Equivalences et forme prénexe pour les formules booléennes quantifiées

Benoit da Mota

Équivalences et forme prénexe pour les formules booléennes quantifiées

Benoit Da Mota

LERIA, Université d'Angers
2, boulevard Lavoisier, 49045, Angers Cedex 01
damota@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 conjonctive. Or, il est rarement naturel d'exprimer un problème directement sous cette forme. Dans ce travail, nous exhibons un motif possédant d'intéressantes propriétés, particulièrement lors de la mise sous forme prénexe. Les résultats expérimentaux montrent qu'utiliser nos équivalences logiques améliore le temps de résolution par les différentes procédures.

Mots-clés : Formules booléennes quantifiées, forme prénexe, fusion de quantificateurs, vérification formelle.

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. Une multitude d'importants problèmes de décision parmi des champs très divers ont des transformations polynomiales vers le problème de validité des QBF. La plupart des procédures actuelles pour décider des QBF nécessitent d'avoir en entrée une formule sous forme normale conjonctive Giunchiglia *et al.* (2004); Biere (2004); Benedetti (2005b). Or, les problèmes sont rarement exprimés directement sous cette forme. Il est plus naturel de les représenter en utilisant la richesse du langage et donc à l'aide d'opérateurs plus expressifs (l'implication, l'équivalence et le ou exclusif) et avec des quantificateurs à l'intérieur des formules. La mise sous forme normale conjonctive nécessite que la formule soit sous forme prénexe, c'est-à-dire avec tous les quantificateurs devant la formule.

La mise sous forme normale conjonctive a été largement étudiée Plaisted & Greenbaum (1986); de la Tour (1992); Egly (1996), car cette forme normale est utilisée pour la satisfiabilité des formules booléennes (non quantifiées). 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 Egly *et al.* (2003);

Giunchiglia *et al.* (2006). Mais aucune règle n'est fournie pour extraire les quantificateurs de formules booléennes quantifiées contenant des équivalences et des ou exclusifs. Le seul choix possible est d'exprimer ces opérateurs en fonction des opérateurs pour lesquels nous connaissons des règles. Nous verrons que ce choix a des conséquences sur la taille de la formule et sur le nombre de variables. Nous nous intéressons dans la mise sous forme prénexe à l'étape qui consiste à ré-écrire une formule en une forme logiquement équivalente sans équivalence ni ou exclusif. Nous proposons dans un premier temps des équivalences permettant de traiter un motif particulier : les variables existentielles qui représentent un résultat intermédiaire. Dans un second temps nous exploitons les propriétés de ce motif afin de proposer une solution au cas général. Pour la partie expérimentale, nous introduisons un problème théorique ad hoc afin de valider notre approche. Puis nous utilisons des QBF codant la vérification formelle de circuits logiques et plus particulièrement l'additionneur n-bits, qui illustre en pratique l'intérêt de notre proposition. Les preuves des théorèmes introduits sont disponibles à l'adresse : `http://forge.info.univ-angers.fr/~damota/rjcia09/preuves.pdf`.

2 Préliminaires

L'ensemble des valeurs booléennes **v** et **f** est noté $BOOL$. L'ensemble des variables (propositionnelles ou booléennes) est noté $\mathcal{V}$. 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 l'équivalence et $\oplus$ pour le ou exclusif. L'ensemble des opérateurs $\{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ est noté $\mathcal{O}$. Le symbole $\exists$ est utilisé pour la quantification existentielle et $\forall$ pour la quantification universelle (q est utilisé pour noter un quantificateur quelconque). Un littéral est une variable ou la négation de celle-ci. L'ensemble QBF des formules booléennes quantifiées est défini inductivement ainsi : tout symbole (constante ou variable) propositionnel est élément 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 ; si F est un élément de QBF et x est une variable alors $(\exists x\ F)$ et $(\forall x\ F)$ sont des éléments de QBF. Par convention, des quantificateurs différents lient des variables différentes. L'ensemble des variables d'une formule F est noté $\mathcal{V}(F)$. Une variable x est libre si et seulement si elle n'apparait pas sous la portée d'un quantificateur $\exists x$ ou $\forall x$. L'ensemble des variables libres d'une formule F est noté $\mathcal{L}(F)$. Une substitution est une fonction de l'ensemble des variables dans l'ensemble des formules (quantifiées ou non). Nous définissons la substitution de x par F dans G, notée $G[x \leftarrow F]$, comme étant la formule obtenue de G en remplaçant toutes les occurrences de la variable x par la formule F. Un lieur est une chaîne de caractères $q_1 x_1 \ldots q_n x_n$ avec $x_1, \ldots, x_n$ des variables distinctes et $q_1 \ldots q_n$ des quantificateurs. Une QBF QF est en forme prénexe si F est une formule booléenne, appelée matrice et Q est un lieur. Elle est sous forme normale conjonctive (FNC) si F est une formule booléenne en forme normale conjonctive (i.e. une conjonction de disjonctions de littéraux).

La sémantique des symboles booléens est définie de manière habituelle. Une valuation est une fonction de l'ensemble des variables dans $BOOL$; elle satisfait une formule si appliquée à celle-ci elle vaut **v** sinon elle la falsifie. La satisfaction propositionnelle

est notée $\models$ et l'équivalence logique $\equiv$. La sémantique des quantificateurs est la suivante : pour toute variable x et QBF F,

$$(\exists x\ F) \equiv (F[x \leftarrow \top] \vee F[x \leftarrow \bot]) \quad \text{et} \quad (\forall x\ F) \equiv (F[x \leftarrow \top] \wedge F[x \leftarrow \bot])$$

Une QBF F est valide si $F \equiv \top$.

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 (QBF) consistant à décider si une formule booléenne quantifiée est valide ou non est le problème canonique de la classe PSPACE-complet.

3 Mise sous forme prénexe et équivalences

3.1 Motivations

Dans Egly *et al.* (2003), 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 l'équivalence ou du ou exclusif. D'une manière générale, la mise sous forme prénexe se décompose en trois étapes :

(i) La suppression des équivalences et des ou exclusifs dont au moins un quantificateur doit être extrait, grâce aux équivalences logiques suivantes :
1) $(F \leftrightarrow G) \equiv ((G \rightarrow F) \wedge (F \rightarrow G))$ 2) $(F \oplus G) \equiv ((G \vee F) \wedge (\neg F \vee \neg G))$
Nous remarquons que les formules F et G sont recopiées deux fois.

(ii) Le renommage des variables afin que des quantificateurs différents lient des variables différentes.

(iii) L'extraction des quantificateurs à l'aide des équivalences logiques utilisées dans Egly *et al.* (2003) que nous complétons avec celles pour l'implication (règles 13 à 16) qui sont facilement déduites des précédentes. Soit F, G et H des QBF et x une variable booléenne ($x \notin \mathcal{L}(H)$), alors :

3) $\exists x(\neg F) \equiv \neg(\forall x\ F)$.
4) $\forall x(\neg F) \equiv \neg(\exists x\ F)$.
5) $\forall x\ F \equiv F$, si $x \notin \mathcal{L}(F)$.
6) $\exists x\ F \equiv F$, si $x \notin \mathcal{L}(F)$.
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)[1] $\forall x(F \wedge G) \equiv (\forall x\ F) \wedge (\forall x\ G)$.
12)[2] $\exists x(F \vee G) \equiv (\exists x\ F) \vee (\exists x\ G)$.
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) $\exists x(H \rightarrow G) \equiv H \rightarrow (\exists x\ G)$,

L'extraction des quantificateurs est un processus qui consiste à appliquer ces équivalences de la droite vers la gauche jusqu'à obtenir 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. Alors que l'équivalence représente le "si et seulement si" du langage naturel, le ou exclusif représente sa négation. Le pouvoir

[1] Il n'y a pas de règle similaire pour le $\forall$ avec l'opérateur logique $\vee$, en effet $\forall x(x \vee (\neg x)) \equiv \top$ mais $((\forall x(x)) \vee (\forall x(\neg x))) \equiv \bot$.

[2] Il n'y a pas de règle similaire pour le $\exists$ avec l'opérateur logique $\wedge$, en effet $\exists x(x \wedge (\neg x)) \equiv \bot$ mais $((\exists x(x)) \wedge (\exists x(\neg x))) \equiv \top$

d'expression de l'équivalence 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 et 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.

Soit a une variable booléenne, ϕ, ϕ_1 et ϕ_2 des formules booléennes quantifiées telles que $\phi = (\phi_1 \leftrightarrow \exists a(\phi_2))$ et $a \notin \mathcal{L}(\phi_1)$. Dans un premier temps il faut exprimer l'équivalence 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 variables a afin d'extraire les quantificateurs de la conjonction. Soit b une nouvelle variable booléenne et $\phi'_2 = \phi_2[a \leftarrow b]$ 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 : $\phi \overset{7,9}{\equiv} \forall b\ \exists a((\phi_1 \rightarrow \phi_2) \wedge (\phi'_2 \rightarrow \phi_1))$. Dans le cas général, si F est une formule booléenne quantifiée, $\forall y\ \exists x(F) \not\equiv \exists x\ \forall y(F)$. 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 variables a et b représentent la même variable de la formule non prénexe mais rien dans la formule prénexe ne semble les mettre en relation.

Nous envisageons le cas où un quantificateur doit traverser plusieurs équivalences. 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 une variable booléenne, ϕ_d, ϕ_g, ϕ_1, ϕ_2 et ϕ_3 des QBF 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 $a \notin \mathcal{L}(\phi_1)$ et $a \notin \mathcal{L}(\phi_2)$ alors $\phi_d \equiv \phi_g$ par associativité de l'équivalence logique. Nous mettons ϕ_g sous forme prénexe :

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

puis ϕ_d :

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

Les formules ϕ_g et ϕ_d, bien qu'équivalentes, n'ont pas du tout la même taille ni le même nombre de variables une fois mises sous forme prénexe. Maintenant, si la formule considérée est $(\phi_1 \leftrightarrow (\phi_2 \leftrightarrow ...(\phi_n \leftrightarrow (\exists a\ \phi_{n+1}))))$, alors 2^n variables sont créées dont la moitié sera 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 variable croissent exponentiellement

avec le nombre d'équivalences traversées. Quant à la formule $((\exists a\ \phi_{n+1}) \leftrightarrow (\phi_n \leftrightarrow ...(\phi_2 \leftrightarrow \phi_1)))$, après la mise sous forme prénexe, elle possède 2 variables dont une quantifiée universellement et chaque ϕ_k est présente seulement 2 fois. Nous pourrions choisir de placer les parenthèses de cette manière, mais si chacune des ϕ_k possède un quantificateur à extraire, nous n'évitons pas le pire cas. Pour notre partie expérimentale, notre problème théorique est de cette forme. Nous remarquerons que très vite la mise sous forme prénexe classique ne permet pas de résoudre ce cas.

3.2 Mise sous forme prénexe et résultats intermédiaires

Il est habituel en programmation et en spécification QBF d'utiliser une variable existentiellement quantifiée pour représenter un résultat intermédiaire soit pour ne pas répéter une sous formule, soit pour la lisibilité, soit enfin par construction. Par extension d'un résultat propositionnel classique Tseitin (1970), nous nous intéressons au motif $\exists x((x \leftrightarrow F) \wedge G)$, avec $x \notin \mathcal{L}(F)$, qui est équivalent à $G[x \leftarrow F]$. Par la suite nous désignerons une telle variable x, *variable intermédiaire* ou *résultat intermédiaire*. Pour les procédures QBF actuelles, les variables intermédiaires sont traitées comme des variables du problème alors qu'il suffirait pour s'en abstraire de faire un remplacement syntaxique. Nous proposons de garder ces variables intermédiaires car elles peuvent aussi avoir un intérêt opérationnel, tout en évitant les recopies et l'explosion du nombre de variables. Pour ce faire nous exhibons un ensemble d'équivalences logiques permettant d'extraire le quantificateur d'un résultat intermédiaire. L'objectif est de ne créer aucune nouvelle variable et de ne pas recopier les sous formules.

Théorème 1
Soit F et G des QBF et x une variable représentant le résultat intermédiaire F, avec $x \notin \mathcal{L}(F)$, alors les équivalences logiques suivantes sont vraies :
1) $(\exists x((F \leftrightarrow x) \wedge G)) \equiv (\forall x((F \leftrightarrow x) \rightarrow G))$
2) $(\exists x((F \leftrightarrow x) \rightarrow G)) \equiv \top$
3) $(\forall x((F \leftrightarrow x) \wedge G)) \equiv \bot$
et pour une formule booléenne quantifiée $H = (qx((F \leftrightarrow x)\ o\ G))$ avec $q \in \{\exists, \forall\}$ et $o \in \{\wedge, \vee, \rightarrow\}$ il est possible de mettre H sous la forme d'une des trois équivalences.

Notre proposition s'appuie sur le théorème suivant qui permet d'extraire la définition de la variable intermédiaire lorsqu'elle est dans une sous formule d'une équivalence (ou d'un ou exclusif).

Théorème 2
Soit F, G et H des formules booléennes quantifiées et x une variable booléenne qui représente le résultat intermédiaire F, avec $x \notin \mathcal{L}(H)$, $x \notin \mathcal{L}(F)$ et $o \in \{\oplus, \leftrightarrow\}$, alors $(H\ o\ (\exists x((F \leftrightarrow x) \wedge G))) \equiv (\exists x((F \leftrightarrow x) \wedge (H\ o\ G)))$.

L'objectif est atteint : le quantificateur qui porte sur x est extrait de l'équivalence ou du ou exclusif, aucune variable n'est créée, aucune sous formule n'est dupliquée. Il est possible de choisir comment sera quantifié x à l'aide du théorème 1, rien ne nous oblige à garder le même quantificateur. De même, pour extraire un quantificateur universel, il

faut appliquer l'équivalence du théorème 1 sur la partie gauche de l'équivalence. 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 équivalence ou sa négation, sans augmenter ni la taille de la formule ni le nombre de variables.

3.3 Mise sous forme prénexe et formules quelconques

Nous possédons maintenant un ensemble d'équivalences permettant de sortir des résultats intermédiaires et donc un quantificateur. Or les quantificateurs à extraire lors de la mise sous forme prénexe d'une formule quelconque ne portent pas uniquement sur des variables intermédiaires. Pouvons-nous toujours nous ramener à des motifs : $\exists a((a \leftrightarrow F) \wedge G)$, avec a variable absente de F ? Si a est la variable que nous souhaitons extraire, la réponse est non. Par contre, il est possible d'introduire un résultat intermédiaire x afin d'extraire F de G, avec F qui contient le quantificateur à extraire. Cette intuition est formalisée dans le théorème suivant, puis illustrée ensuite.

Théorème 3
Soit F et G des QBF, telles que F est une sous formule de G. G ne possède pas de quantificateur portant sur les variables de F autres que ceux de F. Soit x une variable telle que $x \notin \mathcal{L}(G)$. Nous appelons G', la formule G dans laquelle chaque occurrence de F est remplacée par x, alors : $G \equiv \exists x((x \leftrightarrow F) \wedge G')$

La deuxième hypothèse du théorème peut aussi être formulée ainsi : toute variable libre dans F doit être libre dans G. Nous remarquons le cas particulier où $F = G$, alors le théorème énonce que $G \equiv \exists x((x \leftrightarrow G) \wedge x)$.

Reprenons un des exemples de la section 3.1. Soient a et x des variables booléennes, ϕ_d, ϕ_g, ϕ_1, ϕ_2 et ϕ_3 des QBF 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 $a \notin \mathcal{L}(\phi_1)$ et $a \notin \mathcal{L}(\phi_2)$. Pour appliquer le théorème 3 à la formule ϕ_g, nous prenons $G = \phi_g$ et $F = (\exists a\ \phi_3)$. De même pour la formule ϕ_d, sauf que $G = \phi_d$.

$$
\begin{array}{lll}
\phi_g & \overset{th_3}{\equiv} & \exists x\ [(x \leftrightarrow (\exists a\ \phi_3)) \wedge ((\phi_1 \leftrightarrow \phi_2) \leftrightarrow x)] \\
 & \overset{1,16,13,9,7}{\equiv} & \exists x \exists a \forall b\ [((x \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow x)) \wedge ((\phi_1 \leftrightarrow \phi_2) \leftrightarrow x)] \\
\phi_d & \overset{th_3}{\equiv} & \exists x\ [(x \leftrightarrow (\exists a\ \phi_3)) \wedge (\phi_1 \leftrightarrow (\phi_2 \leftrightarrow x))] \\
 & \overset{1,16,13,9,7}{\equiv} & \exists x \exists a \forall b\ [((x \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow x)) \wedge (\phi_1 \leftrightarrow (\phi_2 \leftrightarrow x))]
\end{array}
$$

Nous remarquons que les formules ϕ_g et ϕ_d une fois mises sous forme prénexe sont très proches et il est trivial d'affirmer qu'elles sont équivalentes par associativité de l'équivalence logique. Puisque nous considérons que la variable a n'est pas un résultat intermédiaire, il faut supprimer l'équivalence dans la formule $(x \leftrightarrow (\exists a\ \phi_3))$ afin de terminer la mise sous forme prénexe. La formule ϕ_3 est donc dupliquée, et une variable

b quantifiée universellement est introduite. Comme coût supplémentaire, nous avons aussi introduit un résultat intermédiaire x. En contrepartie, le nombre de variables introduites par quantificateur extrait et le nombre de recopies, ne dépendent plus du nombre d'équivalences et de ou exclusifs traversés. Pour chaque quantificateur à extraire, nous introduisons 2 nouvelles variables et seule la formule sous la portée de ce quantificateur est recopiée une fois. De plus, la définition du résultat intermédiaire introduit est en conjonction avec le reste de la formule. Elle peut être vue comme une condition nécessaire pour satisfaire le cœur de la formule.

3.4 Exploitation de la fusion de quantificateurs

Dans Egly *et al.* (2003), les auteurs suggèrent d'approfondir leur travail en exploitant la fusion de quantificateurs (règles 11 et 12 de la section 3.1). Notre méthode pour mettre sous forme prénexe construit une chaîne de $\wedge$ dont toutes les sous formules sauf une possèdent un quantificateur universel à extraire. La règle 11 peut s'appliquer facilement afin de réduire le nombre de quantificateurs universels.

Soient a, b, c, a', b', c', x, y et z des variables booléennes, ϕ, ϕ_1, ϕ_2 et ϕ_3 des QBF telles que $\phi = (((\forall a\ \phi_1) \leftrightarrow (\exists b\ \phi_2)) \leftrightarrow (\exists c\ \phi_3))$, avec $a \notin \mathcal{L}(\phi_2)$, $a \notin \mathcal{L}(\phi_3)$, $b \notin \mathcal{L}(\phi_1)$, $b \notin \mathcal{L}(\phi_3)$, $c \notin \mathcal{L}(\phi_1)$ et $c \notin \mathcal{L}(\phi_2)$. La présence de $\forall a$ qui porte sur ϕ_1 permet d'illustrer que la fusion de quantificateurs n'est pas limitée par le type des quantificateurs présents dans les formules extraites à l'aide du théorème 3. Nous appliquons ce théorème en priorité.

$$
\begin{array}{lll}
\phi & \overset{th_3}{\equiv} & \exists x \exists y \exists z\ [(x \leftrightarrow (\forall a\ \phi_1)) \wedge (y \leftrightarrow (\exists b\ \phi_2)) \wedge (z \leftrightarrow (\exists c\ \phi_3)) \wedge \\
 & & ((x \leftrightarrow y) \leftrightarrow z)] \\
 & \overset{1,16,15,14,13,9,7}{\equiv} & \exists x \exists y \exists z \exists a' \exists b \exists c[\\
 & & (\forall a((x \rightarrow \phi_1) \wedge (\phi_1[a \leftarrow a'] \rightarrow x))) \wedge \\
 & & (\forall b'((y \rightarrow \phi_2) \wedge (\phi_2[b \leftarrow b'] \rightarrow y))) \wedge \\
 & & (\forall c'((z \rightarrow \phi_3) \wedge (\phi_3[c \leftarrow c'] \rightarrow z))) \wedge \\
 & & ((x \leftrightarrow y) \leftrightarrow z)] \\
 & \overset{11}{\equiv} & \exists x \exists y \exists z \exists a' \exists b \exists c \forall u[\\
 & & ((x \rightarrow \phi_1[a \leftarrow u]) \wedge (\phi_1[a \leftarrow a'] \rightarrow x)) \wedge \\
 & & ((y \rightarrow \phi_2) \wedge (\phi_2[b \leftarrow b'][b' \leftarrow u] \rightarrow y)) \wedge \\
 & & ((z \rightarrow \phi_3) \wedge (\phi_3[c \leftarrow c'][c' \leftarrow u] \rightarrow z)) \wedge \\
 & & ((x \leftrightarrow y) \leftrightarrow z)]
\end{array}
$$

La première étape consiste à extraire tous les maillons de la chaîne d'équivalences contenant des quantificateurs à l'aide du théorème 3. Puis nous effectuons une mise sous forme prénexe classique sauf que la règle 11 est appliquée en priorité. Sans l'application de cette règle nous aurions obtenu :

$$
\begin{array}{lll}
\phi & \equiv & \exists x \exists y \exists z \exists a' \exists b \exists c \forall a \forall b' \forall c'[((x \rightarrow \phi_1) \wedge (\phi_1[a \leftarrow a'] \rightarrow x)) \wedge ((y \rightarrow \phi_2) \\
 & & \wedge (\phi_2[b \leftarrow b'] \rightarrow y)) \wedge ((z \rightarrow \phi_3) \wedge (\phi_3[c \leftarrow c'] \rightarrow z)) \wedge ((x \leftrightarrow y) \leftrightarrow z)]
\end{array}
$$

3.5 Chaînes d'équivalences contenant un résultat intermédiaire

Afin d'évaluer en pratique l'impact de nos propositions sur le cas $(\phi_1 \leftrightarrow (\phi_2 \leftrightarrow ...(\phi_{n-1} \leftrightarrow (\phi_n))))$ évoqué à la fin la section 3.1, nous définissons une classe de formules ψ_n, où chaque ϕ_k contient un résultat intermédiaire. Les ϕ_k sont 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 variables booléennes présentes dans d'autres maillons de la chaîne d'équivalences. La formule ψ_n, pour $n \geq 2$ et $m = n - 2$ est de la forme :

$\psi_n = \exists e_0, ..., e_m \forall u_0, u_1, u_2$
$((\exists x_n((x_n \leftrightarrow (e_{m-2} \vee e_{m-1})) \wedge (x_n \wedge e_m))) \leftrightarrow$
$((\exists x_{n-1}((x_{n-1} \leftrightarrow (e_{m-3} \vee e_{m-2})) \wedge (x_{n-1} \wedge e_{m-1}))) \leftrightarrow$
...
$((\exists x_3((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))) \leftrightarrow$
$[(\exists x_2((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))) \leftrightarrow$
$(\exists x_1((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2)))])...))$

L'exemple suivant, pour $n = 7$, pourtant de taille modeste, une fois mis sous forme prénexe de façon classique, ne peut être résolu par les différentes procédures :
$\psi_7 = \exists e_0, e_1, e_2, e_3, e_4, e_5, \forall u_0, u_1, u_2((\exists x_7((x_7 \leftrightarrow (e_3 \vee e_4)) \wedge (x_6 \wedge e_5))) \leftrightarrow$
$((\exists x_6((x_6 \leftrightarrow (e_2 \vee e_3)) \wedge (x_6 \wedge e_4))) \leftrightarrow ((\exists x_5((x_5 \leftrightarrow (e_1 \vee e_2)) \wedge (x_5 \wedge e_3))) \leftrightarrow$
$((\exists x_4((x_4 \leftrightarrow (e_0 \vee e_1)) \wedge (x_4 \wedge e_2))) \leftrightarrow ((\exists x_3((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))) \leftrightarrow$
$((\exists x_2((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))) \leftrightarrow (\exists x_1((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2)))))))))$

3.6 Vérification formelle de circuits : 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 Ayari *et al.* (1999); Ayari & Basin (2002). La vérification de l'additionneur consiste à vérifier l'équivalence en logique monadique entre la structure du circuit et le comportement souhaité (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) :

$\phi = \forall n \forall A \forall B \forall S \forall c_i \forall c_o(add_{struc}(n, A, B, S, c_i, c_o) \leftrightarrow add_{comp}(n, A, B, S, c_i, c_o))$

Les formules add_{struc} et add_{comp} contiennent des variables existentielles, qui représentent des résultats intermédiaires, entre autres les retenues. Grâce aux théorèmes 1 et 2, nous avons le choix d'extraire les variables intermédiaires soit en existentielles, soit en universelles. Dans les deux cas nous pouvons obtenir (en sortant d'abord les quantificateurs universels puis existentiels) une alternance des quantificateurs de type $\forall\exists$ et conserver cette alternance après la mise sous FNC.

Nous donnons un exemple pour l'additionneur n-bits avec $n = 1$. La QBF ϕ_1 code la vérification de l'additionneur pour $n = 1$ avec les fonctions booléennes suivantes : $C_1(x) = (c_i \leftrightarrow x)$, $C_2(x) = (c_o \leftrightarrow x)$, $C_3(x) = (x \leftrightarrow (A_0 \oplus B_0))$, $C_4(x) = (x \leftrightarrow (A_0 \wedge B_0))$, $C_5(x, y, z) = (x \leftrightarrow (y \wedge z))$, $F_1(x, y) = (S_0 \leftrightarrow (x \oplus y))$, $F_2(x, y, z) = (x \leftrightarrow (y \vee z))$, $F_3(x, y) = ((((A_0 \wedge B_0) \vee (A_0 \wedge x)) \vee (B_0 \wedge x)) \leftrightarrow y)$, et $F_4(x) = (((A_0 \leftrightarrow B_0) \leftrightarrow S_0) \leftrightarrow x)$. Les fonctions C_k, $1 \leq k \leq 5$ représentent les motifs extraits de la QBF initiale ; les fonctions C_3, C_4 et C_5 représentent les définitions des variables intermédiaires x_3, x_4 et x_5 ; les fonctions F_k, $1 \leq k \leq 4$, représentent le cœur de la spécification de l'additionneur n-bits.

$\phi_1 = \forall c_i \forall c_o \forall A_0 \forall B_0 \forall S_0$
$[\exists x_1 \exists x_2 (((C_1(x_1) \wedge C_2(x_2)) \wedge$
$(\exists x_3 \exists x_4 \exists x_5 (C_3(x_3) \wedge (F_1(x_3, x_1) \wedge (C_4(x_4) \wedge (C_5(x_5, x_1, x_3) \wedge F_2(x_2, x_5, x_4)))))))))]$
$\leftrightarrow [\exists x_6 \exists x_7 ((C_1(x_6) \wedge C_2(x_7)) \wedge (F_3(x_6, x_7) \wedge F_4(x_6)))]$

La QBF ci-dessous ϕ_i est la QBF initiale ϕ_1 mise sous forme prénexe selon l'algorithme classique (les variables y_k sont issues des variables x_k par recopie des sous-formules de l'équivalence).

$\phi_i = \forall c_i \forall c_o \forall A_0 \forall B_0 \forall S_0 \forall y_1 \forall y_2 \forall y_3 \forall y_4 \forall y_5 \forall y_6 \forall y_7 \exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \exists x_7$
$[C_1(y_1) \wedge C_2(y_2) \wedge C_3(y_3) \wedge (F_1(y_3, y_1) \wedge C_4(y_4) \wedge C_5(y_5, y_1, y_3) \wedge F_2(y_2, y_5, y_4)]$
$\rightarrow [C_1(x_6) \wedge C_2(x_7) \wedge F_3(x_6, x_7) \wedge F_4(x_6)]] \wedge$
$[C_1(y_6) \wedge C_2(y_7) \wedge F_3(y_6, y_7) \wedge F_4(y_6)]$
$\rightarrow [C_1(x_1) \wedge C_2(x_2) \wedge C_3(x_3) \wedge F_1(x_3, x_1) \wedge C_4(x_4) \wedge C_5(x_5, x_1, x_3) \wedge F_2(x_2, x_5, x_4)]]$

L'ordre des variables influe grandement sur l'efficacité des méthodes de résolution Egly *et al.* (2003) ; dans la mise sous forme prénexe de ϕ_1 en ϕ_i, l'ordre des variables n'est contraint que par le respect de l'ordre partiel qui nécessite que les quantificateurs universels des variables initiales du problème doivent précéder les quantificateurs existentiels.

La QBF ci-dessous ϕ_t est la QBF initiale ϕ_1 transformée à l'aide des équivalences logiques du théorème 2, puis mise sous forme prénexe selon l'algorithme classique.

$\phi_t = \forall c_i \forall c_o \forall A_0 \forall B_0 \forall S_0 \exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \exists x_7$
$[C_1(x_1) \wedge C_2(x_2) \wedge C_3(x_3) \wedge C_4(x_4) \wedge C_5(x_5, x_1, x_3) \wedge C_1(x_6) \wedge C_2(x_7)] \wedge$
$[[F_1(x_3, x_1) \wedge F_2(x_2, x_5, x_4)] \leftrightarrow [F_3(x_6, x_7) \wedge F_4(x_6)]]$

La matrice de la QBF obtenue est partagée en deux parties : la définition des variables intermédiaires suivie de l'exploitation de ces variables dans le cœur de l'équivalence.

4 Résultats expérimentaux

4.1 Chaîne d'équivalences

Nous avons développé un algorithme en Prolog, qui permet de générer les instances des chaînes d'équivalences (cf. section 3.5), puis d'effectuer la mise sous FNC. Notons que les instances sont invalides.

La table 1 montre les préfixes et le nombre de clauses (NC_f et NC_t), pour différentes valeurs de n, de la FNC des chaînes d'équivalences ψ_n avec la méthode de la section 3.3 ainsi que la fusion ($\psi_{n,f}$) et avec l'extraction des résultats intermédiaires ($\psi_{n,t}$). Nous n'avons pas donné les préfixes des chaînes d'équivalences mises sous FNC de façon classique ($\psi_{n,i}$) car ils sont bien trop grands. Pour avoir un ordre d'idée, avec $n = 7$, il y a 1148 variables dont 98 quantifiées universellement et 3038 clauses. Avec $n = 8$, il y

n	préfixe pour $\psi_{n,f}$	NC_f	préfixe pour $\psi_{n,t}$	NC_t
5	∃[4]∀[3]∃[10]∀[1]∃[64]	207	∃[4]∀[3]∃[29]	82
6	∃[5]∀[3]∃[12]∀[1]∃[77]	249	∃[5]∀[3]∃[35]	99
7	∃[6]∀[3]∃[14]∀[1]∃[90]	291	∃[6]∀[3]∃[41]	116
10	∃[9]∀[3]∃[20]∀[1]∃[129]	417	∃[9]∀[3]∃[59]	167
20	∃[19]∀[3]∃[40]∀[1]∃[259]	837	∃[19]∀[3]∃[119]	337
30	∃[29]∀[3]∃[60]∀[1]∃[389]	1257	∃[29]∀[3]∃[179]	507
600	∃[599]∀[3]∃[1200]∀[1]∃[7799]	25197	∃[599]∀[3]∃[3599]	10197
700	∃[699]∀[3]∃[1400]∀[1]∃[9099]	29397	∃[699]∀[3]∃[4199]	11897
800	∃[799]∀[3]∃[1600]∀[1]∃[10399]	33597	∃[799]∀[3]∃[4799]	13597
4000	∃[3999]∀[3]∃[8000]∀[1]∃[51999]	167997	∃[3999]∀[3]∃[23999]	67997
5000	∃[4999]∀[3]∃[10000]∀[1]∃[64999]	209997	∃[4999]∀[3]∃[29999]	84997
6000	∃[5999]∀[3]∃[12000]∀[1]∃[77999]	251997	∃[5999]∀[3]∃[35999]	101997

TAB. 1 – Tailles des QBF transformées pour les chaînes d'équivalences.

a 2301 variables dont 194 universelles et 6110 clauses. Nous avons évalué la taille de la FNC à plus de 10^5 variables pour $n = 20$, plus de 10^{30} pour $n = 100$ et plus de 10^{1800} pour $n = 6000$. La première remarque est donc que nos deux méthodes permettent de réduire exponentiellement la taille de la formule mise sous FNC par rapport à la méthode classique.

Nous allons maintenant mesurer l'impact de nos propositions sur le temps de résolution des problèmes par différentes procédures. Les tests ont été effectués sur un Intel(R) Pentium(R) 4 à 2.80GHz avec 1Go de mémoire vive disponible. Les résultats sont résumés dans la table 2, les temps sont en secondes, un 'T' indique que la procédure n'a pas pu résoudre l'instance en moins de 3600 secondes, un 'M' indique un dépassement de la mémoire disponible et un '-' indique que l'instance n'a pas été testée. Nous utilisons différentes procédures avec leurs réglages par défaut : sKizzo v0.8.2-beta Benedetti (2005b) qui intègre skolémisation symbolique et appel à une procédure SAT, Quantor 3.0 Biere (2004) qui combine Q-résolution Büning *et al.* (1995) et expansion, QuBE-BJ1.2 Giunchiglia *et al.* (2004) qui étend aux QBF la procédure de Davis-Logemann-Loveland Davis *et al.* (1962) et QuBE6.3 une version plus récente qui utilise un préprocesseur Bubeck & Büning (2007) intégrant la Q-résolution. Pour ce problème, QuBE6.3 donne toujours de meilleurs résultats que QuBE-BJ1.2.

Nous avons donné comme exemple dans la section 3.5 la formule ψ_7. Cette formule, certes théorique, est relativement petite dans sa forme non prénexe, et pourtant aucune procédure n'arrive à résoudre le problème $\psi_{7,i}$ (mise sous forme prénexe classique). Bien que ce soit le même problème, les formulations $\psi_{7,f}$ et $\psi_{7,t}$ sont exponentiellement plus succinctes que $\psi_{7,i}$ et pour toutes les procédures la réponse est immédiate, que ce soit pour $\psi_{7,f}$ ou $\psi_{7,t}$. Pour toutes les procédures, les temps de calcul sont très similaires entre $\psi_{n,f}$ et $\psi_{n,t}$. Sur cet exemple, la méthode de la section 3.3 donne d'aussi bons résultats que la méthode traitant particulièrement les résultats intermédiaires.

Par soucis de clarté, nous n'avons pas consigné dans la table 2, les résultats obtenus en utilisant la méthode de la section 3.3 sans la fusion. Les formules obtenues grâce à cette méthode, que nous noterons $\psi_{n,e}$ dans la table 3, demandent une étude supplémentaire. En effet, les 3 procédures utilisent une structure de quantificateurs Giunchiglia *et al.*

ψ	sKizzo			QuBE6.3			Quantor		
n	$\psi_{n,i}$	$\psi_{n,f}$	$\psi_{n,t}$	$\psi_{n,i}$	$\psi_{n,f}$	$\psi_{n,t}$	$\psi_{n,i}$	$\psi_{n,f}$	$\psi_{n,t}$
5	<0,1	<0,1	<0,1	1,0	<0,1	<0,1	0,2	<0,1	<0,1
6	10,5	<0,1	<0,1	T	<0,1	<0,1	M	<0,1	<0,1
7	T	<0,1	<0,1	T	<0,1	<0,1	M	<0,1	<0,1
10	T	<0,1	<0,1	T	<0,1	<0,1	M	<0,1	<0,1
20	-	<0,1	<0,1	-	205,9	315,9	-	<0,1	<0,1
30	-	<0,1	<0,1	-	T	T	-	<0,1	<0,1
600	-	25,4	17,0	-	-	-	-	8,8	10,7
700	-	57,6	52,4	-	-	-	-	13,4	13,1
800	-	M	M	-	-	-	-	16,5	16,9
4000	-	-	-	-	-	-	-	940,0	762,9
5000	-	-	-	-	-	-	-	1900,2	1555,5
6000	-	-	-	-	-	-	-	2974,1	2556,0

TAB. 2 – Résultats pour différentes procédures avec les QBF initiales et les QBF transformées.

(2006); Biere (2004); Benedetti (2005a). Elles construisent un arbre de quantificateurs, où chaque feuille est étiquetée par un ensemble de clauses, afin de réduire la portée de chaque quantificateur. Ce qui revient à appliquer les équivalences de la section 3.1 de la gauche vers la droite, cela peut annuler notre fusion par application de la règle 11. Pour illustrer brièvement l'intérêt de la fusion, nous désactivons la construction de la structure de quantificateurs pour sKizzo. Pour ce problème, l'utilisation de la fusion de quantificateurs permet un gain important pour une procédure sans structure de quantificateurs. Sans la fusion, nous avons besoin d'introduire n variables quantifiées universellement lors de la mise sous forme prénexe, mais elles peuvent toutes fusionner en une seule. Les résultats sont présentés dans la table 3.

n		10	20	40	100	200	300	400	500
$\psi_{n,e}$	(sans fusion)	0,31	T	-	-	-	-	-	-
$\psi_{n,f}$	(avec fusion)	<0,1	<0,1	0,1	0,2	1,1	4,3	9,1	M

TAB. 3 – Résultats pour sKizzo, avec ou sans la fusion, sans construction de l'arbre de quantificateurs.

4.2 L'additionneur n-bits

Nous avons développé un algorithme en Prolog, qui permet de générer les instances de l'additionneur, de chercher le motif $(H \leftrightarrow (\exists x((F \leftrightarrow x) \wedge G)))$ et de le substituer par $(\exists x((F \leftrightarrow x) \wedge (H \leftrightarrow G)))$ ou d'appliquer la méthode décrite dans la section 3.3, puis d'effectuer la mise sous FNC. La recherche du motif est appliquée jusqu'à l'obtention d'un point fixe. La table 4 montre le nombre total de quantificateurs (NQ), le nombre de quantificateurs universels ($NQ^{\forall}$) et le nombre de clauses (NC) de la FNC de l'additionneur n-bits, pour différentes valeurs de n. Un indice i signifie qu'il s'agit de la mise sous forme prénexe classique, un e qu'il s'agit de la méthode de la section 3.3 (sans la fusion) et un t qu'il s'agit de la méthode par recherche de motifs (section 3.2).

Les motifs sont extraits dans la version existentielle. La mise sous forme prénexe appliquée ne cherche pas une forme optimale pour l'ordre des quantificateurs (ce qui demanderait une étude supplémentaire). La mise sous FNC est faite par introduction de variables intermédiaires quantifiées existentiellement qui ne fait croître que polynômialement la taille de la formule. Afin d'alléger la table 4, nous avons seulement détaillé quelques instances. Nous complétons ces informations en précisant que $NC_i = 2NC_t$, $NC_e = NC_i + 19$, $NQ_e = NQ_i + 8$ et $NQ_e^\forall = NQ_i^\forall$. Il est intéressant de noter, que pour un n donné, la formule $\psi_{n,e}$ possède légèrement plus de variables et de clauses que la formule $\psi_{n,i}$.

n	4	8	12	16	20	22
$NQ_i(NQ_i^\forall)$	281(16)	541(68)	801(100)	1061(132)	1321(164)	1451(180)
$NQ_t(NQ_t^\forall)$	147(14)	283(26)	419(38)	555(50)	691(62)	759(68)
NC_t	383	739	1095	1451	1807	1985

TAB. 4 – Tailles des QBF initiales et transformées pour l'additionneur.

Les tests ont été effectués sur la même machine que dans la section précédente et dans les mêmes conditions. QuBE n'a pas réussi à résoudre le problème avec les QBF $\phi_{n,i}$ et $\phi_{n,e}$ pour $n > 3$. Les résultats de la table 5 montrent que les procédures testées obtiennent presque toujours de meilleurs résultats avec les QBF transformées. La procédure sKizzo est la seule pour laquelle les résultats sont moins tranchés. Pour $n < 12$ l'amélioration est sensible, puis les performances se dégradent. Le traitement effectué sur les équivalences handicape sKizzo sur les instances plus grandes. QuBE6.3 ayant de moins bons résultats que QuBE-BJ1.2, nous nous posons des questions sur l'impact de la Q-résolution sur ce problème. Nous avons effectué des tests en désactivant la Q-résolution de sKizzo. Que ce soit avec les QBF initiales ($\phi_{n,i}$) ou les QBF transformées ($\phi_{n,e}$ et $\phi_{n,t}$), désactiver la Q-résolution permet d'obtenir de meilleurs temps. Avec les QBF $\phi_{n,t}$ le gain est significatif.

	sKizzo avec Q-résolution			sKizzo sans Q-résolution			QuBE		Quantor		
							6.3	BJ1.2			
n	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$	$\phi_{n,t}$	$\phi_{n,t}$	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$
4	0,3	0,2	**0,1**	0,5	0,2	**0,1**	3,0	0,3	8,5	0,3	**0,1**
6	1,0	0,9	**0,1**	1,3	0,4	**0,1**	T	17,9	M	10,5	3,0
8	5,5	2,9	0,4	13,3	1,2	**0,2**	T	1199.7	M	168,5	53,4
10	23,7	67,8	6,0	11,5	2,1	**0,4**	T	T	M	M	M
12	170,9	507,8	143,7	18,0	3,8	**0,5**	T	T	M	M	M
14	915,6	T	T	19,7	5,0	**0,8**	T	T	M	M	M
16	T	T	T	24,4	9,1	**0,9**	T	T	M	M	M
18	T	T	T	68,4	16,3	**1,6**	T	T	M	M	M
20	T	T	T	130,9	21,7	**1,3**	T	T	M	M	M
22	T	T	T	101,4	26,6	**2,2**	T	T	M	M	M

TAB. 5 – Résultats pour différentes procédures avec les QBF initiales et les QBF transformées.

Pour la vérification formelle de circuits logiques, la Q-résolution semble travailler à l'encontre des autres méthodes. Une explication possible est que la résolution sur les

variables intermédiaires (du codage ou de la mise sous FNC) enlève des possibilités aux procédures d'élaguer l'espace de recherche. Les mécanismes d'apprentissage retiennent des règles moins générales, le parcours de l'espace de recherche est plus long. Avec les QBF transformées, la Q-résolution a un impact très important sur les grandes instances, à tel point que les QBF initiales conviennent mieux à sKizzo. Les QBF transformées semblent être plus denses, augmentant l'importance des variables intermédiaires et la possibilité de réduire l'espace de recherche en raisonnant sur ces variables.

5 Conclusion

Dans cet article, nous proposons différentes équivalences logiques qui permettent de traiter les résultats intermédiaires contenus dans des équivalences ou des ou exclusifs pour les formules booléennes quantifiées. Ces équivalences permettent de conserver sa taille et le nombre de ses variables, tout en sortant les quantificateurs afin de pouvoir appliquer une mise sous forme prénexe. Il est possible de choisir le type de quantification de la variable intermédiaire et ainsi réduire le nombre d'alternances de quantificateurs. Nos équivalences peuvent être utilisées de deux façons : soit il faut coder les problèmes en les utilisant directement, soit il faut posséder le codage original du problème et effectuer des remplacements.

Puis, nous proposons l'introduction de résultats intermédiaires afin de réaliser la mise sous forme prénexe d'une formule booléenne quantifiée quelconque. Pour chaque quantificateur qui ne peut pas être sorti en conservant la taille de la formule, nous introduisons une variable le représentant ainsi que la formule sous sa portée. Nous montrons que cela permet de limiter la taille et le nombre de variables dans la formule mise sous forme prénexe de cette façon. Cette méthode construisant une conjonction de définitions de résultats intermédiaires avec le cœur de la formule, elle permet de facilement utiliser la fusion de quantificateurs universels. Nous exploitons cette possibilité afin de réduire le nombre de variables.

Nous présentons un problème théorique illustrant le cas limite. Nos différentes méthodes de mise sous forme prénexe permettent une réduction exponentielle de la taille de la formule et du nombre de variables par rapport à la méthode classique. Les procédures peuvent ainsi résoudre beaucoup plus d'instances de ce problème.

Pour illustrer l'intérêt pratique des nos propositions, nous avons choisi la vérification formelle de circuits logiques, où une spécification doit être équivalente à la structure du circuit. Nos résultats montrent que notre codage permet d'avoir de meilleurs résultats. Toutefois, il est possible de rechercher avec un sur-coût négligeable, les motifs de résultats intermédiaires afin d'effectuer un remplacement. Nous remarquons pour ce problème que la Q-résolution est néfaste au temps de résolution ce qui n'était pas le cas pour le problème théorique. Or, le codage de la vérification de l'additionneur possède de nombreux résultats intermédiaires. Il serait intéressant de voir, si la Q-résolution pourrait être de nouveau intéressante si elle ne s'appliquait pas sur ces résultats intermédiaires.

Il est possible d'étendre notre travail en proposant une mise sous forme clausale sans passer par la forme prénexe. Dans de futurs travaux, il serait donc intéressant de spécifier un format pour des QBF sous forme clausale non prénexes, afin que les procédures

prennent directement des formules dans ce format en entrée. En effet, les procédures utilisent de plus en plus une structure de quantificateurs : en interne, elles manipulent déjà une forme clausale non prénexe. Il serait intéressant de travailler directement sur la formulation originale (non prénexe et non FNC), de mettre ensuite en interne sous forme clausale non prénexe et de classer les variables en différentes catégories : les variables du problème, les résultats intermédiaires et les variables propres au codage. Nous pourrions alors exploiter plus d'informations et évaluer l'impact des heuristiques telle la Q-résolution sur ces différentes catégories.

Références

AYARI A. & BASIN D. (2002). Qubos : Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers. In *Formal Methods in Computer-Aided Design, Fourth International Conference, FMCAD 2002* : Springer-Verlag.

AYARI A., BASIN D. A. & FRIEDRICH S. (1999). Structural and Behavioral Modeling with Monadic Logics. In *ISMVL*, p. 142–151.

BENEDETTI M. (2005a). Quantifier Trees for QBFs. In F. BACCHUS & T. WALSH, Eds., *SAT*, volume 3569 of *LNCS*, p. 378–385 : Springer.

BENEDETTI M. (2005b). sKizzo : A Suite to Evaluate and Certify QBFs. In R. NIEUWENHUIS, Ed., *CADE*, volume 3632 of *LNCS*, p. 369–376 : Springer.

BIERE A. (2004). Resolve and Expand. In *7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'04)*.

BUBECK U. & BÜNING H. K. (2007). Bounded Universal Expansion for Preprocessing QBF. In J. MARQUES-SILVA & K. A. SAKALLAH, Eds., *SAT*, volume 4501 of *LNCS*, p. 244–257 : Springer.

BÜNING H. K., KARPINSKI M. & FLÖGEL A. (1995). Resolution for Quantified Boolean Formulas. *Inf. Comput.*, **117**(1), 12–18.

DAVIS M., LOGEMANN G. & LOVELAND D. W. (1962). A machine program for theorem-proving. *Commun. ACM*, **5**(7), 394–397.

DE LA TOUR T. B. (1992). An Optimality Result for Clause Form Translation. *J. Symb. Comput.*, **14**(4), 283–302.

EGLY U. (1996). On Different Structure-Preserving Translations to Normal Form. *J. Symb. Comput.*, **22**(2), 121–142.

EGLY U., SEIDL M., TOMPITS H., WOLTRAN S. & ZOLDA M. (2003). Comparing Different Prenexing Strategies for Quantified Boolean Formulas. In E. GIUNCHIGLIA & A. TACCHELLA, Eds., *SAT*, volume 2919 of *LNCS*, p. 214–228 : Springer.

GIUNCHIGLIA E., NARIZZANO M. & TACCHELLA A. (2004). QuBE++ : An Efficient QBF Solver. In A. J. HU & A. K. MARTIN, Eds., *FMCAD*, volume 3312 of *LNCS*, p. 201–213 : Springer.

GIUNCHIGLIA E., NARIZZANO M. & TACCHELLA A. (2006). Quantifier structure in search based procedures for qbfs. In *Proceedings of the conference on Design, automation and test in Europe (DATE'06)*, p. 812–817.

PLAISTED D. A. & GREENBAUM S. (1986). A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, **2**(3), 293–304.

TSEITIN G. S. (1970). On the complexity of derivation in propositional calculus. In A. O. SLISENKO, Ed., *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, p. 115–125. New York : Consultants Bureau.