Une nouvelle stratégie de mise sous forme prénexe pour des formules booléennes quantifiées avec bi-implications

Benoit da Mota, Igor Stéphan, Pascal Nicolas

Une nouvelle stratégie de mise sous forme prénexe pour des formules booléennes quantifiées avec bi-implications

Benoit Da Mota, Igor Stéphan, Pascal Nicolas

LERIA, Université d'Angers,
2 Boulevard Lavoisier
49045 Angers Cedex 01
{damota,stephan,pn}@info.univ-angers.fr

Résumé

La plupart des procédures pour résoudre le problème de validité des formules booléennes quantifiées prennent en entrée seulement des formules sous forme normale négative, voire sous forme normale conjonctive, et donc prénexes. Mais, il est rarement naturel d'exprimer un problème directement sous cette forme. Par exemple, en spécification, des symboles propositionnels existentiellement quantifiés sont insérés, selon un même motif, pour capturer des résultats intermédiaires. Ainsi, pour pouvoir utiliser les solveurs de l'état de l'art, il est nécessaire de convertir toute formule booléenne quantifiée sous forme prénexe. Un problème majeur de cette mise sous forme prénexe est qu'elle détruit complètement la structure originale de la formule. De plus, lors de la mise sous forme prénexe des bi-implications il y a duplication de leurs sous-formules, ceci incluant les quantificateurs. Cela conduit généralement à une croissance exponentielle de la taille de la formule. Dans ce travail, nous nous focalisons sur le motif très courant des résultats intermédiaires. Nous mettons en évidence des équivalences logiques qui permettent d'extraire les sous-formules améliorant ainsi nettement les performances des solveurs de l'état de l'art.

Mots-clés *: Formules Booléennes Quantifiées (QBF), Mise sous forme prénexe*

Abstract

Most of the recent and efficient decision procedures for quantified Boolean formulae accept only formulae in negation normal form as input or in an even more restrictive format such as conjunctive normal form. In all cases, formulas has to be prenex, but real problems are rarely expressed in such forms. For instance, in specification, intermediate propositional symbols are used to capture local results with

always the same pattern. So, in order to use state-of-the-art solvers the original formula has firstly to be converted in prenex form. A drawback of this preliminary step is to destroy completely the original structures of the formula. Furthermore, during the prenexing process, bi-implications are translated in such a way that there is a duplication of their sub-formulae including the quantifiers. In general, this process leads to an exponential growth of the formula. In this work, we focus on this very common pattern of intermediate result. We introduce new logical equivalences allowing us to extract these sub-formulae in a way that can improve the performance of the state-of-the-art quantified Boolean solvers.

***Key-words**: Quantified Boolean Formulae (QBF), Prenexing strategy*

1 INTRODUCTION

Le problème de validité pour les formules booléennes quantifiées (QBF) est une généralisation du problème de satisfiabilité pour les formules booléennes. Tandis que décider de la satisfiabilité des formules booléennes est NP-complet, décider de la validité des QBF est PSPACE-complet. C'est le prix à payer pour une représentation plus concise pour de très nombreuses classes de formules. D'importants problèmes de décision parmi des champs très divers ont des transformations polynomiales vers le problème de validité des QBF : la planification [24, 2], le «Bounded Model Construction» [2], la vérification formelle de circuit (voir [7] pour un panorama), les jeux finis à deux joueurs [21, 1, 25] et la compilation de langages logiques non-monotones [17, 16, 9, 8] ou d'autre formalismes de même classe de complexité tels que les « diagrammes d'influence possibilistes » (pour le calcul de la stratégie pessimiste optimale [20]).

La plupart des procédures actuelles pour décider des QBF nécessitent d'avoir en entrée une formule sous forme normale négative voire sous forme plus restrictive comme la forme normale conjonctive. Or, les problèmes sont rarement exprimés directement sous cette forme qui détruit complètement la structure originale du problème. Il est plus naturel de les représenter en utilisant la richesse du langage et donc à l'aide d'opérateurs plus expressifs (incluant l'implication, la bi-implication et le ou-exclusif) et avec des quantificateurs à l'intérieur des formules. Ainsi la mise sous forme normale négative nécessite pour les QBF, comme pour les formules de la logique des prédicats, cinq étapes : (i) le remplacement des bi-implications et des ou-exclusifs par leurs définitions avec des implications, des négations, des conjonctions et/ou des disjonctions ; (ii) le renommage des symboles propositionnels de telle manière que des quantificateurs distincts lient des symboles propositionnels distincts ; (iii) l'extraction des quantificateurs ; (iv) le remplacement des implications par leurs définitions avec des négations, des conjonctions et/ou des disjonctions ; (v) l'application des lois de de Morgan. La mise sous forme normale conjonctive nécessite une étape supplémentaire à la mise sous

forme normale négative ; elle a été largement étudiée [23, 14, 15] puisque cette forme normale est aussi le format d'entrée de bon nombre des solveurs pour le problème SAT, de satisfiabilité d'une formule booléenne (non quantifiée). L'ensemble des transformations précédant la soumission d'un problème à un solveur QBF est décrit dans la figure 1.

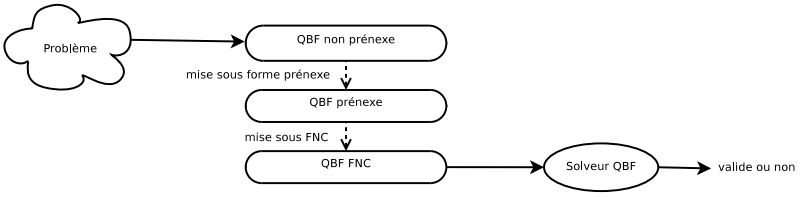

FIG. 1 – Étapes menant à la décision d'un problème via les QBF.

Les trois premières étapes de ces mises sous forme normale sont réunies sous le terme de mise sous forme prénexe. Il n'existe pas une unique forme prénexe associée à une QBF et selon la stratégie choisie pour l'ordre d'extraction des quantificateurs, le temps de résolution peut être fortement influencé [18]. Mais, autant que nous sachions, aucune stratégie n'est fournie pour extraire les quantificateurs de formules booléennes quantifiées contenant des bi-implications et des ou-exclusifs. C'est un problème important puisque l'élimination classique des bi-implications duplique les quantificateurs nichés dans leurs sous-formules conduisant à une croissance exponentielle de leur nombre et de la taille de la formule.

L'article est organisé ainsi : en section 2, nous présentons des préliminaires sur la logique propositionnelle et les formules booléennes quantifiées ; en section 3, nous analysons le lien entre la mise sous forme prénexe et la bi-implication, nous proposons une stratégie pour extraire le motif très récurrent du résultat intermédiaire, nous présentons une famille d'exemples théoriques et une famille d'exemples pratiques dans le but d'évaluer notre stratégie ; en section 4, nous décrivons quelques résultats expérimentaux pour notre stratégie sur nos deux familles d'exemples pour un ensemble de solveurs QBF de l'état de l'art ; en section 5, nous concluons et traçons quelques perspectives.

2 PRÉLIMINAIRES

2.1 Logique propositionnelle

L'ensemble des valeurs booléennes **vrai** et **faux** est noté $\mathbf{BOOL}$. L'ensemble des symboles propositionnels est noté $\mathcal{SP}$. Les symboles $\top$ et $\bot$ sont les constantes booléennes. Le symbole $\wedge$ est utilisé pour la conjonction, $\vee$ pour la disjonction, $\neg$ pour la négation, $\rightarrow$ pour l'implication, $\leftrightarrow$ pour la bi-implication et $\oplus$ pour le ou-exclusif. L'ensemble des opérateurs

binaires $\{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ est noté $\mathcal{O}$. L'ensemble **PROP** des formules propositionnelles est défini inductivement ainsi : tout symbole propositionnel ou constante propositionnelle est élément de **PROP** ; si F est élément de **PROP** alors $\neg F$ est élément de **PROP** ; si F et G sont éléments de **PROP** et $\circ$ est élément de $\mathcal{O}$ alors $(F \circ G)$ est élément de **PROP**. Un littéral est un symbole propositionnel ou la négation de celui-ci. Une clause est une disjonction de littéraux. Une formule propositionnelle est sous forme normale négative (FNN) si la formule n'est constituée exclusivement que de conjonctions, disjonctions et littéraux. Une formule propositionnelle est sous forme normale conjonctive (FNC) si c'est une conjonction de disjonctions de littéraux. Toute formule sous FNC est une formule sous FNN. Une valuation v est une fonction de $\mathcal{SP}$ dans $\mathbf{BOOL}$ (l'ensemble des valuation est noté **VALUATION**).

2.2 Syntaxe des formules booléennes quantifiées

Le symbole $\exists$ est utilisé pour la quantification existentielle et $\forall$ pour la quantification universelle (q est utilisé pour noter un quantificateur quelconque). L'ensemble **QBF** des formules booléennes quantifiées est défini inductivement ainsi : si F est un élément de **PROP** alors c'est un élément de **QBF** ; si F est un élément de **QBF** et x est un symbole propositionnel alors $(\exists x\ F)$ et $(\forall x\ F)$ sont des éléments de **QBF** ; si F est élément de **QBF** alors $\neg F$ est élément de **QBF** ; si F et G sont éléments de **QBF** et $\circ$ est élément de $\mathcal{O}$ alors $(F \circ G)$ est élément de **QBF**. Un symbole propositionnel x est libre s'il n'apparaît pas sous la portée d'un quantificateur $\exists x$ ou $\forall x$. L'ensemble des symboles propositionnels libres d'une QBF F est noté $\mathcal{SPL}(F)$. Une QBF est close si elle ne contient pas de symbole propositionnel libre. Une substitution est une fonction de l'ensemble des symboles propositionnels dans l'ensemble des formules (quantifiées ou non). Nous définissons la substitution de x par F dans G, notée $[x \leftarrow F](G)$, comme étant la formule obtenue de G en remplaçant toutes les occurrences du symbole propositionnel x par la formule F sauf pour les occurrences de x sous la portée d'un quantificateur portant sur x. Un lieur est une chaîne de caractère $q_1 x_1 \ldots q_n x_n$ avec $x_1, \ldots, x_n$ des symboles propositionnels distincts et $q_1 \ldots q_n$ des quantificateurs. Une QBF QF est sous forme prénexe si Q est un lieur et F est une formule booléenne, appelée matrice. Une QBF prénexe QF est sous forme normale négative (resp. conjonctive) si F est une formule booléenne en FNN (resp. FNC).

2.3 Sémantique des formules booléennes quantifiées

La sémantique des QBF présentée ci-dessous s'inspire des sémantiques de la logique propositionnelle et des prédicats présentées dans [19] ; elle fait appel à la sémantique des (constantes et) opérateurs booléens qui est définie de manière habituelle, en particulier, à chaque (constante et) opérateur (resp.

$\top$, $\bot$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) est associée une fonction booléenne (resp. $i_\top, i_\bot :\rightarrow \mathbf{BOOL}$, $i_\neg : \mathbf{BOOL} \rightarrow \mathbf{BOOL}$, $i_\wedge, i_\vee, i_\rightarrow, i_\leftrightarrow : \mathbf{BOOL} \times \mathbf{BOOL} \rightarrow \mathbf{BOOL}$) qui en définit sa sémantique ; la sémantique des QBF fait aussi appel à une sémantique pour les quantificateurs :

$$i^x_\exists, i^x_\forall : (\mathbf{VALUATION} \rightarrow \mathbf{BOOL}) \times \mathbf{VALUATION} \rightarrow \mathbf{BOOL}$$

$$i^x_\exists(f)(v) = i_\vee(f(v[x := \mathbf{vrai}]), f(v[x := \mathbf{faux}]))$$

$$i^x_\forall(f)(v) = i_\wedge(f(v[x := \mathbf{vrai}]), f(v[x := \mathbf{faux}]))$$

enfin elle est définie inductivement :

- $[[\bot]](v) = i_\bot$;
- $[[\top]](v) = i_\top$;
- $[[x]](v) = v(x)$ si $x \in \mathcal{SP}$;
- $[[(F \circ G)]](v) = i_\circ([[F]](v), [[G]](v))$ si $F, G \in$ **QBF** et $\circ \in \mathcal{O}$;
- $[[\neg F]](v) = i_\neg([[F]](v))$ si $F \in$ **QBF** ;
- $[[(\exists x\ F)]](v) = i^x_\exists([[F]])(v)$ si $F \in$ **QBF** ;
- $[[(\forall x\ F)]](v) = i^x_\forall([[F]])(v)$ si $F \in$ **QBF**.

Une QBF close F est valide si $[[F]](v) = \mathbf{vrai}$ pour toute valuation v. Par exemple la QBF $\exists a \exists b \forall c((a \vee b) \leftrightarrow c)$ n'est pas valide tandis que la QBF $\forall c \exists a \exists b((a \vee b) \leftrightarrow c)$ l'est. Cet exemple montre que l'ordre des quantificateurs est crucial pour décider de la validité d'une QBF.

Comme dans le cas propositionnel, une relation d'équivalence notée $\equiv$ est définie pour les QBF par $F \equiv G$ si $[[F]](v) = [[G]](v)$ pour toute valuation v. En lien avec l'exemple qui précède, $\exists x \exists y F \equiv \exists y \exists x F$ et $\forall x \forall y F \equiv \forall y \forall x F$ mais $\exists a \exists b \forall c((a \vee b) \leftrightarrow c) \not\equiv \forall c \exists a \exists b((a \vee b) \leftrightarrow c)$.

Enfin, rappelons que le problème (SAT) consistant à décider si une formule booléenne est satisfiable ou non est le problème canonique de la classe NP-complet. De son côté, le problème consistant à décider si une formule booléenne quantifiée est valide ou non est le problème canonique de la classe PSPACE-complet [26].

3 FORME PRÉNEXE ET BI-IMPLICATIONS

3.1 Motivations

Dans [18], les auteurs définissent des stratégies pour la mise sous forme prénexe selon l'ordre d'extraction des quantificateurs et montrent expérimentalement que cela peut avoir une forte influence sur le temps de résolution nécessaire aux différentes procédures. Cependant, il n'existe pas de règle pour extraire des quantificateurs de la bi-implication (ou du ou-exclusif).

Comme suggéré dans l'introduction, la mise sous forme prénexe se décompose en trois étapes que nous détaillons ici :

1. Le remplacement de la bi-implication et du ou-exclusif par leurs définitions selon l'implication, la négation, la conjonction et/ou la disjonction est généralement réalisé par les deux équivalences suivantes :
   1) $(A \leftrightarrow B) \equiv ((A \rightarrow B) \wedge (B \rightarrow A))$ 2) $(A \oplus B) \equiv ((A \vee B) \wedge \neg(A \wedge B))$

   Chacun peut constater que les formules A et B sont dupliquées.
2. Le renommage des symboles propositionnels tel que des quantificateurs distincts lient des symboles propositionnels distincts peut être réalisé avant l'étape d'extraction des quantificateurs mais peut aussi être interfolié avec celle-ci.
3. L'extraction des quantificateurs est généralement basée sur les équivalences suivantes, héritées de la logique classique du premier ordre, elles restent vraies pour les QBF (F, G et H sont des QBF et x est un symbole propositionnel tel que $x \notin \mathcal{SPL}(H)$)

3) $(\exists x\ \neg F) \equiv \neg(\forall x\ F)$ 4) $(\forall x\ \neg F) \equiv \neg(\exists x\ F)$
5) $(\forall x\ F) \equiv F$, if $x \notin \mathcal{SPL}(F)$ 6) $(\exists x\ F) \equiv F$, if $x \notin \mathcal{SPL}(F)$
7) $(\forall x\ (F \wedge H)) \equiv ((\forall x\ F) \wedge H)$ 8) $(\forall x\ (F \vee H)) \equiv ((\forall x\ F) \vee H)$
9) $(\exists x\ (F \wedge H)) \equiv ((\exists x\ F) \wedge H)$ 10) $(\exists x\ (F \vee H)) \equiv ((\exists x\ F) \vee H)$
11) $(\forall x\ (F \wedge G)) \equiv ((\forall x\ F) \wedge (\forall x\ G))$ 12) $(\exists x\ (F \vee G)) \equiv ((\exists x\ F) \vee (\exists x\ G))$
13) $(\forall x\ (F \rightarrow H)) \equiv ((\exists x\ F) \rightarrow H)$ 14) $(\exists x\ (F \rightarrow H)) \equiv ((\forall x\ F) \rightarrow H)$
15) $(\forall x\ (H \rightarrow G)) \equiv (H \rightarrow (\forall x\ G))$ 16) $(\exists x\ (H \rightarrow G)) \equiv (H \rightarrow (\exists x\ G))$

Les équivalences 13 à 16 pour l'implication sont aisément déductibles des équivalences 3 à 12. L'extraction des quantificateurs est un processus qui consiste à appliquer ces équivalences de la droite vers la gauche jusqu'à obtenir un point fixe correspondant à une QBF sous forme prénexe. La mise sous forme prénexe conserve la validité mais ne garantit ni de garder la taille de la formule ni l'espace de recherche associé. Le pouvoir d'expression de la bi-implication s'étend bien au delà de la simple équivalence logique entre $(F \leftrightarrow G)$ et $((G \rightarrow F) \wedge (F \rightarrow G))$: un quantificateur présent dans F ou G, de part les équivalences logiques 13 à 16, sera extrait aussi bien sous sa forme universelle que sous sa forme existentielle, et ce quelle que soit sa forme initiale. Si cela n'a aucun impact vis-à-vis de la validité, il n'en est pas de même vis-à-vis du calcul. Nous montrons dans les exemples suivants qu'à la fin des trois étapes de la mise sous forme prénexe, l'augmentation de la taille de la formule n'est pas le seul problème, même pour une formule très simple.

Soit a un symbole propositionnel, ϕ, ϕ_1 et ϕ_2 des formules booléennes quantifiées telles que $\phi = (\phi_1 \leftrightarrow \exists a(\phi_2))$ et $a \notin \mathcal{SPL}(\phi_1)$. Dans un premier temps il faut exprimer la bi-implication à l'aide de la conjonction et de l'implication : $\phi \overset{1}{\equiv} ((\phi_1 \rightarrow \exists a(\phi_2))) \wedge ((\exists a\ \phi_2) \rightarrow \phi_1))$. Les formules ϕ_1 et ϕ_2 ont été dupliquées. Ensuite, il faut extraire les quantificateurs des implications : $\phi \overset{16,13}{\equiv} ((\exists a\ (\phi_1 \rightarrow \phi_2)) \wedge (\forall a\ (\phi_2 \rightarrow \phi_1)))$. Il faut renommer une des occurrences du symbole propositionnel a afin d'extraire les quantificateurs de la conjonction : soit b un nouveau symbole propositionnel et

$\phi'_2 = [a \leftarrow b](\phi_2)$ alors : $\phi \overset{9,7}{\equiv} (\exists a\ (\forall b\ ((\phi_1 \rightarrow \phi_2) \wedge (\phi'_2 \rightarrow \phi_1))))$. L'ordre d'extraction des quantificateurs aurait pu être inversé, nous aurions obtenu : $\phi \overset{7,9}{\equiv} (\forall a\ (\exists b\ ((\phi_1 \rightarrow \phi_2) \wedge (\phi'_2 \rightarrow \phi_1))))$. Dans le cas général, si F est une formule booléenne quantifiée, $(\forall y\ (\exists x\ F))$ n'est pas équivalente à $(\exists x\ (\forall y\ F))$. Le fait de savoir que les quantificateurs peuvent s'inverser contrairement au cas général est une information importante qui pourrait être exploitée par les procédures de décision pour le problème QBF. Les symboles propositionnels a et b représentent le même symbole propositionnel de la formule non prénexe mais rien dans la formule prénexe ne semble les mettre en relation.

Pour conclure cette partie sur les motivations, nous envisageons le cas où un quantificateur doit traverser plusieurs bi-implications. Nous montrons dans l'exemple suivant, que le choix de la position des parenthèses peut influer sur la taille de la formule mise sous forme prénexe. Soit a un symbole propositionnel, ϕ_d, ϕ_g, ϕ_1, ϕ_2 et ϕ_3 des formules booléennes quantifiées telles que $\phi_g = ((\phi_1 \leftrightarrow \phi_2) \leftrightarrow (\exists a\ \phi_3))$, $\phi_d = (\phi_1 \leftrightarrow (\phi_2 \leftrightarrow (\exists a\ \phi_3)))$ avec $a \notin \mathcal{SPL}(\phi_1)$ et $a \notin \mathcal{SPL}(\phi_2)$ alors $\phi_d \equiv \phi_g$ par associativité de la bi-implication. Nous mettons ϕ_g sous forme prénexe :

$$\begin{array}{lcl}
\phi_g & \overset{1}{\equiv} & (((\phi_1 \leftrightarrow \phi_2) \rightarrow (\exists a\ \phi_3)) \wedge ((\exists a\ \phi_3) \rightarrow (\phi_1 \leftrightarrow \phi_2))) \\
\phi_g & \overset{16,13,9,7}{\equiv} & \exists a \forall b (((\phi_1 \leftrightarrow \phi_2) \rightarrow \phi_3) \wedge ([a \leftarrow b](\phi_3) \rightarrow (\phi_1 \leftrightarrow \phi_2)))
\end{array}$$

puis ϕ_d :

$$\begin{array}{lcl}
\phi_d & \overset{1}{\equiv} & (\phi_1 \leftrightarrow ((\phi_2 \rightarrow (\exists a\ \phi_3)) \wedge ((\exists a\ \phi_3) \rightarrow \phi_2))) \\
\phi_d & \overset{1}{\equiv} & ((\phi_1 \rightarrow ((\phi_2 \rightarrow (\exists a\ \phi_3)) \wedge ((\exists a\ \phi_3) \rightarrow \phi_2))) \wedge \\
 & & (((\phi_2 \rightarrow (\exists a\ \phi_3)) \wedge ((\exists a\ \phi_3) \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{13,14,15,16,9,7}{\equiv} & \exists a \exists c \forall b \forall d \\
 & & ((\phi_1 \rightarrow ((\phi_2 \rightarrow \phi_3) \wedge ([a \leftarrow b](\phi_3) \rightarrow \phi_2))) \wedge \\
 & & (((\phi_2 \rightarrow [a \leftarrow d](\phi_3)) \wedge ([a \leftarrow c](\phi_3) \rightarrow \phi_2)) \rightarrow \phi_1))
\end{array}$$

Les formules ϕ_g et ϕ_d, bien qu'équivalentes, n'ont pas du tout la même taille ni le même nombre de symboles propositionnels une fois mises sous forme prénexe. Maintenant, si la formule considérée est

$$(\phi_1 \leftrightarrow (\phi_2 \leftrightarrow \ldots (\phi_n \leftrightarrow (\exists a\ \phi_{n+1}))))$$

alors 2^n symboles propositionnels sont créés dont la moitié est quantifiée universellement. Les formules ϕ_n et ϕ_{n+1} sont recopiées 2^n fois, la formule ϕ_{n-1} est recopiée 2^{n-1} fois, et ainsi de suite, jusqu'à ϕ_1 qui est recopiée 2 fois. La taille de la formule et le nombre de symboles propositionnels croissent exponentiellement avec le nombre de bi-implications traversées. Si chacune des ϕ_k possède un quantificateur à extraire, il sera impossible d'éviter le pire cas.

3.2 Forme prénexe et résultats intermédiaires

La section précédente a montré l'importance du problème de la mise sous forme prénexe lorsque des quantificateurs apparaissent dans les bi-implications. Puisque $(A \oplus B) \equiv \neg(A \leftrightarrow B) \equiv (\neg A \leftrightarrow B)$, dans ce qui suit, nous ne traiterons que des bi-implications. Autant que nous sachions, ce problème est généralement occulté dans le processus de traduction du problème spécifié en une QBF sous FNC équivalente donnée en entrée à une procédure de décision QBF. Dans cette section, nous portons notre attention sur deux cas très fréquents qui apparaissent en programmation ou en spécification : la déclaration de résultats intermédiaires et la déclaration de domaine. Le premier cas est basé sur l'introduction d'un symbole propositionnel existentiellement quantifié en association avec une conjonction dans le but d'améliorer la lisibilité de la spécification ou de capturer les résultats intermédiaires d'un calcul présent en de maints endroits. Pour les QBF, par extension des résultats propositionnels classiques [27], nous nous intéressons au motif $(\exists x\ ((x \leftrightarrow F) \wedge G))$, x symbole propositionnel intermédiaire absent de F, qui est équivalent à $[x \leftarrow F](G)$. Le second cas est basé sur l'introduction d'un symbole propositionnel quantifié universellement en association avec une implication pour exprimer le domaine du symbole propositionnel (ie : une propriété à vérifier par le symbole). Dans les deux cas, nous appelons $(x \leftrightarrow F)$ la définition de x. Le résultat suivant démontre qu'en fait les deux techniques sont équivalentes dans le cas des QBF.

Théorème 1
Soient F et G deux QBF et x un symbole propositionnel qui représente un resultat intermédiaire F, avec $x \notin \mathcal{SPL}(F)$, alors

$$(\exists x\ ((x \leftrightarrow F) \wedge G)) \equiv (\forall x\ ((x \leftrightarrow F) \rightarrow G)).$$

Preuve 1

$$\begin{array}{ll}
 & (\exists x\ ((F \leftrightarrow x) \wedge G)) \\
\equiv & (((F \leftrightarrow \top) \wedge [x \leftarrow \top](G)) \vee ((F \leftrightarrow \bot) \wedge [x \leftarrow \bot](G))) \\
\stackrel{*}{\equiv} & (((F \leftrightarrow \top) \vee [x \leftarrow \bot](G)) \wedge ((F \leftrightarrow \bot) \vee [x \leftarrow \top](G))) \\
\equiv & ((\neg(F \leftrightarrow \top) \rightarrow [x \leftarrow \bot](G)) \wedge (\neg(F \leftrightarrow \bot) \rightarrow [x \leftarrow \top](G))) \\
\equiv & (((F \leftrightarrow \bot) \rightarrow [x \leftarrow \bot](G)) \wedge ((F \leftrightarrow \top) \rightarrow [x \leftarrow \top](G))) \\
\equiv & ((\forall x\ (F \leftrightarrow x)) \rightarrow G)
\end{array}$$

car $((\neg A \vee B) \wedge (A \vee C) \wedge (B \vee C)) \stackrel{*}{\equiv} ((\neg A \vee B) \wedge (A \vee C))$. □

Par le théorème précédent, nous nous focalisons uniquement sur le motif existentiel.

Puisque les solveurs prennent en entrée généralement des QBF sous FNC, le statut spécial des symboles propositionnels intermédiaires est totalement perdu et ils sont traités comme les symboles propositionnels du problème

initial. Une manière simple de se séparer de ces symboles intermédiaires est d'appliquer l'équivalence déjà introduite $(\exists x\ ((x \leftrightarrow F) \wedge G)) \equiv [x \leftarrow F](G)$ mais cela peut conduire à une croissance exponentielle de la taille de la formule.

À la place, nous proposons d'extraire ces symboles propositionnels grâce au théorème suivant.

Théorème 2
Soient F, G et H trois QBF et x un symbole propositionnel qui représente le résultat intermédiaire F, avec $x \notin \mathcal{SPL}(H)$, $x \notin \mathcal{SPL}(F)$ alors

$$(H \leftrightarrow (\exists x\ ((x \leftrightarrow F) \wedge G))) \equiv (\exists x\ ((x \leftrightarrow F) \wedge (H \leftrightarrow G))).$$

Preuve 2
La preuve pour le ou exclusif est similaire.

$$\begin{array}{ll}
& H \leftrightarrow (\exists x((F \leftrightarrow x) \wedge G)) \\
\equiv & ((H \rightarrow (\exists x\ ((F \leftrightarrow x) \wedge G))) \wedge ((\exists x\ ((F \leftrightarrow x) \wedge G)) \rightarrow H)) \\
\equiv & \exists x \forall x'((H \rightarrow ((F \leftrightarrow x) \wedge G)) \wedge (((F \leftrightarrow x') \wedge [x \leftarrow x'](G)) \rightarrow H)) \\
\equiv & \exists x \forall x'((\neg H \vee ((F \leftrightarrow x) \wedge G)) \wedge (\neg((F \leftrightarrow x') \wedge [x \leftarrow x'](G)) \vee H)) \\
\equiv & \exists x \forall x'((\neg H \wedge \neg((F \leftrightarrow x') \wedge [x \leftarrow x'](G))) \vee (((F \leftrightarrow x) \wedge G) \wedge H)) \\
\equiv & ((\forall x'\ (\neg H \wedge (\neg(F \leftrightarrow x') \vee \neg[x \leftarrow x'](G))))\vee \\
& (\exists x\ (((F \leftrightarrow x) \wedge G) \wedge H))) \\
\equiv & (((\neg H \wedge (\neg(F \leftrightarrow \bot) \vee \neg[x \leftarrow x'][x' \leftarrow \bot](G)))\wedge \\
& (\neg H \wedge (\neg(F \leftrightarrow \top) \vee \neg[x \leftarrow x'][x' \leftarrow \top](G))))\vee \\
& ((((F \leftrightarrow \bot) \wedge [x \leftarrow \bot](G)) \wedge H) \vee (((F \leftrightarrow \top) \wedge [x \leftarrow \top](G)) \wedge H))) \\
\equiv & ((((\neg H \wedge \neg(F \leftrightarrow \bot)) \vee (\neg H \wedge \neg[x \leftarrow \bot](G)))\wedge \\
& ((\neg H \wedge \neg(F \leftrightarrow \top)) \vee (\neg H \wedge \neg[x \leftarrow \top](G))))\vee \\
& ((((F \leftrightarrow \bot) \wedge [x \leftarrow \bot](G)) \wedge H) \vee (((F \leftrightarrow \top) \wedge [x \leftarrow \top](G)) \wedge H))) \\
\equiv & (((\neg H \wedge \neg(F \leftrightarrow \bot)) \wedge (\neg H \wedge \neg(F \leftrightarrow \top)))\vee \\
& ((\neg H \wedge \neg[x \leftarrow \bot](G)) \wedge (\neg H \wedge \neg(F \leftrightarrow \top)))\vee \\
& ((\neg H \wedge \neg(F \leftrightarrow \bot)) \wedge (\neg H \wedge \neg[x \leftarrow \top](G)))\vee \\
& ((\neg H \wedge \neg[x \leftarrow \bot](G)) \wedge (\neg H \wedge \neg[x \leftarrow \top](G)))\vee \\
& ((((F \leftrightarrow \bot) \wedge [x \leftarrow \bot](G)) \wedge H) \vee (((F \leftrightarrow \top) \wedge [x \leftarrow \top](G)) \wedge H))) \\
\equiv & (((\neg H \wedge \neg[x \leftarrow \bot](G)) \wedge (F \leftrightarrow \bot))\vee \\
& ((\neg H \wedge \neg[x \leftarrow \top](G)) \wedge (F \leftrightarrow \top))\vee \\
& ((H \wedge [x \leftarrow \bot](G)) \wedge (F \leftrightarrow \bot))\vee \\
& ((H \wedge [x \leftarrow \top](G)) \wedge (F \leftrightarrow \top))) \\
\equiv & (\exists x\ (((\neg H \wedge \neg G) \wedge (F \leftrightarrow x)) \vee ((H \wedge G) \wedge (F \leftrightarrow x)))) \\
\equiv & (\exists x\ ((F \leftrightarrow x) \wedge ((\neg H \wedge \neg G) \vee (H \wedge G)))) \\
\equiv & (\exists x\ ((F \leftrightarrow x) \wedge (H \leftrightarrow G)))
\end{array}$$

□

3.3 Mise sous forme prénexe et résultats intermédiaires

Les quatre algorithmes suivant appliquent ensemble et récursivement ce théorème à toutes les bi-implications satisfaisant le motif.

Algorithme 1 $recherche$

Entrée: Une QBF F
Entrée: Un ensemble de symboles propositionnels S_{sp}
Sortie: Un ensemble de définitions
  selon F **faire**
   cas $(\exists x\ G)$
   retourner $recherche(G, S_{sp} \cup \{x\})$
   cas $(G \wedge H)$
   retourner $recherche(G, S_{sp}) \cup recherche(H, S_{sp})$
   cas $(x \leftrightarrow G)$
   si $x \in S_{sp}$ **alors**
    retourner $\{(x \leftrightarrow G)\}$
   sinon
    retourner $\emptyset$
   fin si
   par défaut
   retourner $\emptyset$
  fin selon

Soit $(\phi_H \leftrightarrow \phi)$ une QBF, l'algorithme $recherche$ recherche toute définition satisfaisant le motif $(\exists x\ ((x \leftrightarrow F) \wedge G))$ dans ϕ tout en relâchant la contrainte $x \notin \mathcal{SPL}(F)$. En fait, le motif peut être encastré dans une succession de motifs similaires grâce à l'équivalence 9, l'équivalence $(\exists x\ (\exists y\ F)) \equiv (\exists y\ (\exists x\ F))$, ainsi que l'associativité et la commutativité de la conjonction. Par exemple

$recherche((\phi_1 \wedge (\exists a\ (\exists b\ ((a \leftrightarrow \phi_a) \wedge ((b \leftrightarrow \phi_b) \wedge \phi_2))))), \emptyset) =$
$\{(a \leftrightarrow \phi_a), (b \leftrightarrow \phi_b)\}$.

Dans ce qui suit, nous dénotons $\{(a \leftrightarrow \phi_a), (b \leftrightarrow \phi_b)\}$ par S_d.

Algorithme 2 $extrait$

Entrée: Un ensemble de définitions S_d
Entrée: Un ensemble de symboles propositionnels S_{sp}
Sortie: Une liste de définitions
  $L_d := nil$
  tant que il existe $(e \leftrightarrow d) \in S_d$ tel que $\mathcal{SPL}(d) \subseteq S_{sp}$ **faire**
   $S_d := S_d$ privé de l'ensemble des définitions sur e
   $S_{sp} := S_{sp} \cup \{e\}$
   $L_d := \cdot((e \leftrightarrow d), L_d)$
  fin tant que
  retourner L_d

L'algorithme $extrait$ applique la contrainte $x \notin \mathcal{SPL}(F)$ à l'ensemble des définitions obtenu par l'algorithme $recherche$ grâce à un tri topologique qui extrait une liste de définitions. Toutes les définitions de l'ensemble ne

sont pas insérées dans la liste : par exemple si $a \in \mathcal{SPL}(\phi_b)$, $b \in \mathcal{SPL}(\phi_a)$ et S_{sp} est un ensemble de symboles propositionnels tel que $a \notin S_{sp}$ et $b \notin S_{sp}$ alors $extrait(S_d, S_{sp}) = nil$ sinon si $a \notin \mathcal{SPL}(\phi_b)$ mais $b \in \mathcal{SPL}(\phi_a)$ alors

$$extrait(S_d, S_{sp}) = \cdot((b \leftrightarrow \phi_b), \cdot((a \leftrightarrow \phi_a), nil))$$

puisque

$$\begin{array}{l}(\phi_1 \wedge (\exists a\ (\exists b\ ((a \leftrightarrow \phi_a) \wedge ((b \leftrightarrow \phi_b) \wedge \phi_2)))))\\ \equiv \quad (\exists b\ ((b \leftrightarrow \phi_b) \wedge (\exists a\ ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge \phi_2)))))\end{array}$$

Algorithme 3 *applique*

Entrée: Une QBF F
Entrée: Une liste de définitions L_d
Sortie: Une QBF

Une liste de définitions $L_{temp} := L_d$
tant que non $vide(L_d)$ **faire**
$(e \leftrightarrow d) := tête(L_d)$
$F := ((e \leftrightarrow d) \wedge [(e \leftrightarrow d) \leftarrow \top](F))$
$L_d := queue(L_d)$
fin tant que
tant que non $vide(L_{temp})$ **faire**
$(e \leftrightarrow d) := tête(L_{temp})$
élimination de Fdu quantificateur existentiel portant sur e
$F := (\exists e\ F)$
$L_{temp} := queue(L_{temp})$
fin tant que
retourner F

L'algorithme *applique* applique effectivement le théorème 2 en deux étapes sur la QBF ϕ pour la liste de définitions extraites par l'algorithme $extrait$. Premièrement, les définitions sont remplacées par $\top$ dans la QBF puisque les définitions sont connectées conjonctivement et sont réintroduites à l'extérieur de la QBF : par exemple $(\phi_1 \wedge (\exists a\ (\exists b\ ((a \leftrightarrow \phi_a) \wedge ((b \leftrightarrow \phi_b) \wedge \phi_2)))))$ est remplacé par

$$\begin{array}{l}((b \leftrightarrow \phi_b) \wedge ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge (\exists a\ (\exists b\ (\top \wedge (\top \wedge \phi_2)))))))\\ \equiv \quad ((b \leftrightarrow \phi_b) \wedge ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge (\exists a\ (\exists b\ \phi_2))))).\end{array}$$

Deuxièmement, les quantificateurs existentiels pour les symboles propositionnels de la liste de définitions sont éliminés de la formule et réintroduits à l'extérieur de celle-ci : par exemple, $((b \leftrightarrow \phi_b) \wedge ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge (\exists a\ (\exists b\ \phi_2)))))$ est remplacé par

$$\begin{array}{l}(\exists b\ (\exists a\ ((b \leftrightarrow \phi_b) \wedge ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge \phi_2)))))\\ \equiv \quad (\exists b\ ((b \leftrightarrow \phi_b) \wedge (\exists a\ ((a \leftrightarrow \phi_a) \wedge (\phi_1 \wedge \phi_2))))).\end{array}$$

Algorithme 4 $rec\_def\_extract$

Entrée: Une QBF F
Entrée: Un ensemble de symboles propositionnels S_{sp}
Sortie: Une QBF équivalente à F avec ses définitions extraites

```
selon F faire
  cas (qx G)
  retourner (qx rec_def_extract(G, {x} ∪ S_sp))
  cas ¬G
  retourner ¬rec_def_extract(G, S_sp)
  cas (G ∘ H) et ∘ ∈ {∧, ∨, →}
  retourner (rec_def_extract(G, S_sp) ∘ rec_def_extract(H, S_sp))
  cas (G ↔ H)
  Une QBF G' := rec_def_extract(G, S_sp)
  Une QBF H' := rec_def_extract(H, S_sp)
  Une liste de définitions L_d := recherche(G', ∅) ∪ recherche(H', ∅)
  retourner applique((G' ↔ H'), extrait(L_d, S_sp))
  par défaut
  retourner F
fin selon
```

L'algorithme $rec\_def\_extract$ applique récursivement l'extraction des motifs sur toutes les définitions possibles dans un processus allant de l'intérieur vers l'extérieur de la formule. Ainsi, des définitions peuvent franchir plusieurs bi-implications.

Il n'est pas difficile de prouver grâce au théorème 2 le théorème de correction suivant.

Théorème 3
Soit F une QBF alors $rec\_def\_extract(F, \emptyset) \equiv F$.

Preuve 3
Soit F une QBF alors, par induction sur la structure, $recherche(F, \emptyset)$ est un ensemble de définitions de F qui sont telles qu'elles ne sont séparées de la racine de la formule F (considérée comme un arbre) que par des nœuds étiquetés par des existentielles ou des conjonctions.

Soient F et G deux QBF alors, par point fixe sur l'ordre d'inclusion des ensembles, $extrait(recherche(F, \emptyset) \cup recherche(G, \emptyset), S)$ est une liste $[(e_1 \leftrightarrow d_1), \ldots, (e_n \leftrightarrow d_n)]$ telle que (par la propriété ci-dessus) chaque $(e_i \leftrightarrow d_i)$ est une définition de $(F \leftrightarrow G)$ séparée de la racine de la formule uniquement par des nœuds étiquetés par des existentielles ou des conjonctions et pour tout e_i, $1 \leq i \leq n$, $\mathcal{SPL}(d_i) \subseteq S \cup \{e_1, \ldots, e_n\}$.

Soient F et G deux QBF alors, par récurrence sur le nombre d'éléments de la liste $extrait(recherche(F, \emptyset) \cup recherche(G, \emptyset), S)$, le théorème 2, l'équivalence 9 et l'équivalence $(\exists x\ (\exists y\ H)) \equiv (\exists y\ (\exists x\ H))$, H une QBF

quelconque (avec $S_{sp} = \mathcal{SPL}((F \leftrightarrow G))$*) :*
$applique((F \leftrightarrow G), extrait(recherche(F, \emptyset) \cup recherche(G, \emptyset), S_{sp})) \equiv (F \leftrightarrow G)$.

Enfin, par induction sur la structure et par la propriété exprimée ci-dessus, $rec\_def\_extract(F, \emptyset) \equiv F$ *pour une QBF* F *quelconque.* □

3.4 Chaînes de bi-implications

Nous définissons un exemple théorique pour évaluer l'impact de notre stratégie sur une chaîne de bi-implications : $th_n = (\xi_1 \leftrightarrow (\xi_2 \leftrightarrow ...(\xi_{n-1} \leftrightarrow (\xi_n))))$ dans laquelle chaque ξ_k contient un résultat intermédiaire comme cela a été proposé dans la section 3.1 sur les motivations. Nous avons choisi des ξ_k de la forme $\exists x_k((x_k \leftrightarrow (c \vee b)) \wedge (x_k \wedge a))$, avec x_k le résultat intermédiaire et a, b et c des symboles propositionnels présents dans d'autres maillons de la chaîne de bi-implications. La QBF th_n, pour $n \geq 4$ et $m = n - 2$ est de la forme :

$$
\begin{array}{l}
th_n = \exists e_0 ... \exists e_m \forall u_0 \forall u_1 \forall u_2 \\
(\ (\exists x_n((x_n \leftrightarrow (e_{m-2} \vee e_{m-1})) \wedge (x_n \wedge e_m))) \leftrightarrow \\
\quad (\ (\exists x_{n-1}((x_{n-1} \leftrightarrow (e_{m-3} \vee e_{m-2})) \wedge (x_{n-1} \wedge e_{m-1}))) \leftrightarrow \\
\quad\quad ... \\
\quad\quad\quad (\ (\exists x_4((x_4 \leftrightarrow (e_0 \vee e_1)) \wedge (x_4 \wedge e_2))) \leftrightarrow \\
\quad\quad\quad\quad (\ (\exists x_3((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))) \leftrightarrow \\
\quad\quad\quad\quad\quad (\ (\exists x_2((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))) \leftrightarrow \\
\quad\quad\quad\quad\quad\quad (\exists x_1((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2))) \ ) \ ) \ )...))
\end{array}
$$

Pour tout n, la QBF th_n n'est pas valide. Si l'on applique la transformation traditionnelle, l'alternance de quantificateurs et de bi-implications mène à une croissance exponentielle de la taille de la QBF th_n sous FNC vis-à-vis de n ce qui n'est pas le cas avec notre stratégie.

L'exemple suivant, pour $n = 7$, une fois mis sous forme prénexe de façon classique, ne peut être résolu par les différents solveurs de notre étude expérimentale en moins d'une heure :

$$
\begin{array}{l}
th_7 = \exists e_0 e_1 e_2 e_3 e_4 e_5 \forall u_0 u_1 u_2 \\
((\exists x_7 \ ((x_7 \leftrightarrow (e_3 \vee e_4)) \wedge (x_6 \wedge e_5))) \leftrightarrow \\
((\exists x_6 \ ((x_6 \leftrightarrow (e_2 \vee e_3)) \wedge (x_6 \wedge e_4))) \leftrightarrow \\
((\exists x_5 \ ((x_5 \leftrightarrow (e_1 \vee e_2)) \wedge (x_5 \wedge e_3))) \leftrightarrow \\
((\exists x_4 \ ((x_4 \leftrightarrow (e_0 \vee e_1)) \wedge (x_4 \wedge e_2))) \leftrightarrow \\
((\exists x_3 \ ((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))) \leftrightarrow \\
((\exists x_2 \ ((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))) \leftrightarrow \\
\quad (\exists x_1 \ ((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2)))))))))
\end{array}
$$

La version FNC de cette QBF contient 1148 symboles propositionnels différents (98 sont universellement quantifiés) et 3038 clauses tandis que la QBF transformée par l'algorithme $rec\_def\_extract$ puis convertie sous

FNC a seulement 33 symboles propositionnels (3 universellement quantifiés) et 82 clauses ; grâce à notre stratégie, la QBF th_{6000} transformée a été décidée en moins d'une heure (voir la section 4).

3.5 Vérification formelle : l'additionneur n-bits

Lors de la conception de circuits logiques, le but est de construire un circuit qui corresponde au comportement souhaité. C'est-à-dire que pour chaque jeu d'entrées possible, la sortie attendue est obtenue. Le plus souvent le modèle booléen du circuit diffère grandement du circuit conçu par l'ingénieur pour des raisons pratiques (encombrement, prix, intégration, etc...). La vérification formelle de circuit est un des problèmes qui s'expriment à l'aide de formules de la logique monadique. La construction de modèles bornés (Bounded Model Construction), est une méthode pour résoudre des problèmes de validité de formules de la logique monadique en les transformant en QBF et en cherchant la validité des formules obtenues. Parmi les circuits, la vérification de l'additionneur n-bits est devenu un problème classique dans sa version QBF et sa description complète peut être trouvée dans [3, 2]. La vérification de l'additionneur consiste à vérifier l'équivalence en logique monadique entre la structure du circuit et le comportement souhaité. Dans ce qui suit, la QBF add_{impl} correspond à l'implantation du circuit, sa structure, et add_{spec} à la spécification, son comportement. (n est la taille de l'additionneur, A et B sont les deux n-bits à additionner, S le résultat de l'addition, c_i la retenue en entrée et c_o la retenue en sortie.) Ainsi, la validité de la QBF ci-dessous add_n assure la correction de l'additionneur n-bits physique.

$$\begin{aligned} add_n = {} & \forall n \forall A \forall B \forall S \forall c_i \forall c_o \\ & (add_{impl}(n, A, B, S, c_i, c_o) \leftrightarrow add_{spec}(n, A, B, S, c_i, c_o)) \end{aligned}$$

Nous donnons en exemple la QBF

$$add_1 = (add_{impl}(1, A, B, S, c_i, c_o) \leftrightarrow add_{spec}(1, A, B, S, c_i, c_o))$$

avec

$$\begin{aligned} & add_{impl}(1, A, B, S, c_i, c_o) \\ & \quad = \quad \exists x_1 \exists x_2 (((D_1(x_1) \wedge D_2(x_2)) \wedge (\exists x_3 \exists x_4 \exists x_5 (D_3(x_3) \wedge \\ & \qquad (C_1(x_3, x_1) \wedge (D_4(x_4) \wedge (D_5(x_5, x_1, x_3) \wedge C_2(x_2, x_5, x_4))))))))) \\ & add_{spec}(1, A, B, S, c_i, c_o) \\ & \quad = \quad \exists x_6 \exists x_7 ((D_1(x_6) \wedge D_2(x_7)) \wedge (C_3(x_6, x_7) \wedge C_4(x_6))) \end{aligned}$$

qui encode la vérification de l'additionneur pour $n = 1$ avec les fonctions

booléennes suivantes :

$$
\begin{array}{l}
D_1(x) = (c_i \leftrightarrow x),\\
D_2(x) = (c_o \leftrightarrow x),\\
D_3(x) = (x \leftrightarrow (A_0 \oplus B_0)),\\
D_4(x) = (x \leftrightarrow (A_0 \wedge B_0)),\\
D_5(x,y,z) = (x \leftrightarrow (y \wedge z)),\\
C_1(x,y) = (S_0 \leftrightarrow (x \oplus y)),\\
C_2(x,y,z) = (x \leftrightarrow (y \vee z)),\\
C_3(x,y) = ((((A_0 \wedge B_0) \vee (A_0 \wedge x)) \vee (B_0 \wedge x)) \leftrightarrow y),\\
C_4(x) = (((A_0 \leftrightarrow B_0) \leftrightarrow S_0) \leftrightarrow x).
\end{array}
$$

Les fonctions C_k, $1 \leq k \leq 4$, représentent le cœur de l'implantation et de la spécification. Les formules add_{impl} et add_{spec} contiennent des instances des fonctions D_k, $1 \leq k \leq 5$, qui sont des définitions à extraire. Les formules add_{impl} et add_{spec} contiennent des symboles propositionnels existentiellement quantifiés, qui représentent des résultats intermédiaires, entre autres les retenues.

La QBF ci-dessous add'_1 est la QBF initiale add_1 mise sous forme prénexe selon l'algorithme classique (les symboles propositionnels y_k sont issus des symboles propositionnels x_k par recopie des sous-formules de la bi-implication). Les quantificateurs universels sont extraits en premier des deux parties de la bi-implication dans le but de minimiser l'alternance de quantificateurs.

$$
\begin{array}{l}
add'_1 = \forall A_0 \forall B_0 \forall S_0 \forall c_i \forall c_o\\
\forall y_1 \forall y_2 \forall y_3 \forall y_4 \forall y_5 \forall y_6 \forall y_7 \exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \exists x_7\\
[[(D_1(y_1) \wedge D_2(y_2) \wedge D_3(y_3) \wedge C_1(y_3,y_1) \wedge D_4(y_4) \wedge D_5(y_5,y_1,y_3)\\
\wedge C_2(y_2,y_5,y_4)) \rightarrow (D_1(x_6) \wedge D_2(x_7) \wedge C_3(x_6,x_7) \wedge C_4(x_6))] \wedge\\
[(D_1(y_6) \wedge D_2(y_7) \wedge C_3(y_6,y_7) \wedge C_4(y_6)) \rightarrow (D_1(x_1) \wedge D_2(x_2)\\
\wedge D_3(x_3) \wedge C_1(x_3,x_1) \wedge D_4(x_4) \wedge D_5(x_5,x_1,x_3) \wedge C_2(x_2,x_5,x_4))]]
\end{array}
$$

L'ordre des symboles propositionnels influe grandement sur l'efficacité des méthodes de résolution [18] ; dans la mise sous forme prénexe de add_1 en add'_1, l'ordre des symboles propositionnels n'est contraint que par le respect de l'ordre partiel qui nécessite que les quantificateurs universels des symboles propositionnels initiaux du problème doivent précéder les quantificateurs existentiels.

La QBF ci-dessous add^t_1 est la QBF initiale add_1 transformée à l'aide de nos équivalences logiques puis mise sous forme prénexe selon l'algorithme classique.

$$
\begin{array}{l}
add^t_1 = \forall A_0 \forall B_0 \forall S_0 \forall c_i \forall c_o \exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \exists x_7 [D_1(x_1) \wedge\\
\quad D_2(x_2) \wedge D_3(x_3) \wedge D_4(x_4) \wedge D_5(x_5,x_1,x_3) \wedge D_1(x_6) \wedge D_2(x_7)] \wedge\\
\quad [[C_1(x_3,x_1) \wedge C_2(x_2,x_5,x_4)] \leftrightarrow [C_3(x_6,x_7) \wedge C_4(x_6)]]
\end{array}
$$

La matrice de la QBF obtenue est partagée en deux parties : la définition des symboles propositionnels intermédiaires suivie de l'exploitation de ces symboles propositionnels intermédiaires dans le cœur de la bi-implication.

4 RÉSULTATS EXPÉRIMENTAUX

Dans le but d'évaluer l'impact de notre stratégie, nous avons développé un programme Prolog [1] pour générer des instances de nos chaînes de bi-implications avec résultats intermédiaires th_n comme décrit en section 3.4 et des instances de l'additionneur n-bits add_n comme décrit dans la section 3.5. La mise sous forme prénexe appliquée ne recherche pas l'ordre optimal des quantificateurs. L'ordre des définitions dans la matrice n'a pas non plus été optimisé. La mise sous forme normale conjonctive est calculée grâce à l'introduction de symboles propositionnels existentiellement quantifiés. Cette mise sous forme normale fait croître polynomialement la taille de la QBF.

Les tests ont été réalisés sur un *Intel(R) Pentium(R) 4 (2.80 GHz)* avec $600Mo$ de mémoire. Pour nos expérimentations, nous avons utilisé les solveurs QBF : sKizzo v0.8.2-beta [6] qui est basé sur la skolemization symbolique [4], Quantor 3.0 [10] qui combine la Q-résolution [12] et l'expansion, QuBE-BJ1.2 [22] qui étend aux QBF la procédure Davis-Logemann-Loveland [13] et une version plus récente, QuBE6.3, qui intègre pour pré-processeur la Q-résolution [11]. Les expériences ont été menées d'abord avec les options par défaut des différents solveurs (en particulier, la Q-résolution comme pré-processeur de sKizzo).

4.1 Chaînes de bi-implications

Aucun des solveurs QBF testés n'a été en mesure de décider si th_n est valide ou non pour $6 < n$ sans appliquer notre stratégie (QuBE6.3 a décidé en 1 seconde pour $n = 5$ mais le temps de calcul a excédé les 3600 secondes pour $n = 6$, Quantor a décidé en 0.2 seconde pour $n = 5$ mais a excédé l'espace mémoire disponible pour $n = 6$, sKizzo a décidé en 10.5 secondes pour $n = 6$ mais le temps de calcul a excédé les 3600 secondes pour $n = 7$). En fait, la taille de la QBF mise sous FNC croît exponentiellement avec n. La figure 2 rapporte les résultats pour les solveurs QBF testés sur les QBF issues de l'algorithme $rec\_def\_extract$ et mises sous FNC. Les deux axes de la figure sont en échelle logarithmique.

4.2 Additionneur n-bits

Les résultats pour l'additionneur n-bits sont rapportés dans le tableau 1 : les temps de calcul sont en secondes, T signifie que le temps de calcul a excédé les 3600 secondes, M signifie que le calcul a dépassé l'espace mémoire disponible. Les résultats de la colonne «sans» sont obtenus pour les QBF originales mises sous FNC et ceux de la colonne «avec» sont ceux obtenus pour les QBF calculées par notre stratégie puis mises sous FNC.

[1] L'ensemble des jeux d'essais et les programmes développés sont disponible sur `http://www.info.univ-angers.fr/pub/damota/qbf`.

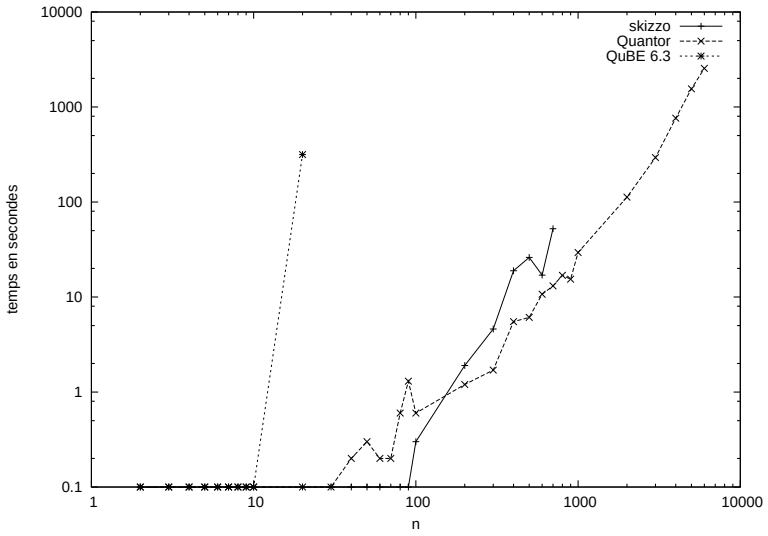

FIG. 2 – Temps de calcul en secondes pour des chaînes de bi-implications avec résultats intermédiaires traduites selon notre stratégie puis mises sous FNC.

Les résultats du tableau 1 montrent que notre stratégie a permis d'obtenir quasiment toujours le meilleur résultat. Pour tous les solveurs à part sKizzo, les résultats sont nets. Pour ce dernier, avec $n < 12$, l'amélioration est significative mais ces performances décroissent ensuite pour l'additionneur n-bits pour des tailles supérieures. Le résultat de notre stratégie pénalise les grandes instances : il peut-être observé grâce à une trace que sKizzo élimine, par l'utilisation de la Q-résolution comme pré-processeur, des résultats intermédiaires (annulant ainsi en partie le travail accompli par notre algorithme). Puisque QuBE6.3, qui intègre la Q-résolution aussi comme pré-processeur, obtient des résultats pires que ceux de QuBE-BJ1.2 qui est sensé être une version moins performante, nous pensons que la Q-résolution a un grand impact sur ce type de problème. Dans le tableau 1 sont aussi rapportés des tests pour sKizzo mais sans la Q-résolution pour lesquels nous obtenons les meilleurs résultats de toute notre étude. L'amélioration obtenue par l'application de notre stratégie est donc très significative.

add	sKizzo avec Q-résolution		sKizzo sans Q-résolution		QuBE 6.3	QuBE BJ1.2	Quantor	
n	sans	avec	sans	avec	avec	avec	sans	avec
4	0,3	**0,1**	0,5	**0,1**	2,98	0,26	8,53	0,14
5	0,5	**0,1**	0,8	**0,1**	177,48	2,16	70,16	0,50
6	1,0	**0,1**	1,3	**0,1**	T	17,90	M	3,04
7	2,5	0,2	4,1	**0,1**	T	155,93	M	15,02
8	5,5	0,4	13,3	**0,2**	T	1199.67	M	53,43
9	9,4	1,1	5,1	**0,3**	T	T	M	M
10	23,7	6,0	11,5	**0,4**	T	T	M	M
11	125,5	31,1	24,4	**0,4**	T	T	M	M
12	170,9	143,7	18,0	**0,5**	T	T	M	M
13	247,6	756,1	11,9	**0,6**	T	T	M	M
14	915,6	T	19,7	**0,8**	T	T	M	M
15	3501,4	T	15,6	**0,9**	T	T	M	M
16	T	T	24,4	**0,9**	T	T	M	M
17	T	T	78,0	**1,4**	T	T	M	M
18	T	T	68,4	**1,6**	T	T	M	M
19	T	T	59,4	**1,1**	T	T	M	M
20	T	T	130,9	**1,3**	T	T	M	M
21	T	T	78,1	**2,4**	T	T	M	M
22	T	T	101,4	**2,2**	T	T	M	M
23	T	T	119,9	**1,7**	T	T	M	M

TAB. 1 – Temps de calcul en secondes pour l'additionneur n-bits «avec» ou «sans» notre stratégie puis mis sous FNC.

5 CONCLUSION

Dans cet article, nous avons proposé différentes équivalences qui permettent de traiter les résultats intermédiaires contenus dans des bi-implications ou des ou-exclusifs pour des QBF non prénexes. Ces équivalences permettent de conserver la taille de la formule et le nombre de symboles propositionnels, tout en sortant les quantificateurs afin de pouvoir réaliser une mise sous forme prénexe. Il est possible de choisir le type de quantification du symbole propositionnel intermédiaire et ainsi réduire le nombre d'alternances de quantificateurs. Les QBF peuvent être vues comme un jeu à deux joueurs : le joueur existentiel essaye de rendre la formule valide alors que le joueur universel essaye de l'invalider. Un résultat intermédiaire est une conséquence des choix précédents, il n'y a aucun choix à faire, peu importe quel joueur joue ce coup. Notre proposition va dans ce sens, en permettant de faire jouer un seul des deux joueurs au choix lors d'un résultat intermédiaire dans une bi-implication, sans augmenter ni la taille de la formule ni le nombre de symboles propositionnels.

Pour évaluer l'impact pratique de notre méthodologie, nous avons choisi de réaliser deux types de tests. Tout d'abord nous avons défini un exemple artificiel pour illustrer que dans certains cas la mise sous forme prénexe classique conduit à une croissance exponentielle de la taille de la formule prénexe. L'étude expérimentale montre que notre nouvelle stratégie de mise sous forme prénexe évite cette croissance exponentielle. La conséquence directe de notre contribution est de rendre possible la décision par les solveurs de

l'état de l'art pour de grandes instances de ces QBF alors que cela était impossible même pour de toutes petites. Ensuite, nous avons choisi un exemple pratique et réaliste : la vérification formelle de circuits logiques, où une spécification doit être équivalente à l'implantation du circuit. Les résultats montrent l'impact positif de notre nouvelle stratégie de mise sous forme prénexe. Nos tests montrent que pour le problème choisi, la Q-résolution est néfaste au temps de résolution. Il serait intéressant de voir, si celle-ci pourrait être de nouveau performante si elle ne s'appliquait pas sur les résultats intermédiaires.

L'étape suivante de ce travail consiste à étendre la stratégie de mise sous forme prénexe pour non seulement extraire les résultats intermédiaires mais l'ensemble des bi-implications et des ou-exclusifs. Un autre travail possible est de comparer les différentes stratégies de mise sous forme prénexe possibles car il apparaît que la manière dont sont extraites les définitions peut influencer grandement les temps de calcul. De plus, ayant constaté que les solveurs QBF intégraient de plus en plus une représentation structurée du lieur [5] et une forme clausale non prénexe des QBF, il serait intéressant de travailler sur l'encodage (non prénexe, non FNC) du problème original et rechercher une QBF non prénexe FNC dont les symboles propositionnels seraient triés selon qu'ils sont issus du problème initial, des résultats intermédiaires ou des symboles liés à l'encodage en FNC. Nous serions alors à même d'exploiter plus d'information en vue d'évaluer l'impact des heuristiques, comme la Q-résolution, sur ces différents types de symboles propositionnels.

RÉFÉRENCES

[1] C. Ansotegui, C. Gomes et B. Selman. Achilles' heel of QBF. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 275–281, 2005.

[2] A. Ayari et D. Basin. Qubos : Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers. In *Proocedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD'02)*, volume 2517 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2002.

[3] A. Ayari, D. Basin et S. Friedrich. Structural and behavioral modeling with monadic logics. In Rolf Drechsler et Bernd Becker, éditeurs, *Proocedings of the 29th IEEE International Symposium on Multiple-Valued Logic*, pages 142–151, Freiburg, Germany, 1999. IEEE Computer Society.

[4] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05)*, volume 3452 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2005.

[5] M. Benedetti. Quantifier trees for QBFs. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, volume 3569 of *Lecture Notes in Computer Science*, pages 378–385. Springer, 2005.

[6] M. Benedetti. skizzo : a suite to evaluate and certify QBFs. In *Proceedings of the 20th International Conference on Automated Deduction (CADE'05)*, pages 369–376, 2005.

[7] M. Benedetti et H. Mangassarian. Experience and perspectives in qbf-based formal verification. *Journal on Satisfiability, Boolean Modeling and Computation*, 2008 (to appear).

[8] P. Besnard, A. Hunter et S. Woltran. Encoding Deductive Argumentation in Quantified Boolean Formulae. *Journal of Artificial Intelligence*, 173 :1406–1423, 2009.

[9] P. Besnard, T. Schaub, H. Tompits et S. Woltran. Paraconsistent reasoning via quantified Boolean formulas, i : Axiomatising signed systems. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 320–331, 2002.

[10] A. Biere. Resolve and Expand. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, 2004.

[11] U. Bubeck et H. K. Büning. Bounded Universal Expansion for Preprocessing QBF. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing (SAT'07)*, volume 4501 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2007.

[12] H. K. Büning, M. Karpinski et A. Flögel. Resolution for quantified boolean formulas. *Information and Computation*, 117(1) :12–18, 1995.

[13] M. Davis, G. Logemann et D. Loveland. A machine program for theorem-proving. *Communication of the ACM*, 5, 1962.

[14] T. Boy de la Tour. An Optimality Result for Clause Form Translation. *Journal of Symbolic Computation*, 14(4) :283–302, 1992.

[15] U. Egly. On Different Structure-Preserving Translations to Normal Form. *Journal of Symbolic Computation*, 22(2) :121–142, 1996.

[16] U. Egly, T. Eiter, V. Klotz, H. Tompits et S. Woltran. Computing Stable Models with Quantified Boolean Formulas : Some Experimental Results. In *Proceedings of the 2001 AAAI Spring Symposium on Answer Set Programming*, pages 53–59, 2001.

[17] U. Egly, T. Eiter, H. Tompits et S. Woltran. Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00) and 12th Conference on on Innovative Applications of Artificial Intelligence (IAAI'00)*, pages 417–422, 2000.

[18] U. Egly, M. Seidl, H. Tompits, S. Woltran et M. Zolda. Comparing Different Prenexing Strategies for Quantified Boolean Formulas. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, volume 2919 of *Lecture Notes in Computer Science*, pages 214–228. Springer, 2003.

[19] J.H. Gallier. *Logic for computer science : foundations of automatic theorem proving*. Harper & Row Publishers, Inc., 1985.

[20] L. Garcia et R. Sabbadin. Complexity results and algorithms for possibilistic influence diagrams. *Artificial Intelligence*, 172(8-9) :1018–1044, 2008.

[21] I. Gent et G.D. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. In *Proceedings of the 2nd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, pages 78–93, 2003.

[22] E. Giunchiglia, M. Narizzano et A. Tacchella. Qube++ : an efficient QBF solver. In *Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design (FMCAD'04)*, volume 3312 of *Lecture Notes in Computer Science*. Springer, 2004.

[23] D. A. Plaisted et S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3) :293–304, 1986.

[24] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10 :323–352, 1999.

[25] A. Sabharwal, C. Ansótegui, C.P. Gomes, J.W. Hart et B. Selman. QBF Modeling : Exploiting Player Symmetry for Simplicity and Efficiency. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 382–395, 2006.

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

[27] G. S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, éditeur, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970.