en élaborer des modes de représentation implicites, et nous proposons une telle approche, fondée sur les diagrammes de décision binaire (BDD). Enfin, nous évoquons la version "optimisation" des problèmes QBF.

Mots Clef

Contraintes (CSP, SAT, ordonnancement), Algorithmique de l'IA ou de la RF, Planification, QBF, Complexité.

Abstract

We study Quantified Boolean Formulae from different points of view. We first point out that from the practical side, solving only the decision problem associated to a QBF is not sufficient. Then, we propose two function problems associated to a QBF : one finds a solution policy, the other one a sound and maximal decision policy. In addition, we investigate the representation of these policies : as they may be exponentially large, finding implicit representations is necessary. We propose a BDD-based representation for such policies. Finally, the optimization side of QBF is sketched.

*Ce travail a été réalisé lorsque l'auteur travaillait au Business & Technology Research Laboratory, The University of Newcastle, Australia.

Keywords

Constraints (CSP, SAT, scheduling), Algorithms for AI and PR, Planning, QBF, Complexity.

1 Introduction

1.1 Motivations

La résolution de problèmes de *formules booléennes quantifiées* (QBF) est, depuis quelques années, un sujet d'importance en logique propositionnelle : l'intérêt grandissant des industriels pour la vérification formelle basée sur le raisonnement propositionnel en est l'une des explications.

Une formule booléenne quantifiée consiste informellement en une formule propositionnelle classique à laquelle on associe une partition ordonnée de l'ensemble de ses variables, correspondant à des alternances de quantificateurs. Ainsi, si Φ est une formule propositionnelle sur $\{a, b, c, d\}$, $(\exists\{a\})(\forall\{b, d\})(\exists\{c\})\Phi$ est une QBF. Une QBF est dite *positive* si l'énoncé correspondant, où les quantificateurs sur des variables portent en fait sur les *valeurs de vérité* de ces variables, est valide : ainsi, la QBF précédente est positive s'il existe une affectation (à vrai ou à faux) de la variable a telle que pour toute affectation des variables b et d, il existe une affectation de c telle que Φ soit vérifiée. On voit tout de suite que le problème SAT est un problème QBF particulier, obtenu en n'autorisant qu'une seule classe de variables, existentiellement quantifiées (ainsi, la formule Φ précédente est satisfaisable si et seulement si $(\exists\{a, b, c, d\}\Phi)$ est une instance positive de QBF).

Historiquement, les formules booléennes quantifiées ont été introduites par Stockmeyer [21] de façon conjointe avec la hiérarchie polynomiale, bien connue en théorie de la complexité. Il y a une excellente raison pour cela : le problème QBF est le problème cano-

nique complet pour la classe de complexité PSPACE, et les problèmes $\text{QBF}_{k,\exists}$, $\text{QBF}_{k,\forall}$ (dont les définitions sont rappelées plus loin) sont les problèmes canoniques complets pour les classes Σ_k^p et Π_k^p de la hiérarchie polynomiale.

Bien plus récemment, la résolution pratique de problèmes QBF est devenu un champ d'investigation important en intelligence artificielle. On peut se demander pourquoi il s'est écoulé 20 ans entre l'introduction de ces problèmes et l'intérêt pour leur résolution. On peut avancer deux types de raison à cela.

La première est d'ordre technique : la résolution de problèmes QBF est bien plus complexe que la résolution du problème SAT, lequel est déjà difficile. Or, les progrès dans la résolution de ce dernier au cours des cinq dernières années ont été extrêmement significatifs. C'est donc très naturellement que les chercheurs se sont alors intéressés à la résolution de problèmes plus généraux et plus complexes.

La seconde est d'ordre pratique. Au cours de années 90, on a identifié la complexité de beaucoup de problèmes majeurs en intelligence artificielle, et il se trouve qu'une bonne partie de ces problèmes se situe au deuxième ou au troisième niveau de la hiérarchie polynomiale (de nombreux problèmes d'inférence dans les logiques non monotones, de révision et de mise-à-jour des croyances, d'inférence tolérante à l'incohérence, de raisonnement sur l'action, d'abduction, de diagnostic, de décision dans l'incertain), et une autre bonne partie au niveau de la PSPACE-complétude (en particulier, de nombreux problèmes de planification). Or, les problèmes QBF sont les problèmes canoniques pour ces classes de complexité. Être capable de les résoudre constitue donc une avancée importante pour la résolution pratique des problèmes que nous venons d'évoquer.

Par ailleurs, l'intelligence artificielle n'est pas la seule discipline pour laquelle les problèmes QBF ont un intérêt. Ils en ont un en logique mathématique, puisque le problème de la satisfaisabilité dans de nombreuses logiques modales est PSPACE-complet. Ils en ont également un en vérification formelle : trouver une anomalie dans une conception peut s'exprimer en terme d'un problème d'accessibilité dans un automate, ou encore d'un problème de planification. Enfin, la proximité évidente de QBF avec les jeux séquentiels à deux joueurs et à information complète, ainsi qu'avec les jeux contre la nature, suggère encore un champ d'application évident. En effet, une façon naturelle d'interpréter un problème QBF en pratique est bien en terme de jeu. Par exemple, $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ représente un jeu en quatre coups, à deux joueurs, dénotés $J_\forall$ et $J_\exists$, jouant alternativement en affectant une valeur de vérité à une variable propositionnelle : $J_\exists$ commence en affectant une valeur à a, puis $J_\forall$ joue en affectant une valeur à b, puis $J_\exists$ affecte une valeur à c, et enfin $J_\forall$ termine le jeu en affectant une valeur à d. Le but de $J_\exists$ est que la formule Φ (formée sur $\{a,b,c,d\}$) soit satisfaite à l'issue du jeu ; ainsi, $\exists\{a\}\forall\{b\}\exists\{c\}\forall\{d\}\Phi$ est une instance positive de QBF si et seulement si il existe une stratégie gagnante pour $J_\exists$. On peut classer ici les problèmes de décision séquentielle (ou planification) dans l'incertain, qui peuvent etre compris comme des jeux contre la nature : les affectations de variables universellement quantifiées correspondent alors aux "coups" joués par la nature, et les stratégies gagnantes sont les politiques de décision (les plans conditionnels). Lorsqu'un seul quantificateur existentiel apparaît dans la formule, on a affaire à des problèmes de décision sous incertitude non séquentiels [10]. En particulier, dans les formules booléennes quantifiées du deuxième niveau (de la forme $\langle 2,\exists,X,Y,\Phi\rangle$ et de la forme $\langle 2,\forall,Y,X,\Phi\rangle$), les variables existentiellement quantifiées (X) représentent les variables de décision et les variables universellement quantifiées (Y) représentent les variables d'état (non observables) du problème. La forme de Φ peut être plus ou moins complexe, et dans la forme la plus simple, on peut considérer que Φ représente un ensemble de buts, d'objectifs à satisfaire. $\langle 2,\forall,Y,X,\Phi\rangle$ représente donc les problèmes de décision sous observabilité totale (il faut une décision pour chaque observation possible des variables d'état) et inversement, $\langle 2,\exists,X,Y,\Phi\rangle$ correspond aux cas de totale inobservabilité des variables d'état (il faut une décision qui soit satisfaisante quelle que soit la configuration des variables d'état). On représente au troisième niveau les problèmes de décision non séquentiels partiellement observables, par une formule de la forme $\langle 3,\forall,Y_1,X,Y_2,\Phi\rangle : Y_1$ (resp. Y_2) correspond aux variables d'état observables (resp. non observables).

À la lumière de ces interprétations des problèmes QBF, on voit que ce que l'on attend de la *résolution* d'un tel problème est souvent davantage qu'une simple résolution du problème de décision qui consiste à déterminer si une instance est positive ou non. En effet, la résolution du problème de décision ne permet que de déterminer s'il existe une stratégie gagnante (ou en d'autres termes, un plan conduisant au but avec certitude). En pratique, on voudrait aussi déterminer un tel plan, ou du moins une approximation ou une partie de ce plan.

Il s'agit donc de résoudre non plus le problème de décision QBF mais le *problème de recherche* associé, que nous dénotons FQBF. Ce problème n'a, à notre connaissance, pas encore été étudié, sauf dans des cas particuliers très simples. Dans un premier temps, après avoir rappelé la définition de QBF, nous nous fixerons comme objectif de définir formellement ce qu'est une solution pour un problème FQBF. Dans un second temps, nous aborderons la résolution des problèmes FQBF et insisterons sur la question de la *taille de la représentation*

des solutions : en effet, celles-ci sont dans le pire cas exponentiellement grandes et il est indispensable de rechercher des façons les plus concises possibles de les représenter. Enfin, puisque la résolution d'un problème QBF est typiquement très difficile, nous nous attarderons sur la notion d'*approximation* de QBF, ce qui nous conduira sur deux pistes différentes. La première consiste à chercher des solutions partielles ; nous nous focaliserons en particulier sur les problèmes $\text{QBF}_{2,\forall}$ et $\text{QBF}_{3,\forall}$ à ce sujet. La seconde piste consiste à passer à un problème d'optimisation en généralisant le problème MAXSAT en un problème MAXQBF.

1.2 Notations et rappels de complexité

Soit PS un ensemble fini de variables propositionnelles et $PROP_{PS}$ le langage propositionnel construit sur l'ensemble des constantes booléennes $\{\top, \bot\}$ et PS de la façon usuelle.

Nous noterons $\vec{x}$ les instantiations des variables de $X \subseteq PS$ et 2^X l'ensemble de toutes les instantiations possibles de X. Ainsi, si $X = \{a, b, c\}$, $\vec{x} = \{a, \neg b, c\}$ est une instantiation complète de X. Soient X et Y deux sous-ensembles disjoints de $PROP_{PS}$, $(\vec{x}, \vec{y})$ est la concaténation de $\vec{x}$ et $\vec{y}$: dans cette instantiation, chaque variable de X (resp. Y) prend la valeur indiquée par $\vec{x}$ (resp. $\vec{y}$).

Lorsque $\Phi \in PROP_{PS}$ et $\vec{x} \in 2^X$, nous noterons $\Phi_{\vec{x}}$ la formule obtenue en conditionnant Φ par $\vec{x}$; cette formule, appelée conditionnement de Φ par $\vec{x}$ est obtenue en remplaçant dans Φ chaque occurrence de chaque variable x de X par $\top$ (resp. $\bot$) si $x \in \vec{x}$ (resp. $\neg x \in \vec{x}$). Pour simplifier certaines notations, lorsque $X = \{x\}$, nous noterons $\Phi_{x=1}$ (resp. $\Phi_{x=0}$) le conditionnement de Φ par $\vec{x} = \{x\}$ (resp. $\vec{x} = \{\neg x\}$).

Nous utiliserons également les notations suivantes :

Définition 1 (quantificateurs)
On note $Q = \{\exists, \forall\}$. Si $q \in Q$ alors le complément de q, noté $\bar{q}$, est simplement défini par $\bar{\exists} = \forall$ et $\bar{\forall} = \exists$. Enfin, si $q \in Q$ et k est un entier positif, on note :

- $last(k, q) = \begin{cases} q & \text{si } k \text{ est impair} \\ \bar{q} & \text{si } k \text{ est pair} \end{cases}$
- $|\exists(k, q)| = \begin{cases} \frac{k}{2} & \text{si } k \text{ est pair} \\ \frac{k+1}{2} & \text{si } k \text{ est impair et } q = \exists \\ \frac{k-1}{2} & \text{si } k \text{ est impair et } q = \forall \end{cases}$
- $|\forall(k, q)| = k - |\exists(k, q)|$.

$last(k, q)$ représente le quantificateur le plus imbriqué de la QBF considérée. $|\exists(k, q)|$ (resp. $|\forall(k, q)|$) représente le nombre de quantifications existentielles (resp. universelles) de cette QBF.

Dans cet article, nous ferons également référence à certaines classes de complexité de la hiérarchie polynomiale construite au dessus de NP et coNP (pour plus de détails, voir par exemple [18]). Nous supposons que le lecteur connaît les classes P, NP et coNP, ainsi que les problème de satisfaisabilité et de validité en logique propositionnelle (notés respectivement SAT et VAL).

Étant donné un problème A, nous dénotons par $\overline{\text{A}}$ le problème complémentaire de A, et si C est une classe de complexité, la classe de complexité coC est définie comme l'ensemble de tous les langages L dont le complémentaire $\bar{L}$ est dans C. Étant donné une classe de complexité C, la classe NP^{C} est l'ensemble de tous les langages reconnaissables en temps polynomial sur une machine de Turing non déterministe *munie d'un oracle* C, où un oracle C résout une instance quelconque d'un problème dans C en temps unité. Nous définissons maintenant les classes suivantes :

- $\Sigma_0^p = \Pi_0^p = \text{P}$;
- pour tout $k \geq 1$, $\Sigma_k^p = \text{NP}^{\Sigma_{k-1}^p}$, et $\Pi_k^p = \text{co}\Sigma_k^p$;
- $\text{PH} = \bigcup_{k \geq 0} \{\Sigma_k^p, \Pi_k^p\}$;
- PSPACE est l'ensemble de tous les langages reconnaissables par une machine de Turing déterministe utilisant un espace polynomial.

On peut noter les inclusions suivantes : pour tout k, $\Sigma_k^p \subseteq \Sigma_{k+1}^p$, $\Sigma_k^p \subseteq \Pi_{k+1}^p$, $\Pi_k^p \subseteq \Sigma_{k+1}^p$, $\Pi_k^p \subseteq \Pi_{k+1}^p$, et $\text{PH} \subseteq \text{PSPACE}$.

2 QBF : le problème de décision

2.1 Définitions

Définition 2 (formules booléennes quantifiées)
Soit k un entier positif et $q \in Q$ un quantificateur. Une formule booléenne quantifiée (QBF) est un $(k+3)$-uplet $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$ où $\{X_1, ..., X_k\}$ est une partition de l'ensemble des variables propositionnelles de Φ, une formule de $PROP_{PS}$.

Notons qu'il est en théorie inutile de spécifier k, puisqu'il est déterminé par le nombre d'éléments de P. Cependant, pour une meilleure lisibilité, on préfère le mentionner. Par un même souci de lisibilité, on notera souvent le problème $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$ de la manière suivante :

$\exists X_k \forall X_{k-1} \ldots \forall X_1 \Phi$ si $q = \exists$ et k est pair
$\exists X_k \forall X_{k-1} \ldots \exists X_1 \Phi$ si $q = \exists$ et k est impair
$\forall X_k \exists X_{k-1} \ldots \exists X_1 \Phi$ si $q = \forall$ et k est pair
$\forall X_k \exists X_{k-1} \ldots \forall X_1 \Phi$ si $q = \forall$ et k est impair

C'est la convention d'écriture que nous avons adoptée dans l'introduction de ce papier.

On appelle *formule booléenne quantifiée de rang k et d'initiateur q* (en abrégé $\text{QBF}_{k,q}$) une formule booléenne quantifiée du type $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$.

Il reste maintenant à définir les QBF positives, ou en d'autres termes les instances positives du problème de décision QBF.

Définition 3 (instances positives de QBF)
$P = \langle k, q, X_k, ..., X_1, \Phi \rangle$ *est une instance* positive *de* QBF *si et seulement si l'une de ces conditions est vraie :*

- $k = 0$ *et* $\Phi = \top$;
- $k \geq 1$ *et* $q = \exists$ *et il existe une X_k-instantiation* $\vec{x_k} \in 2^{X_k}$ *telle que* $\langle k - 1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}} \rangle$ *soit une instance positive de* $\text{QBF}_{k-1,\forall}$;

– $k \geq 1$ *et* $q = \forall$ *et pour toute* X_k*-instantiation* $\vec{x_k} \in 2^{X_k}$*, on a :* $\langle k-1, \exists, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}} \rangle$ *est une instance positive de* $\text{QBF}_{k-1,\exists}$.

Une définition équivalente est : $P = \langle k, q, X_k, ..., X_1, \Phi \rangle$ est une instance *positive* de QBF si et seulement si

$$q\vec{x_k} \in 2^{X_k} \bar{q}\vec{x_{k-1}} \in 2^{X_{k-1}} ... last(k,q)\vec{x_1}(\vec{x_1}, ..., \vec{x_k}) \models \Phi$$

Ainsi, par exemple, $\langle 3, \forall, X_3, X_2, X_1, \Phi \rangle$ est une instance *positive* de QBF si et seulement si pour tout $\vec{x_3} \in 2^{X_3}$ il existe $\vec{x_2} \in 2^{X_2}$ tel que pour tout $\vec{x_1} \in 2^{X_1}$ on ait $(\vec{x_1}, \vec{x_2}, \vec{x_3}) \models \Phi$.
Il est intéressant de s'attarder sur le cas $k = 1$: une conséquence de la définition précédente est que
– $P = \langle 1, \exists, X_1, \Phi \rangle$ est positive si et seulement si Φ est satisfaisable : $\text{QBF}_{1,\exists}$ coïncide avec le problème SAT.
– $P = \langle 1, \forall, X_1, \Phi \rangle$ est positive si et seulement si Φ est valide : $\text{QBF}_{1,\forall}$ coïncide avec le problème VAL.
Plus largement, $\text{QBF}_{k,\exists}$ (respectivement $\text{QBF}_{k,\forall}$) est le problème Σ_k^p-complet (respectivement Π_k^p-complet) canonique.

2.2 Résolution pratique de QBF

Il existe au moins deux approches utilisées actuellement pour résoudre des problèmes QBF : la première est fondée sur une transformation du problème original en une instance SAT, la seconde sur l'utilisation de prouveurs spécialisés.

Résoudre QBF avec SAT. Cette approche est utilisée actuellement avec succès dans le cadre de la vérification formelle (*model checking*) [3]. On dispose d'une formule propositionnelle quantifiée de la forme $qX_k\bar{q}X_{k-1} \ldots \exists X_1 \Phi$. L'idée est d'éliminer toutes les variables quantifiées appartenant à $X_k \ldots X_2$ afin de retrouver une formule de la forme $\exists X_1 \Phi'$, c'est-à-dire un problème SAT. On utilise pour cela les règles suivantes : une formule du type $\forall x \Phi(x)$ est remplacée par $\Phi_{x=1} \wedge \Phi_{x=0}$ et une formule du type $\exists x \Phi(x)$ par $\Phi_{x=1} \vee \Phi_{x=0}$. On dit que la variable x a été *dépliée*. Les quantificateurs sont traités de la gauche vers la droite. L'inconvénient de cette méthode est de construire une formule propositionnelle de taille exponentielle par rapport à celle de départ. Des techniques de substitution et de déplacement des quantificateurs peuvent être utilisées pour limiter le nombre de variables à déplier (voir par exemple [22]). La formule Φ' est ensuite transformée en CNF et peut être traitée par n'importe quel prouveur SAT.

Résoudre QBF comme SAT. Parce que SAT est un cas particulier de QBF, les méthodes de résolution dédiées à QBF sont souvent des généralisations des techniques employées avec succès pour résoudre SAT. Des algorithmes polynomiaux ont été présentés par [1] pour des formules 2-SAT quantifiées et par [13] pour des clauses de Horn. Plus récemment, des prouveurs QBF fondés sur la procédure de Davis et Putnam (DPLL) [8] sont apparus : le connecteur de branchement (un $\vee$ dans la procédure originale) devient dépendant du quantificateur de la variable de branchement ($\vee$ pour $\exists$, $\wedge$ pour $\forall$). *Evaluate* [5] se focalise sur la généralisation de la propagation des clauses unitaires et des littéraux purs. *Decide* [19, 20] incorpore des techniques de *lookahead* pour fixer la valeur de vérité des variables dans l'esprit de SATZ [16]. Les tout nouveaux *QuBE* [11] et *SemProp* [14] incorporent le retour arrière intelligent (*backjumping*) de RELSAT [2]. Ces algorithmes suivent donc les mêmes évolutions que DPLL pour SAT : on peut penser que la prochaine génération de prouveurs utilisera les techniques probabilistes (*randomization*) et d'apprentissage (*learning*) utilisées dans les derniers prouveurs SAT [17] (voir [14] pour plus de détails sur les différentes formes d'apprentissage pour QBF).

SAT versus QBF. Pour l'instant, les techniques basées sur une transformation du problème original en problème SAT donnent de meilleurs résultats que les prouveurs spécialisés [20]. On peut avancer plusieurs raisons pour expliquer cela :
– l'intérêt porté au problème SAT durant les dix dernières années a contribué au développement de prouveurs très efficaces (voir par exemple le tout nouveau *Chaff* [17]) ;
– les heuristiques utilisées pour SAT ne peuvent pas être utilisées directement pour QBF car le choix des variables de branchement est restreint par leur quantificateur ;
– la transformation en SAT est surtout appliquée avec succès dans le cadre d'instances QBF *structurées*, (vérification formelle, planification conditionnelle) car il est possible d'utiliser des connaissances du domaine pour simplifier ces instances.

3 FQBF : le problème de recherche

On sait que, lorsqu'un problème donné se réduit à un problème SAT, résoudre seulement le problème de décision SAT ne permet en général que de déterminer si le problème de départ admet une solution ; si l'on veut exhiber une *solution* de ce dernier, il faudra résoudre le problème de recherche[1] FSAT correspondant, c'est-à-dire trouver un modèle (s'il en existe un) de la formule propositionnelle correspondante. De la même façon, lorsqu'un problème donné s'exprime comme un problème QBF, la résolution de ce problème (de décision) QBF ne fait que déterminer si le problème a ou non une solution. Si l'on veut exhiber une solution du problème il faudra *résoudre le problème de recherche associé à* QBF. Dans le cadre de SAT, on

[1] en anglais, *function problem*.

ne fait pas vraiment *en pratique* la différence entre le problème de décision et le problème de recherche (les meilleurs prouveurs SAT actuels cherchent un modèle ou un impliquant de la formule). Il n'en est pas de même pour QBF.

À notre connaissance, les problèmes de recherche associés à QBF n'ont jamais été étudiés, sauf bien sûr pour $(k,q) = (1,\exists)$. La première chose à faire est d'identifier les objets jouant le même rôle que celui que les interprétations propositionnelles jouent pour SAT. Ces objets sont des *politiques*. Intuitivement, une politique est une application qui associe à chaque affectation d'un groupe de variables universellement quantifiées une affectation du groupe de variables existentiellement quantifiées qui le suit.

Définition 4 (politiques totales)
L'ensemble $TP(k,q,X_k,...,X_1)$ *est défini récursivement comme suit :*
- $TP(0,q) = \{\lambda\}$;
- $TP(k,\exists,X_k,...,X_1) =$
  $\{\vec{x_k}\ ;\pi_{k-1} \mid \pi_{k-1} \in TP(k-1,\forall,X_{k-1},...,X_1)\}$;
- $TP(k,\forall,X_k,...,X_1)$
  $= [2^{X_k} \to TP(k-1,\exists,X_{k-1},...,X_1)\}]$[2].

La politique λ désigne la *politique vide*. L'opérateur ";" représente la composition en séquence de politiques. Par abus de notation, on omet la plupart du temps λ dans les politiques composées : ainsi, $\pi;\lambda$ est abrégée en π.

Par exemple, voici une politique de $TP(3,\exists,\{e,f\},\{a,b\},\{c,d\})$:
$\pi = (e,\neg f);\pi'$ où
$\pi'(\neg a,\neg b)) = (c,d)$;
$\pi'(\neg a,b)) = (c,d)$;
$\pi'(a,\neg b)) = (\neg c,d)$;
$\pi'(a,b)) = (\neg c,d)$.

Intuitivement, exécuter π consiste à d'abord instancier e à vrai et f à faux, puis à observer les valeurs que prennent a et b et d'agir en conséquence, c'est-à-dire d'instancier c et d à vrai si a est faux, ou c à faux et d à vrai sinon.

On vérifie que :
- une politique de $TP(1,\exists,X_1)$ a la forme $(\vec{x_1};\lambda)$, c'est-à-dire $\vec{x_1}$;
- $TP(1,\forall,X_1)$ est réduit à une politique unique : la fonction constante qui associe λ à toute instanciation de X_1. On note cette politique λ_{X_1} ;
- une politique de $TP(2,\exists,X_2,X_1)$ a la forme $(\vec{x_2}\ ;\lambda_{X_1})$;
- une politique de $TP(2,\forall,X_2,X_1)$ est une fonction de 2^{X_2} dans 2^{X_1}.

Une politique π de $TP(k,q,X_k,...,X_1)$ peut être vue comme un arbre dont la profondeur est égale à $|\forall(k,q)|$, dont les feuilles sont étiquetées par des affectations de X_1 si $last(k,q) = \exists$ et par λ si $last(k,q) = \forall$ et dont les nœuds intermédiaires sont étiquetés par des affectations de groupes de variables existentiellement quantifiées et dont les branches sont étiquetées par des affectations de groupes de variables universellement quantifiées.

[2] c'est-à-dire l'ensemble des fonctions totales de 2^{X_k} dans $TP(k-1,\exists,X_{k-1},...,X_1)$.

Définition 5 (satisfaction par une politique)
On définit récursivement la satisfaction d'une instance $P = \langle k,q,X_k,...,X_1,\Phi\rangle$ *de* $\text{QBF}_{k,q}$ *par une politique* π *de* $TP(k,q,X_k,...,X_1)$ *comme suit :* π *satisfait* P *(noté* $\pi \models P$*) si et seulement si l'une de ces conditions est vérifiée :*
- $k = 0$ *et* $\pi = \lambda$ *et* $\Phi = \top$;
- $k \geq 1$ *et* $q = \exists$ *et* $\pi = (\vec{x_k};\pi')$ *avec* $\pi' \models \langle k-1,\forall,X_{k-1},...,X_1,\Phi_{\vec{x_k}}\rangle$;
- $k \geq 1$ *et* $q = \forall$ *et pour tout* $\vec{x_k} \in 2^{X_k}$ *on a* $\pi(\vec{x_k}) \models \langle k-1,\exists,X_{k-1},...,X_1,\Phi_{\vec{x_k}}\rangle$.

Donnons quelques exemples.

Exemple 1 $\langle 1,\exists,\{a,b\},a\to b\rangle$ *est satisfait par* $\pi = (a,b)$.

Exemple 2 *Il n'existe pas de politique satisfaisant* $\langle 1,\forall,\{a,b\},a\to b\rangle$.

Exemple 3 $\langle 2,\exists,\{a,b\},\{c\},(a\vee b)\wedge(a\to c)\wedge(b\vee c)\rangle$ *est satisfait par* $\pi = (\neg a,b)$.

Exemple 4 *Il n'existe pas de politique satisfaisant* $\langle 2,\forall,\{a,b\},\{c\},(a\vee b)\wedge(a\to c)\wedge(b\vee c)\rangle$ *ni de politique satisfaisant* $\langle 2,\forall,\{a,b\},\{c\},(a\vee c)\wedge(a\to\neg c)\rangle$.

Exemple 5 $\langle 2,\forall,\{a,b\},\{c,d\},(a\to(c\vee d))\wedge(b\leftrightarrow\neg c)\rangle$ *est satisfait par* π *:* $\pi((a,b)) = \pi((\neg a,b)) = (\neg c,d)$ *;* $\pi((a,\neg b)) = \pi((\neg a,\neg b)) = (c,d)$. *On notera une telle politique ainsi :*

$$\pi = \begin{bmatrix} (a,b) & \mapsto (\neg c,d) \\ (\neg a,b) & \mapsto (\neg c,d) \\ (a,\neg b) & \mapsto (c,d) \\ (\neg a,\neg b) & \mapsto (c,d) \end{bmatrix} \text{ ou encore :}$$

$$\pi = \begin{bmatrix} (a,b),(\neg a,b) & \mapsto (\neg c,d) \\ (a,\neg b),(\neg a,\neg b) & \mapsto (c,d) \end{bmatrix}.$$

Exemple 6 $\langle 3,\exists,\{a\},\{b\},\{c,d\},(a\to(c\wedge d))\wedge(b\leftrightarrow\neg c)\rangle$ *est satisfait par* $\pi = \neg a;\begin{bmatrix} (b), & \mapsto (\neg c,d) \\ (\neg b) & \mapsto (c,d) \end{bmatrix}$.

On vérifie que pour $(k,q) = (1,\exists)$, $\pi = \vec{x_1}$ satisfait $P = \langle 1,\exists,X_1,\Phi\rangle$ si et seulement si $\vec{x_1} \models \Phi$ et que pour $(k,q) = (1,\forall)$, $\pi = \lambda_{X_1}$ satisfait $P = \langle 1,\forall,X_1,\Phi\rangle$ si et seulement si Φ est valide. Plus généralement, on a le résultat suivant, qui nous montre que les politiques satisfaisant P constituent bien le résultat désiré du problème de recherche associé à P. On les appelle des *politiques solutions* pour P.

Proposition 1 $P = \langle k,q,X_k,...,X_1,\Phi\rangle$ *est une instance positive de* $\text{QBF}_{k,q}$ *si et seulement si il existe une politique* $\pi \in TP(k,q,X_k,...,X_1)$ *telle que* $\pi \models P$.

La preuve (par récurrence sur k) ne pose aucune difficulté.
Cette proposition nous permet de définir formellement le problème de recherche $\text{FQBF}_{k,q}$ comme suit :

Définition 6 (FQBF : problème de recherche) *Soit $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ une QBF. Résoudre le problème de recherche associé à P consiste à exhiber une politique π telle que $\pi \models P$, s'il en existe une. On appelle* FQBF *(respectivement $\text{FQBF}_{k,q}$) le problème de recherche associé à* QBF *(respectivement $\text{QBF}_{k,q}$).*

Il est utile de donner ici une interprétation intuitive de FQBF dans le cas où $k = 2$. On a vu en introduction qu'au second niveau, les problèmes QBF peuvent représenter des problème de décision non séquentiel sous incertitude :
- sous absence totale d'observabilité, il faut essayer de trouver une décision qui satisfasse les buts de Φ quelles que soient les valeurs des variables d'état, c'est-à-dire trouver une solution à la formule $\exists X \forall Y \Phi$ - en d'autres termes résoudre le problème $\text{FQBF}_{2,\exists}$ associé à $\langle 2, \exists, X, Y, \Phi\rangle$.
- sous observabilité totale, il faut préparer, pour chaque combinaison possible des variables d'état, une décision qui satisfasse les buts de Φ, c'est-à-dire trouver une solution à la formule $\forall Y \exists X \Phi$ - en d'autres termes résoudre le problème $\text{FQBF}_{2,\forall}$ associé à $\langle 2, \forall, Y, X, \Phi\rangle$.

4 Politiques partielles

4.1 Motivations et définitions

En pratique, la définition d'une politique solution est trop exigeante. En effet, considérons $P = \langle 2, \forall, \{a, b\}, \{c\}, (a \to c) \wedge (b \to \neg c)\rangle$. P n'a pas de politique solution parce que l'affectation (a, b) rend Φ insatisfaisable : ainsi, si la nature joue le coup (a, b), l'agent ne pourra plus rien faire qui puisse le conduire à satisfaire Φ. En revanche, si la nature joue un autre coup que (a, b) alors l'agent *peut* faire quelque chose de satisfaisant, à savoir, $(a, \neg b) \mapsto c$, $(\neg a, b) \mapsto \neg c$, $(\neg a, \neg b) \mapsto c$ (ou $\neg c$). Ces politiques ne sont pas définies pour toutes les affectations possibles des groupes de variables universellement quantifiées, d'où leur nom de *politiques partielles*, dont nous donnons maintenant une définition formelle :

Définition 7 (politique partielle) *L'ensemble $PP(k, q, X_k, ..., X_1)$ des politiques partielles pour la QBF $P = \langle k, q, X_k, \ldots, X_1\rangle$ est défini récursivement comme suit :*
- $TP(1, \exists, X_1) = 2^{X_1} \cup \{\otimes\}$;
- $TP(1, \forall, X_1) = (2^{X_1} \to \{\lambda, \otimes\})$;
- $TP(k, \exists, X_k, ..., X_1) =$
  $\{\vec{x_k}; \pi_{k-1} \,|\, \pi_{k-1} \in PP(k-1, \forall, X_{k-1}, .., X_1) \cup \{\otimes\}$;
- $TP(k, \forall, X_k, ..., X_1)$
  $= (2^{X_k} \to PP(k-1, \exists, X_{k-1}, ..., X_1))$.

$\otimes$ représente l'échec (i.e. il n'est pas possible de trouver une politique solution pour la QBF considérée).

Définition 8 (politique partielle saine) *Une politique partielle $\pi \in PP(k, q, X_k, ..., X_1)$ est* saine *pour $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ si et seulement si l'une des conditions suivantes est satisfaite :*

1. *$q = \exists$ et $\pi = \otimes$;*
2. *$(k, q) = (1, \exists)$, $\pi = \vec{x_1}$ et $\vec{x_1} \models \Phi$;*
3. *$(k, q) = (1, \forall)$ et pour tout $\vec{x_1} \in 2^{X_1}$ on a soit $\pi(\vec{x_1}) = \otimes$, soit $(\pi(\vec{x_1}) = \lambda$ et $\vec{x_1} \models \Phi)$;*
4. *$k > 1$, $q = \exists$, $\pi = \vec{x_k}; \pi_{k-1}$ et π_{k-1} est saine pour $\langle k-1, \forall, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$;*
5. *$k > 1$, $q = \forall$, et pour tout $\vec{x_k} \in 2^{X_k}$, $\pi(\vec{x_k})$ est saine pour $\langle k-1, \exists, X_{k-1}, ..., X_1, \Phi_{\vec{x_k}}\rangle$;*

Alors qu'il existe des instances de QBF qui n'ont pas de politique solution, il est clair que toutes les instances de QBF ont une politique partielle saine. Lorsqu'un nœud de l'arbre est associé à $\otimes$, on renonce à choisir une affectation ("échec") pour ce nœud et tous les suivants. Il s'agit bien évidemment d'éviter autant que faire se peut les nœuds échec, c'est-à-dire n'associer "échec" à un nœud que lorsqu'on ne peut pas faire autrement :

Définition 9 (politiques saines maximales) *Soient π et π' deux politiques partielles de $PP(q, k, X_k, ..., X_1)$. On dit que π est au moins aussi couvrante que π', noté $\pi \sqsupseteq \pi'$, si et seulement si l'une des conditions suivantes est vérifiée :*
- *$q = \exists$ et $\pi' = \otimes$;*
- *$q = \exists$, $\pi = [\vec{x_k}; \pi_{k-1}]$, $\pi' = [\vec{x'_k}; \pi'_{k-1}]$, et $\pi_{k-1} \sqsupseteq \pi'_{k-1}$;*
- *$q = \forall$ et pour tout $\vec{x_k} \in 2^{X_k}$, on a $\pi(\vec{x_k}) \supseteq \pi'(\vec{x_k})$.*

π est un préordre partiel ; on note $\pi \gg \pi'$ pour $\pi \sqsupseteq \pi'$ et non $(\pi' \sqsupseteq \pi)$.
π est une politique partielle saine maximale *pour une instance P de* QBF *si et seulement si π est saine pour P et il n'existe pas de politique π' saine pour P telle que $\pi' \gg \pi$ et $\pi' \models P$.*

Définition 10 (SQBF : 2^e problème de recherche) *Soit $P = \langle k, q, X_k, ..., X_1, \Phi\rangle$ une QBF. Résoudre le second problème de recherche associé à P consiste à exhiber une politique π saine maximale pour P. On appelle* SFQBF *(respectivement $\text{SFQBF}_{k,q}$) le second problème de recherche associé à* QBF *(respectivement $\text{QBF}_{k,q}$).*

4.2 Résolution de $\text{SFQBF}_{3,\forall}$

Nous nous intéressons ici à la résolution pratique de $\text{SFQBF}_{3,\forall}$. Il convient d'abord de justifier ce choix de $(k, q) = (3, \exists)$ comme sujet d'étude. Premièrement, il est important, avant de se lancer dans la résolution de problèmes $\text{SFQBF}_{k,q}$ trop complexes, d'essayer de résoudre les problèmes des premiers rangs (ils sont déjà

bien assez complexes). Le cas $k = 1$ a été énormément étudié ; $\text{SFQBF}_{2,\exists}$ n'est pas nouveau non plus, puisqu'il se ramène à un problème d'abduction : il suffit en effet de trouver *une* affectation $\vec{x_1}$ telle que $\Phi_{\vec{x_1}}$ soit valide ; ce problème est connu et ne pose aucune difficulté de taille. Les choses se compliquent avec $\text{SFQBF}_{2,\forall}$ et nous aurions pu nous fixer ce problème comme cas d'étude ; cependant, du point de vue de la taille des politiques, qui est le point qui nous intéresse le plus ici, $\text{SFQBF}_{3,\forall}$ n'est pas bien plus complexe que $\text{SFQBF}_{2,\forall}$, d'où notre choix (notons que la résolution de $\text{SFQBF}_{3,\forall}$ nous permet *a fortiori* de résoudre $\text{SFQBF}_{2,\forall}$ qui en est un cas particulier).
Nous exposons d'abord les limitations inhérentes à la résolution de ce problème avant de présenter une méthode pratique de résolution fondée sur l'idée de compilation.

Position du problème. Le problème $\text{SFQBF}_{3,\forall}$ peut être défini plus simplement ainsi :
- Entrée : une formule $\forall X \exists Y \forall Z \Phi$ de $\text{QBF}_{3,\forall}$;
- Résultat : une fonction totale π de 2^X dans $2^Y \cup \{\otimes\}$ qui associe à tout $\vec{x}$ de 2^X un $\vec{y}$ de 2^Y tel que $(\vec{x}, \vec{y}) \models \forall Z \Phi$ s'il en existe un et $\otimes$ sinon.

Lorsqu'une $\text{QBF}_{3,\forall}$ est utilisée pour modéliser un problème de prise de décision dans l'incertain, $\vec{x}$ représente une observation et $\vec{y} = \pi(x)$ est une décision qui couvre $\vec{x}$, c'est-à-dire si la décision $\vec{y}$ est prise lorsque $\vec{x}$ est observée alors Φ est satisfaite.
La résolution de $\text{SFQBF}_{3,\forall}$ est nécessairement au moins aussi difficile que la résolution de $\text{FQBF}_{3,\forall}$ puisque si une politique solution existe, toute politique saine maximale est forcément une politique solution.
Maintenant, toutes les politiques saines maximales sont-elles aussi intéressantes les unes que les autres ? Clairement, la réponse est non. En effet, un critère important dans le choix d'une politique saine maximale π concerne les aspects calculatoires inhérents à sa représentation et à son utilisation (ressources de calcul consommées, temps et espace). π en tant que fonction doit non seulement être calculable (sinon on ne voit pas vraiment ce que l'on pourrait en faire) mais l'être en temps polynomial en la taille de sa représentation.

Définition 11 (politiques saines efficaces)
Une politique π saine pour $P = \langle 3, \forall, X, Y, Z, \Phi \rangle$ est efficace *si et seulement si π est calculable efficacement en tant que fonction, i.e., il existe un algorithme en temps polynomial en la taille de l'entrée qui, pour toute entrée $\vec{x} \in 2^X$, retourne $\pi(\vec{x})$.*

En outre, la taille de la représentation de la politique doit être la plus concise possible (de façon à ce que l'espace mémoire consommé soit le plus petit possible) et idéalement polynomiale en la taille de l'entrée.
Malheureusement, il n'existe pas de politique saine maximale ayant une représentation de taille polynomiale qui soit *explicite*, c'est-à-dire un ensemble de couples $(\vec{x}, \pi(\vec{x}))$ tel que $(\vec{x}, \pi(\vec{x})) \models \Phi$ pour tout $\vec{x} \in 2^X$.

Proposition 2 *[10] Il existe des formules positives $P = \langle 2, \forall, X, Y, \Phi \rangle$ de $\text{QBF}_{2,\forall}$ dont toute politique solution est injective.*

Preuve : il suffit de considérer la formule $\forall \{x_1, \ldots, x_n\} \exists \{y_1, \ldots, y_n\} \bigwedge_{i=1}^{n} (x_i \leftrightarrow y_i)$.
Pour autant, on peut facilement construire une politique efficace π pour cette formule : il suffit de donner à chaque y_i la valeur de vérité prise par x_i dans $\vec{x}$.
On peut alors se tourner du côté des représentations implicites (i.e., π est une fonction calculée et pas une fonction tabulée (observation, décision)) mais là encore, il est peu vraisemblable qu'une représentation de taille polynomiale d'une politique saine maximale efficace puisse exister en toute généralité, même dans le cas plus simple $\text{QBF}_{2,\forall}$.

Proposition 3 *Si à toute formule $P = (2, \forall, X, Y, \Phi)$ de $\text{QBF}_{2,\forall}$, on savait associer une structure de données de taille polynomiale en $|P|$ représentant une politique efficace π qui soit saine et maximale pour P, alors la hiérarchie polynomiale s'effondrerait au deuxième niveau.*

Preuve : si cela était possible, on saurait en particulier associer à toute formule CNF Σ telle que $P = Var(\Sigma) = \{p_1, \ldots, p_n\}$ la formule suivante de $\text{QBF}_{2,\forall}$:

$$\forall L \exists P (\Sigma \wedge \bigwedge_{i=1}^{n} (\neg l_i^+ \vee p_i) \wedge (\neg l_i^- \vee \neg p_i))$$

avec $L = \bigcup_{i=1}^{n} \{l_i^+, l_i^-\} \subseteq PS \setminus P$.
Toute politique efficace π pour cette formule permet de rendre traitable l'interrogation clausale à partir de Σ. En effet, pour tout clause γ construite sur les variables de P, on a $\Sigma \models \gamma$ ssi $\Sigma \wedge \neg\gamma$ est contradictoire. Chaque terme non contradictoire $\neg\gamma$ possible correspond à un vecteur $\vec{l}$ de 2^L tel que $\forall p_i \in P$ si $p_i \in \neg\gamma$ alors $l_i^+ \in \vec{l}$ et $\neg l_i^- \in \vec{l}$, si $\neg p_i \in \neg\gamma$ alors $\neg l_i^+ \in \vec{l}$ et $l_i^- \in \vec{l}$ et si $p_i \notin \neg\gamma$ et $\neg p_i \notin \neg\gamma$ alors $\neg l_i^+ \in \vec{l}$ et $\neg l_i^- \in \vec{l}$. Par conséquent, pour toute clause non valide γ sur P, on a $\Sigma \models \gamma$ si et seulement si $\Sigma \wedge \bigwedge_{i=1}^{n} (\neg l_i^+ \vee p_i) \wedge (\neg l_i^- \vee \neg p_i) \wedge \vec{l}$ est contradictoire si et seulement si $\pi(\vec{l}) = \otimes$. Si π est efficace alors le test $\pi(\vec{l}) \stackrel{?}{=} \otimes$ peut être réalisé en temps polynomial, donc la réponse au test $\Sigma \stackrel{?}{\models} \gamma$ peut également être obtenue en temps polynomial. Or, l'existence d'une forme compilée de taille polynomiale permettant d'assurer une interrogation clausale traitable pour toute formule propositionnelle Σ entraînerait $\mathsf{NP} \subseteq \mathsf{P/poly}$, ce qui est considéré comme peu vraisemblable car cela provoquerait l'effondrement de la hiérarchie polynomiale au second niveau [12].
On se limite donc en pratique à rechercher des politiques efficaces dont la représentation est la plus

concise possible, sans garantie de polynomialité dans tous les cas.

Génération et représentation compacte de politiques. Comment résoudre une instance $\forall X \exists Y \forall Z \Phi$ de $\text{SFQBF}_{3,\forall}$?
Une méthode simple pour ce faire consiste à remarquer tout d'abord que, si l'on considère la formule $\Psi = \forall Z \Phi$, il suffit de résoudre $\forall X \exists Y \Psi$, qui est une instance de $\text{SFQBF}_{2,\forall}$. Le principe de cette méthode est de *compiler* $\forall Z \Phi$ sous forme d'une "formule" logiquement équivalente *comp* qui permet, pour n'importe quel $\vec{x}$, de :

1. Conditionner *comp* par $\vec{x}$ en temps polynomial pour produire une nouvelle forme compilée $comp_{\vec{x}}$;
2. Extraire en temps polynomial un modèle de $comp_{\vec{x}}$ sur Y (ou $\otimes$ s'il n'en existe pas).

Le point important ici est que toute politique représentée par un tel *comp* est efficace. En effet, étant donnée une instantiation $\vec{x}$, le modèle retourné par l'algorithme en temps polynomial (dont l'existence est postulée ci-dessus) est (s'il existe) une instantiation $\vec{y}$ telle que $\vec{y} \models comp_{\vec{x}}$. Or on peut montrer que $\vec{y} \models comp_{\vec{x}}$ est satisfaisable si et seulement si $(\vec{x}, \vec{y}) \models \forall Z \Phi$: on a $\vec{x} \models \exists Y \forall Z \Phi$ si et seulement si $\vec{x} \models \exists Y comp$ si et seulement si $(\exists Y comp)_{\vec{x}}$ est valide (une propriété centrale du conditionnement est en effet qu'une instantiation $\vec{x}$ est un impliquant d'une formule Ψ si et seulement si le conditionnement $\Psi_{\vec{x}}$ est valide). Or, $(\exists Y comp)_{\vec{x}}$ est valide si et seulement si $\exists Y (comp_{\vec{x}})$ est valide (puisque $X \cap Y = \emptyset$). Enfin, remarquons que $\exists Y (comp_{\vec{x}})$ est valide si et seulement si $comp_{\vec{x}}$ est satisfaisable.
Quelles sont les classes "cibles" candidates à considérer pour *comp* ? Toute classe acceptable doit permettre de supprimer les quantifications universelles, conditionner (comme une loi interne) et rechercher un modèle en temps polynomial. Parmi celles-ci figurent toutes les classes polynomiales CNF du problème SAT qui sont stables par simplification (i.e., dans lesquelles supprimer des littéraux dans les clauses ou des clauses conduit toujours à une formule de la classe). En effet, la suppression des quantifications universelles $\forall z$ (avec $z \in Z$) peut s'effectuer en temps linéaire à partir d'une formule CNF ne contenant pas de clauses valides (ceci est le dual d'une propriété bien connue concernant la suppression de quantifications existentielles à partir de formules DNF) : il suffit de supprimer chaque littéral z ou $\neg z$ des clauses dans laquelle il apparait. Si la classe est stable par simplification, la formule résultante est encore dans la classe. Maintenant, toute formule stable par simplification l'est aussi par conditionnement (il suffit de se référer aux définitions pour s'en persuader). Enfin, pour toute formule appartenant à une classe polynomiale pour SAT stable par conditionnement, il est possible d'exhiber un modèle de la formule en temps polynomial (et même d'énumérer tous ses modèles en temps polynomial en la taille de la formule et en leur nombre [7]).

Un exemple d'une telle classe de formules est celle des formules CNF Horn renommables [15], c'est-à-dire il existe une substitution de certaines variables de la formule CNF par leur négation qui rend celle-ci Horn (chaque clause comporte au plus un littéral positif). Notons qu'il ne s'agit pas d'une classe complète pour la logique propositionelle : certains Φ ne sont logiquement équivalents à aucune formule CNF Horn renommable. De nombreuses autres classes polynomiales pour SAT sont complètes et stables par conditionnement et conviennent donc comme classes cibles pour *comp* : les "formules" DNF, $OBDD$ [4], et plus généralement $DNNF$ [6], etc. D'un point de vue pratique, ce qui est important est la disponibilité d'algorithmes (en espace exponentiel dans le pire des cas) permettant de construire une formule équivalente à $\forall Z \Phi$ appartenant à l'une de ces classes. C'est le cas par exemple pour la classe des "formules" $OBDD$.

Exemple 7 *Considérons la formule* $\langle 3, \forall, \{a,b\}, \{c,d\}, \{e,f\}, (a \wedge e \rightarrow (c \vee d \vee f)) \wedge (b \rightarrow \neg c) \wedge (\neg b \rightarrow c) \rangle$. *Cette formule étant une CNF (à une réecriture de* $\rightarrow$ *près), on obtient* $\forall \{e,f\} (a \wedge e \rightarrow (c \vee d \vee f)) \wedge (b \rightarrow \neg c) \wedge (\neg b \rightarrow c)$ *par simple suppression des littéraux portant sur* e *ou sur* f. *Ce qui donne la formule* $\Psi = (a \rightarrow (c \vee d)) \wedge (b \rightarrow \neg c) \wedge (\neg b \rightarrow c)$. *Le problème à résoudre est maintenant* $\langle 2, \forall, \{a,b\}, \{c,d\}, \Psi \rangle$. *La figure suivante décrit deux BDDs ordonnés et réduits (ROBDDs), associés à l'ordre sur les variables* $a < b < c < d$. *Celui de gauche représente la compilation sous forme de ROBDD de la formule* Ψ ; *celui de droite correspond au conditionnement* $\Psi_{\vec{x}}$ *de* Ψ *par* $\vec{x} = \{a,b\}$. *On peut facilement vérifier que le seul modèle de* $\Psi_{\vec{x}}$ *sur* $\{c,d\}$ *est* $\{\neg c, d\}$ *qui couvre bien l'observation* $\vec{x}$.

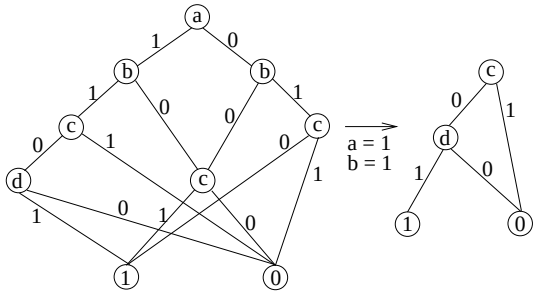

FIG. 1 – OBDDs de l'exemple 7

5 MAXQBF : optimisation

5.1 Motivations

Lorsqu'une formule QBF est une instance négative de QBF, se focaliser sur la recherche d'une politique partielle saine n'est pas toujours satisfaisant. Par exemple, lorsqu'une instance du problème $\text{QBF}_{3,\forall}$ est prise en compte, la recherche d'une politique partielle conduit à ne proposer aucune décision ($\otimes$) pour les observations problématiques (c'est-à-dire c'est à dire pour les cas où l'on ne peut pas couvrir toutes les configurations des variables d'état non observables). Pourtant, il est évident que, dans une telle situation, mieux vaut une bonne décision, c'est à dire une décision qui satisfait au mieux Φ, que pas de décision du tout. Il s'agit donc ici d'*optimiser* la satisfaction de Φ. Plusieurs approches sont possibles pour ce faire ; nous les esquissons dans la suite, en centrant notre propos sur $\text{QBF}_{2,\exists}$[3].

5.2 $\text{MAXQBF}_{2,\exists}$

Pour une formule $\exists X \forall Y \Phi$ de $\text{QBF}_{2,\exists}$, une politique $\vec{x}; \lambda_X$ est saine si et seulement si $\vec{x} \models \forall Y \Phi$. Dans un contexte d'optimisation, l'idée est d'associer à toute politique une utilité globale qui dépend de la manière dont $\vec{x}$ satisfait $\forall Y \Phi$, plus précisement de la manière dont $(\vec{x}; \vec{y})$ satisfait Φ pour chaque $\vec{y}$.
On définit l'utilité d'une instantiation $(\vec{x}; \vec{y})$ vis-à-vis d'une formule Φ sur $X \cup Y$ comme une fonction u_Φ de $2^{X \cup Y}$ dans $[0, 1]$ telle que si $(\vec{x}; \vec{y}) \models \Phi$ alors $u_\Phi(\vec{x}; \vec{y}) = 1$. 1 et 0 représentent respectivement les utilités maximale et minimale[4]. Par exemple, si Φ est une conjonction de n formules (sous-buts) $\varphi_1, \ldots, \varphi_n$ et qu'il peut y avoir compensation entre les satisfactions et les violations des sous-buts, le score d'une instantiation est le nombre de sous-buts de Φ satisfaits par cette instanciation :

$$u_\Phi(\vec{x}; \vec{y}) = Card(\{\varphi_i \in \Phi \mid (\vec{x}; \vec{y}) \models \varphi_i\}).$$

Ce critère est facilement généralisable de manière à tenir compte de buts d'importances inégales.
L'utilité d'une politique $(\vec{x}; \lambda_X)$ peut donc être définie comme une fonction croissante des $u_\Phi(\vec{x}; \vec{y})$, maximale si $\vec{x} \models \Phi$. Dans le cadre d'une interprétation du problème en termes de décision sous incertitude, la théorie de la décision permet de proposer plusieurs critères (cette liste n'étant pas exhaustive) :
- $U_\Phi(\vec{x}; \lambda_X) = \sum_{\vec{y} \in 2^Y} u_\Phi(\vec{x}; \vec{y})$: ceci correspond à l'hypothèse que tous les mondes $\vec{y}$ sont d'égale probabilité[5].
- $U_\Phi(\vec{x}; \lambda_X) = \min_{\vec{y} \in 2^Y} u_\Phi(\vec{x}; \vec{y})$[6].
- $U_\Phi(\vec{x}; \lambda_X) = Card(\{\vec{y} \mid \forall \vec{y}' \in 2^Y, U_\Phi(\vec{x}; \vec{y}) \geq U_\Phi(\vec{x}; \vec{y}')\})$.

Supposons qu'un choix ait été fait pour la définition de U_Φ. Par analogie avec le problème MAXSAT, où il faut trouver une instantiation qui maximise le nombre de formules satisfaites, on définit $\text{MAXQBF}_{2,\exists}$ comme le problème d'optimisation défini par :
- Entrée : une formule $P = \langle 2, \exists, X, Y, \Phi \rangle$ de $QBF_{2,\exists}$ où Φ est une conjonction de sous-formules,
- Résultat : un $\vec{x} \in 2^X$ maximisant $U_\Phi(\vec{x}; \lambda_X)$.

6 Conclusion

Dans cet article, nous nous sommes attachés au problème de recherche associé à QBF et à une approximation de celui-ci (second problème de recherche). Nous avons défini les notions de politique saine, politique solution, politique saine maximale, politique efficace. Nous avons montré que même dans des cas restreints il n'existe vraisemblablement pas de représentation de taille polynomiale pour les politiques saines maximales et efficaces. Pour pallier ce problème, nous avons proposé une approche de type compilation qui, dans des cas restreints, exploite les classes cibles pour la compilation de l'inférence clausale, bien connues dans la littérature. Enfin, nous avons esquissé une autre famille d'approximation de FQBF.

Les perspectives sont nombreuses. En plus du développement plus poussé des algorithmes de résolution (en particulier pour les problèmes MAXQBF), nous avons l'intention d'*expérimenter* nos méthodes sur des instances générées aléatoirement. Ceci pose plusieurs problèmes, en particulier celui d'établir les paramètres à étudier. En effet, lorsqu'il s'agit d'étudier le problème de décision QBF, on se focalise sur la variation de la proportion d'instances positives, ainsi que sur le temps moyen passé à déterminer si l'instance est positive ou non ; pour ce qui est des problèmes de recherche, les paramètres précédents ne sont plus pertinents : il faudra plutôt étudier la variation de la *taille* moyenne d'une politique et le *taux de couverture* moyen d'une politique (c'est-à-dire la proportion de branches qui ne conduisent pas à un échec). Enfin, nous envisageons à plus long terme des applications à la planification dans l'incertain.

Remerciements

Sylvie Coste-Marquis, Daniel Le Berre et Pierre Marquis remercient l'IUT de Lens, la Région Nord/Pas-de-Calais et les Communautés Européennes pour leur support.

[3] La définition de $\text{MAXQBF}_{k,q}$ pour k et q arbitraires peut se faire ensuite par induction de manière similaire à $\text{FQBF}_{k,q}$ mais nous l'omettons par manque de place. Par ailleurs, $\text{MAXQBF}_{2,\exists}$ est un problème original et déjà suffisamment complexe.

[4] Ce choix n'entraine pas de perte de généralité. On pourrait prendre n'importe quel autre intervalle numérique borné.

[5] Si ce n'était pas le cas, il suffirait d'utiliser la formule plus générale de l'utilité espérée : $U_\Phi(\vec{x}; \lambda_X) = \sum_{\vec{y} \in 2^Y} u_\Phi(\vec{x}; \vec{y}).pr(\vec{y})$.

[6] Là encore, si la vraisemblance des mondes peut être graduée par une mesure, des approches plus nuancées, qui s'appuyent sur la théorie de l'utilité qualitative, peuvent être suivies.

Références

[1] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8 :121–123, 1979.

[2] Roberto J. Jr. Bayardo and Robert C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of AAAI'97*, pages 203–208, 1997.

[3] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99), number 1579 in LNCS*, 1999.

[4] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 8 :677–692, C-35.

[5] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, pages 262–267, Madison (Wisconsin - USA), 1998.

[6] A. Darwiche. Compiling devices into decomposable negation normal form. In *Proceedings of IJCAI99*, pages 284–289.

[7] A. Darwiche and P. Marquis. A perspective on knowledge compilation. In *Proceedings of IJCAI-01*.

[8] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7) :394–397, July 1962.

[9] Uwe Egly, Rainer Feldmann, and Hans Tompits, editors. *Proceedings of QBF2001 workshop at IJCAR'01*, Siena, Italy, June 2001.

[10] H. Fargier, J. Lang, and P. Marquis. Propositional logic and one-stage decision making. In *Proc. of the 7th Conference on Principles of Knowledge Representation and Reasoning*, pages 445–456, Breckenridge (CO), 2000.

[11] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Backjumping for quantified boolean logic satisfiability. In *Proceedings of IJCAI-01*, pages 275–281.

[12] R.M. Karp and R.J. Lipton. Some connections between non-uniform and uniform complexity classes. In *Proceedings of the Twelfth ACM Symposium on Theory of Computing (STOC'80)*, pages 302–309, 1980.

[13] H. Kleine-Büning, M. Karpinski, and A. Flögel. Resolution for quantified boolean formulas. *Information and computation*, 117(1) :12–18, 1995.

[14] Reinhold Letz. Advances in decision procedures for quantified boolean formulas. In Egly et al. [9], pages 55–64.

[15] H.R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the Association for Computing Machinery*, 25 :134–135, 1978.

[16] Chu-Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of IJCAI-97*, pages 366–371.

[17] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff : Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, June 2001.

[18] Ch. H. Papadimitriou. *Computational complexity*. Addison–Wesley, 1994.

[19] Jussi Rintanen. Improvements to the evaluation of quantified boolean formulae. In *Proceedings of IJCAI99*, pages 1192–1197.

[20] Jussi Rintanen. Partial implicit unfolding in the Davis-Putnam procedure for Quantified Boolean Formulae. In Egly et al. [9], pages 84–93.

[21] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3 :1–22, 1977.

[22] Poul F. Williams, Armin Biere, Edmund M. Clarke, and Anubhav Gupta. Combining Decision Diagrams and SAT Procedures for Efficient Symbolic Model Checking. In *Proceedings of CAV'00*, 2000.