Équivalences 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. *De nombreux problèmes d'intelligence artificielle et de vérification formelle se ramènent à un test de validité d'une formule booléenne quantifiée (QBF). Pour effectuer ce test les solveurs QBF actuels ont besoin d'une formule sous une forme syntaxique restrictive prénexe, comme la forme normale conjonctive ou la forme normale de négation. La mise sous cette forme prénexe préalable peut mener à une explosion de la taille de la formule en présence de bi-implications et de ou exclusifs. Dans ce travail nous décrivons une nouvelle méthode de mise sous forme prénexe à l'aide d'un motif particulier. Nos résultats expérimentaux montrent l'intérêt d'une telle approche pour les solveurs QBF actuels.*

ABSTRACT. *Many problems in artificial intelligence and in formal verification can be reduced to a validity test of a quantified boolean formula (QBF). To perform this test, existing QBF solvers need the formula to be in a restrictive prenex form, as conjunctive normal form or negation normal form. This prior prenexing process can lead to an exponential growth of the formula in presence of bi-implications and exclusive disjunctions. In this work, we describe a new method of prenexing with a particular pattern. Our experimental results show the interest of such a method with actual QBF solvers.*

MOTS-CLÉS : *Formules booléennes quantifiées, forme prénexe, fusion de quantificateurs.*

KEYWORDS: *Quantified boolean formulae, prenex form, fusion of quantifiers.*

2e soumission à *RIA*, le 20 septembre 2010

1. Introduction

Un intérêt constant est porté à l'amélioration des algorithmes pour SAT, le problème de satisfiabilité de la logique propositionnelle classique (PROP). En effet, de nombreuses tâches en représentation de connaissances peuvent être efficacement réduites à SAT, qui est le problème NP-complet canonique. Cette technique a été appliquée avec succès pour de nombreuses applications comme par exemple la conception de circuits intégrés, la planification classique, l'ordonnancement, la cryptographie et bien d'autres. La maturité des procédures de calcul permet aujourd'hui de résoudre certains problèmes industriels de grande taille.

Mais ce n'est pas suffisant pour une multitude d'importants problèmes de décision parmi des champs très divers et dont l'étude théorique démontre qu'ils possèdent une complexité supérieure à celle de SAT (Egly *et al.*, 2001b; Demaine, 2001). Le problème de validité pour les formules booléennes quantifiées (QBF), qui est une généralisation du problème SAT, est le problème PSPACE-complet canonique (Stockmeyer, 1977). C'est donc naturellement que le problème de décision des QBF a fait l'objet d'études, d'une part pour mettre au point des procédures de résolution efficaces et d'autre part pour exprimer différents problèmes dans ce langage.

En intelligence artificielle, de nombreuses tâches se réduisent à QBF (Egly *et al.*, 2000; Egly *et al.*, 2001a; Besnard *et al.*, 2009): l'abduction, l'existence d'extension stable en logique auto-épistémique, l'existence d'extension en logique des défauts, l'existence de modèle stable en programmation logique disjonctive, la circonscription propositionnelle et l'argumentation déductive. Les QBF peuvent aussi servir pour les jeux finis à deux joueurs (Demaine, 2001) comme le puissance 4 (Gent *et al.*, 2003) ou encore la planification conditionnelle (Rintanen, 1999).

Dans un autre domaine, la vérification formelle consiste à prouver ou refuter la correction d'un système devant respecter certaines propriétés en utilisant des méthodes formelles. L'utilisation de QBF comme langage pour certaines tâches permet d'obtenir des formules plus compactes et expressives que si elles avaient été décrites en logique propositionnelle (Benedetti *et al.*, 2008). Comme application nous pouvons citer, la vérification de circuits logiques et de descriptions de protocoles (Ayari *et al.*, 2000; Pan *et al.*, 2004; Ling *et al.*, 2005), la vérification de modèles (model checking et bounded model checking) (Dershowitz *et al.*, 2005; Biere, 2004), la vérification de convergence (Bryant *et al.*, 2003), le calcul de diamètre d'un système (Mneimneh *et al.*, 2003), le diagnostic de pannes (Scholl *et al.*, 2001; Mangassarian *et al.*, 2007; Staber *et al.*, 2007).

La plupart des procédures actuelles pour décider des QBF nécessitent d'avoir en entrée une formule sous forme normale conjonctive (Giunchiglia *et al.*, 2004a; 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, la bi-implication 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

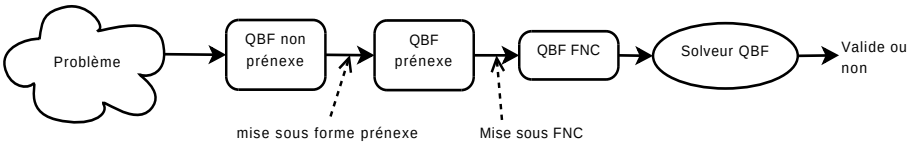


Figure 1. *Étapes menant à la décision d'un problème via les QBF.*

avec tous les quantificateurs devant la formule. La procédure qpro (Egly *et al.*, 2009) est une alternative intéressante mais elle se limite à la forme normale de négation et par conséquent n'accepte pas les bi-implications. Les différentes étapes menant à la décision d'un problème avec les QBF sont illustrées dans la figure 1.

La mise sous forme normale conjonctive a été largement étudiée (Plaisted *et al.*, 1986; de la Tour, 1992; Egly, 1996), car cette forme normale est utilisée pour la satisfiabilité des formules booléennes (non quantifiées). Cependant, il n'existe pas une unique forme prénexe associée à une QBF. Selon la stratégie choisie pour l'ordre d'extraction des quantificateurs, le résultat varie et par conséquent le temps de résolution peut être fortement influencé (Egly *et al.*, 2003; Giunchiglia *et al.*, 2006). En outre, aucune règle n'est fournie pour extraire les quantificateurs de formules booléennes quantifiées contenant des bi-implications 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. Dans cet article, nous commençons par rappeler quelques notions formelles nécéssaires. Puis lors la mise sous forme prénexe, nous nous intéressons à l'étape qui consiste à ré-écrire une formule sous une forme logiquement équivalente sans bi-implication ni ou exclusif. Enfin, nous utilisons des QBF codant la vérification formelle de circuits logiques et plus particulièrement l'additionneur n-bits pour illustrer en pratique l'intérêt de notre proposition.

2. Préliminaires

2.1. *La logique propositionnelle*

L'ensemble des variables (propositionnelles ou booléennes) est noté $\mathcal{V}$. Le symbole $\top$ (resp. $\bot$) est la constante propositionnelle représentant ce qui est toujours vrai (resp. faux). 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 $\{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$ est noté $\mathcal{O}$. L'équivalence logique est notée $\equiv$. Un littéral est une variable propositionnelle ou la négation de celle-ci. L'ensemble des propositions $PROP$ est défini inductivement comme suit: tout élément de $\mathcal{V} \cup \{\top, \bot\}$ 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 $o \in \mathcal{O}$ alors $(F\ o\ G)$

est élément de $PROP$. Soit a et b deux variables propositionnelles, alors la formule suivante appartient à $PROP$:

$$((a \wedge b) \leftrightarrow \neg(\neg b \vee \neg a))$$

Une formule est sous *forme normale de négation (FNN)* si elle est exclusivement constituée de conjonctions, de disjonctions et de littéraux. La formule suivante est sous FNN:

$$((a \wedge b) \vee (\neg b \vee \neg a))$$

Une formule est sous *forme normale conjonctive (FNC)* si c'est une conjonction de disjonctions de littéraux. La formule suivante est sous FNC:

$$((a \vee b) \wedge (\neg b \vee \neg a))$$

L'ensemble des valeurs booléennes $\mathbf{vrai}$ et $\mathbf{faux}$ est noté $BOOL$. L'ensemble des variables propositionnelles d'une proposition F est noté $\mathcal{V}(F)$. Une *valuation* v est une fonction de $\mathcal{V}(F)$ dans $BOOL$, qui affecte une valeur booléenne à chaque variable propositionnelle ; son extension de $PROP$ dans $BOOL$ est notée v^* et est défini avec la sémantique classique des opérateurs logiques. Une proposition F est satisfiable si il existe au moins une valuation v telle que l'interprétation booléenne de la proposition soit $\mathbf{vrai}$, c'est à dire $v^*(F) = \mathbf{vrai}$. Le problème SAT, consistant à décider si une proposition est satisfiable, est NP-complet.

2.2. *Les 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 : tout élément de $\mathcal{V} \cup \{\top, \bot\}$ 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 booléenne alors $(\exists x\ F)$ et $(\forall x\ F)$ sont des éléments de QBF. Par convention, des quantificateurs différents lient des variables différentes. Nous pouvons remarquer que si F est élément de $PROP$ alors F est également élément de QBF. Soit a,b et c trois variables booléennes, alors la formule suivante appartient à QBF:

$$(\exists a\ ((\exists b\ (a \wedge b)) \leftrightarrow \neg(\forall c\ (\neg c \vee \neg a))))$$

L'ensemble des variables d'une formule F est noté $\mathcal{V}(F)$. Une variable x est *libre* si et seulement si elle n'apparaît pas sous la portée d'un quantificateur $\exists x$ ou $\forall x$. L'ensemble des variables libres d'une formule F est noté $\mathcal{L}(F)$.

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 $q_1x_1 \dots q_nx_n$ avec $x_1, \dots, x_n$ des variables distinctes et $q_1 \dots q_n$ des quantificateurs. Une QBF QF est en *forme prénexe* si F est une formule booléenne, appelée *matrice* et Q est un *lieur*. La formule suivante est en forme prénexe:

$$\exists a \forall c \exists b((a \wedge b) \leftrightarrow \neg(\neg c \vee \neg a))$$

Une QBF prénexe QF est sous *forme normale de négation (FNN)* si F est une formule booléenne en forme normale de négation. La QBF prénexe suivante est sous FNN:

$$\exists a \forall c \exists b((a \wedge b) \vee (\neg c \vee \neg a))$$

De la même manière, une QBF prénexe QF 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 QBF prénexe suivante est sous FNC:

$$\exists a \forall c \exists b((a \vee b) \wedge (\neg c \vee \neg a \vee b))$$

La sémantique des quantificateurs est la suivante : pour toute variable x et QBF F,

$$(\exists x\ F) \equiv (F[x \leftarrow \top] \vee F[x \leftarrow \bot])$$
$$(\forall x\ F) \equiv (F[x \leftarrow \top] \wedge F[x \leftarrow \bot])$$

Une QBF F est *valide* si $F \equiv \top$. Enfin, rappelons que le problème (QBF) consistant à décider si une formule booléenne quantifiée est valide ou non est le problème canonique de la classe PSPACE-complet.

2.3. *Mise sous forme prénexe et mise sous forme normale conjonctive*

Parmi les étapes menant à la décision d'un problème avec les QBF (cf. figure 1), il y a la mise sous forme normale conjonctive, qui nécessite que la formule soit sous forme prénexe. La mise sous forme normale conjonctive a été largement étudiée (Plaisted *et al.*, 1986; de la Tour, 1992; Egly, 1996) et il existe des travaux sur l'ordre d'extraction des quantificateurs (Egly *et al.*, 2003; Giunchiglia *et al.*, 2006). Aucune règle n'est fournie pour extraire les quantificateurs de formules booléennes quantifiées contenant des bi-implications et des ou exclusifs, à l'exception d'un motif particulier (Da Mota *et al.*, 2010). D'une manière générale, la mise sous forme prénexe se décompose en trois étapes :

(i) La suppression des bi-implications et des ou exclusifs, grâce aux équivalences logiques suivantes :

$$(F \leftrightarrow G) \quad \equiv \quad ((G \rightarrow F) \wedge (F \rightarrow G)) \qquad [1]$$

$$(F \oplus G) \quad \equiv \quad ((G \vee F) \wedge (\neg F \vee \neg G)) \qquad [2]$$

Il n'est pas nécéssaire de supprimer une bi-implication ou un ou exclusif si F et G ne contiennent pas de quantificateur. 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 est généralement basée sur les équivalences suivantes, héritées de la logique classique du premier ordre, elles restent vraies pour les QBF. Soit F, G et H des QBF et x une variable booléenne ($x \notin \mathcal{L}(H)$), alors:

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

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.

Nous choisissons d'effectuer la mise sous forme normale conjonctive par introduction de variables booléennes afin que la taille de la QBF ne croisse que polynomialement (Tseitin, 1970):

$$
\begin{aligned}
F &= \exists a\forall b\exists c\exists d\exists e\exists f(((((a \leftrightarrow b) \leftrightarrow c) \leftrightarrow d) \leftrightarrow e) \leftrightarrow f) \\
&\equiv \exists a\forall b\exists c\exists d\exists e\exists f\exists v_1 \\
&\quad ((v_1 \leftrightarrow (a \leftrightarrow b)) \wedge ((((v_1 \leftrightarrow c) \leftrightarrow d) \leftrightarrow e) \leftrightarrow f)) \\
&\equiv \exists a\forall b\exists c\exists d\exists e\exists f\exists v_1\exists v_2((v_1 \leftrightarrow (a \leftrightarrow b)) \wedge (v_2 \leftrightarrow (v_1 \leftrightarrow c)) \wedge \\
&\quad (((v_2 \leftrightarrow d) \leftrightarrow e) \leftrightarrow f)) \\
&\equiv \exists a\forall b\exists c\exists d\exists e\exists f\exists v_1\exists v_2\exists v_3((v_1 \leftrightarrow (a \leftrightarrow b)) \wedge (v_2 \leftrightarrow (v_1 \leftrightarrow c)) \wedge \\
&\quad (v_3 \leftrightarrow (v_2 \leftrightarrow d)) \wedge ((v_3 \leftrightarrow e) \leftrightarrow f)) \\
&\quad \ldots \\
&\equiv \exists a\forall b\exists c\exists d\exists e\exists f\exists v_1\exists v_2\exists v_3\exists v_4\exists v_5 \\
&\quad ((v_1 \leftrightarrow (a \leftrightarrow b)) \wedge (v_2 \leftrightarrow (v_1 \leftrightarrow c)) \wedge (v_3 \leftrightarrow (v_2 \leftrightarrow d)) \wedge \\
&\quad (v_4 \leftrightarrow (v_3 \leftrightarrow e)) \wedge (v_5 \leftrightarrow (v_4 \leftrightarrow f)) \wedge v_5) \\
&\equiv \exists a\forall b\exists c\exists d\exists e\exists f\exists v_1\exists v_2\exists v_3\exists v_4\exists v_5(\\
&\quad (\neg v_1 \vee a \vee \neg b) \wedge (\neg v_1 \vee \neg a \vee b) \wedge (v_1 \vee a \vee b) \wedge \\
&\quad (v_1 \vee \neg a \vee \neg b) \wedge (\neg v_2 \vee v_1 \vee \neg c) \wedge (\neg v_2 \vee \neg v_1 \vee c) \wedge \\
&\quad (v_2 \vee v_1 \vee c) \wedge (v_2 \vee \neg v_1 \vee \neg c) \wedge (\neg v_3 \vee v_2 \vee \neg d) \wedge \\
&\quad (\neg v_3 \vee \neg v_2 \vee d) \wedge (v_3 \vee v_2 \vee d) \wedge (v_3 \vee \neg v_2 \vee \neg d) \wedge \\
&\quad (\neg v_4 \vee v_3 \vee \neg e) \wedge (\neg v_4 \vee \neg v_3 \vee e) \wedge (v_4 \vee v_3 \vee e) \wedge \\
&\quad (v_4 \vee \neg v_3 \vee \neg e) \wedge (\neg v_5 \vee v_4 \vee \neg f) \wedge (\neg v_5 \vee \neg v_4 \vee f) \wedge \\
&\quad (v_5 \vee v_4 \vee f) \wedge (v_5 \vee \neg v_4 \vee \neg f) \wedge v_5)
\end{aligned}
$$

La même formule possédant k variables est mise sous forme normale conjonctive à l'aide de $k-1$ nouvelles variables booléennes quantifiées existentiellement. Cette forme possède $1+4(k-1)$ clauses de longueur 3.

Une autre méthode serait d'utiliser la distributivité du $\vee$ et du $\wedge$. Cette méthode peut potentiellement faire croître exponentiellement la taille de la formule et est plus coûteuse en temps. La formule F possède 6 variables booléennes, soit 2^6 valuations possibles dont 2^5 valuations qui falsifient la matrice de F. Sa mise sous forme normale conjonctive à l'aide de la distributivité possède 2^5 clauses de longueur 6. Pour une formule semblable possédant k variables booléennes, sa mise sous forme normale conjonctive possède 2^{k-1} clauses de longueur k.

3. Mise sous forme prénexe et bi-implications

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 la bi-implication ou du ou exclusif.

Alors que la bi-implication représente le "si et seulement si" du langage naturel, le ou exclusif représente sa négation. Le pouvoir d'expression de la bi-implication s'étend bien au delà de la simple équivalence logique: $(F \leftrightarrow G) \equiv ((G \rightarrow F) \wedge (F \rightarrow G))$. Un quantificateur présent dans F et G, de part les équivalences logiques de la section 2.3, 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 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 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:

$$\phi \overset{9,7}{\equiv} \exists a\, \forall b((\phi_1 \rightarrow \phi_2) \wedge (\phi_2' \rightarrow \phi_1)) \qquad [17]$$

L'ordre d'extraction des quantificateurs aurait pu être inversé, nous aurions obtenu:

$$\phi \overset{7,9}{\equiv} \forall b\, \exists a((\phi_1 \rightarrow \phi_2) \wedge (\phi'_2 \rightarrow \phi_1)) \qquad [18]$$

Les formules [17] et [18] sont équivalentes, or 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. Sans traiter directement les bi-implications il est impossible d'éviter cet écueil.

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 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 la bi-implication. Nous mettons ϕ_g sous forme prénexe :

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

puis ϕ_d:

$$\begin{array}{lll}
\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 & \equiv & ((\phi_1 \rightarrow ((\phi_2 \rightarrow (\exists a\, \phi_3)) \wedge ((\exists b\, \phi_3[a \leftarrow b]) \rightarrow \phi_2))) \wedge \\
 & & (((\phi_2 \rightarrow (\exists c\, \phi_3[a \leftarrow c])) \wedge ((\exists d\, \phi_3[a \leftarrow d]) \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{13,16}{\equiv} & ((\phi_1 \rightarrow (\exists a\, (\phi_2 \rightarrow \phi_3) \wedge \forall b\, (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & ((\exists c\, (\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge \forall d\, (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{9,7}{\equiv} & ((\phi_1 \rightarrow \exists a \forall b\, ((\phi_2 \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & ((\exists c\, (\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge \forall d\, (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{7,9}{\equiv} & ((\phi_1 \rightarrow \exists a \forall b\, ((\phi_2 \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & (\forall d \exists c\, ((\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{16,9}{\equiv} & \exists a((\phi_1 \rightarrow \forall b\, ((\phi_2 \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & (\forall d \exists c\, ((\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{14,9}{\equiv} & \exists a \exists d((\phi_1 \rightarrow \forall b\, ((\phi_2 \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & (\exists c\, ((\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{15,7}{\equiv} & \exists a \exists d \forall b((\phi_1 \rightarrow ((\phi_2 \rightarrow \phi_3) \wedge (\phi_3[a \leftarrow b] \rightarrow \phi_2))) \wedge \\
 & & (\exists c\, ((\phi_2 \rightarrow \phi_3[a \leftarrow c]) \wedge (\phi_3[a \leftarrow d] \rightarrow \phi_2)) \rightarrow \phi_1)) \\
\phi_d & \overset{15,7}{\equiv} & \exists a \exists d \forall b \forall c\, ((\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 c]) \wedge (\phi_3[a \leftarrow d] \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 variables croissent exponentiellement avec le nombre de bi-implications 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. Nous remarquerons que très vite la mise sous forme prénexe classique ne permet pas de résoudre ce cas. Dans notre travail, nous proposons de nouvelles équivalences et stratégies afin de limiter les recopies et les pertes d'informations.

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), dans un précédent travail (Da Mota *et al.*, 2010), nous nous sommes intéressés au motif $\exists x((x \leftrightarrow F) \wedge G)$, avec $x \notin \mathcal{L}(F)$, qui est équivalent à $G[x \leftarrow F]$. Dans notre terminologie nous désignons une telle variable x, *variable intermédiaire* ou *résultat intermédiaire* et nous appelons $(x \leftrightarrow F)$ la *définition de* x. 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. Cependant, cela peut conduire à une croissance exponentielle de la taille de la formule. Nous donnons, en versions étendues, les deux théorèmes centraux présentés dans (Da Mota *et al.*, 2010). Ils sont utiles à la compréhension et à la justification de notre proposition.

Théorème 1. *Soit F et G des QBF et x une variable représentant le résultat intermédiaire F, avec $x \notin \mathcal{L}(F)$, alors*

$$(\exists x((F \leftrightarrow x) \wedge G)) \equiv (\forall x((F \leftrightarrow x) \rightarrow G)) \qquad [19]$$

A l'aide de ce théorème, il est possible de choisir comment quantifier une variable intermédiaire. Nous appelons la version existentielle (resp. universelle) déclaration de résultats intermédiaires (resp. déclaration de domaine). 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 bi-implication (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*

$$(H\ o\ (\exists x((F \leftrightarrow x) \wedge G))) \equiv (\exists x((F \leftrightarrow x) \wedge (H\ o\ G))) \quad [20]$$

Le quantificateur qui porte sur x est extrait de la bi-implication ou du ou exclusif, aucune variable n'est créée, aucune sous formule n'est dupliquée. 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. Le théorème 2 valide cette intuition, en permettant de faire jouer un seul des deux joueurs, au choix, lors d'un résultat intermédiaire dans une bi-implication ou sa négation, sans augmenter ni la taille de la formule ni le nombre de variables.

3.3. *Mise sous forme prénexe de 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, sous formule 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.

Démonstration.
Cas de base:

Si $G = F$ et $G' = x$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge x) \equiv (((\top \leftrightarrow F) \wedge \top) \vee ((\bot \leftrightarrow F) \wedge \bot)) \equiv (\top \leftrightarrow F) \equiv F$

Si $G = y$ et $G' = y$, avec y une variable, alors:

$G \equiv \exists x((x \leftrightarrow F) \wedge y) \equiv (\exists x(x \leftrightarrow F) \wedge y) \equiv (\top \wedge y) \equiv y$
Même démonstration pour $G = \top$ et $G = \bot$

Soit F et J des QBF, telles que F est une sous formule de J. J ne possède pas de quantificateur portant sur les variables de F autres que ceux de F. Soit x une variable telle que $x \notin \mathcal{L}(J)$. Nous appelons J', la formule J dans laquelle chaque occurrence de F est remplacée par x, alors: $J \equiv \exists x((x \leftrightarrow F) \wedge J')$
L'induction s'effectue sur la QBF G obtenue en composant J avec: $Qy, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus$ et si besoin une QBF H. y est variable, par hypothèse $y \notin \mathcal{L}(F)$, $Q \in \{\exists, \forall\}$. Pour les opérateurs binaires, $G = H\ o\ J$ et par hypothèse $x \notin \mathcal{L}(H)$. Donc $G' = H\ o\ J'$ avec $o \in \{\wedge, \vee, \rightarrow, \leftrightarrow, \oplus\}$.
Par hypothèse nous pouvons appliquer le théorème 1 (Da Mota *et al.*, 2010), rappelé dans la section 3.2. Soient F et J' deux QBF et x une variable qui représente un resultat intermédiaire F, avec $x \notin \mathcal{L}(F)$, alors $(\exists x\ ((x \leftrightarrow F) \wedge J')) \equiv (\forall x\ ((x \leftrightarrow F) \rightarrow J'))$.

Si $G = Qy\ J$ et $G' = Qy\ J'$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge (Qy\ J')) \equiv Qy\ \exists x((x \leftrightarrow F) \wedge J') \equiv Qy\ J$

Si $G = \neg J$ et $G' = \neg J'$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge (\neg J')) \equiv \forall x((x \leftrightarrow F) \rightarrow (\neg J')) \equiv \forall x((\neg(x \leftrightarrow F)) \vee (\neg J'))$
$G \equiv \forall x(\neg((x \leftrightarrow F) \wedge J')) \equiv \neg(\exists x((x \leftrightarrow F) \wedge J')) \equiv \neg J$

Si $G = (H \wedge J)$ et $G' = (H \wedge J')$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge (H \wedge J')) \equiv (H \wedge (\exists x((x \leftrightarrow F) \wedge J'))) \equiv (H \wedge J)$

Si $G = (H \vee J)$ et $G' = (H \vee J')$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge (H \vee J')) \equiv \forall x((x \leftrightarrow F) \rightarrow (H \vee J'))$
$G \equiv \forall x(H \vee ((x \leftrightarrow F) \rightarrow J')) \equiv H \vee (\forall x((x \leftrightarrow F) \rightarrow J')$
$G \equiv H \vee (\exists x((x \leftrightarrow F) \wedge J')) \equiv (H \vee J)$
Même démonstration pour $G = (H \rightarrow J)$ et $G' = (H \rightarrow J')$

Si $G = (H \leftrightarrow J)$ et $G' = (H \leftrightarrow J')$, alors:
$G \equiv \exists x((x \leftrightarrow F) \wedge (H \leftrightarrow J')) \equiv H \leftrightarrow (\exists x((x \leftrightarrow F) \wedge J')) \equiv (H \leftrightarrow J)$
Par utilisation de l'équivalence [20] du théorème 2 (Da Mota *et al.*, 2010), rappelé dans la section 3.2. Même démonstration pour $G = (H \oplus J)$ et $G' = (H \oplus J')$

□

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 $G = \phi_d$.

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

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 la bi-implication. Puisque nous considérons que la variable a n'est pas un résultat intermédiaire, il faut supprimer la bi-implication 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 de bi-implications 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 2.3). A l'aide du théorème 3, il est possible d'effectuer une mise sous forme prénexe contenant une chaîne de $\wedge$ dont toutes les sous formules sauf une possèdent au moins un quantificateur universel à extraire. La règle [11] peut s'appliquer facilement afin de réduire le nombre de quantificateurs universels. L'exemple suivant illustre cette possibilité.

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 1. Nous appliquons ce théorème en priorité.

$$
\begin{array}{lcl}
\phi & \overset{th_1}{\equiv} & \exists x \exists y \exists z\,[((x \leftrightarrow y) \leftrightarrow z) \wedge \\
& & (x \leftrightarrow (\forall a\ \phi_1)) \wedge (y \leftrightarrow (\exists b\ \phi_2)) \wedge (z \leftrightarrow (\exists c\ \phi_3))]
\end{array}
$$

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

La première étape consiste à extraire tous les maillons de la chaîne de bi-implications contenant des quantificateurs à l'aide du théorème 1. Puis nous effectuons une mise sous forme prénexe classique sauf que la règle [11] (cf. section 2.3) est appliquée, sans quoi nous aurions obtenu:

$$
\begin{array}{lll}
\phi & \overset{7}{\equiv} & \exists x \exists y \exists z \exists a' \exists b \exists c \forall a \forall b' \forall c' [((x \leftrightarrow y) \leftrightarrow z) \wedge \\
& & ((x \rightarrow \phi_1) \wedge (\phi_1[a \leftarrow a'] \rightarrow x)) \wedge \\
& & ((y \rightarrow \phi_2) \wedge (\phi_2[b \leftarrow b'] \rightarrow y)) \wedge \\
& & ((z \rightarrow \phi_3) \wedge (\phi_3[c \leftarrow c'] \rightarrow z))]
\end{array}
$$

4. Problèmes utilisés pour l'étude expérimentale

4.1. *Chaînes de bi-implications 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 reprenons la classe de formules ψ_n définie dans (Da Mota *et al.*, 2010). 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. Nous rappelons la forme de la formule ψ_n. Pour $n \geq 2$, ψ_n est de la forme:

$$
\begin{array}{l}
\psi_n = \exists e_0, ..., e_{n-2} \forall u_0, u_1, u_2 \\
(\ (\exists x_n((x_n \leftrightarrow (e_{n-4} \vee e_{n-3})) \wedge (x_n \wedge e_{n-2}))) \leftrightarrow \\
\quad (\ (\exists x_{n-1}((x_{n-1} \leftrightarrow (e_{n-5} \vee e_{n-4})) \wedge (x_{n-1} \wedge e_{n-3}))) \leftrightarrow \\
\qquad ... \\
\qquad\quad (\ (\exists x_3((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))) \leftrightarrow \\
\qquad\qquad (\ (\exists x_2((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))) \leftrightarrow \\
\qquad\qquad\quad (\exists x_1((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2)))\ )\ )...))
\end{array}
$$

Le but étant d'étudier le cas général et non de nous comparer à l'extraction directe du motif présentée dans (Da Mota *et al.*, 2010), nous considérons les x_k comme n'étant pas des résultats intermédiaires. Aussi nous introduisons des variables intermédiaires v_k à l'aide du théorème 3, telles que:

$$
\begin{array}{l}
\psi_n = \exists e_0, ..., e_{n-2} \forall u_0, u_1, u_2 \exists v_1, ..., v_n \\
(v_n \leftrightarrow (\exists x_n((x_n \leftrightarrow (e_{n-4} \vee e_{n-3})) \wedge (x_n \wedge e_{n-2}))))\wedge \\
(v_{n-1} \leftrightarrow (\exists x_{n-1}((x_{n-1} \leftrightarrow (e_{n-5} \vee e_{n-4})) \wedge (x_{n-1} \wedge e_{n-3}))))\wedge \\
... \\
(v_3 \leftrightarrow (\exists x_3((x_3 \leftrightarrow (u_2 \vee e_0)) \wedge (x_3 \wedge e_1))))\wedge \\
(v_2 \leftrightarrow (\exists x_2((x_2 \leftrightarrow (u_1 \vee u_2)) \wedge (x_2 \wedge e_0))))\wedge \\
(v_1 \leftrightarrow (\exists x_1((x_1 \leftrightarrow (u_0 \vee u_1)) \wedge (x_1 \wedge u_2))))\wedge \\
(v_n \leftrightarrow (v_{n-1} \leftrightarrow ...(v_3 \leftrightarrow (v_2 \leftrightarrow v_1)))))
\end{array}
$$

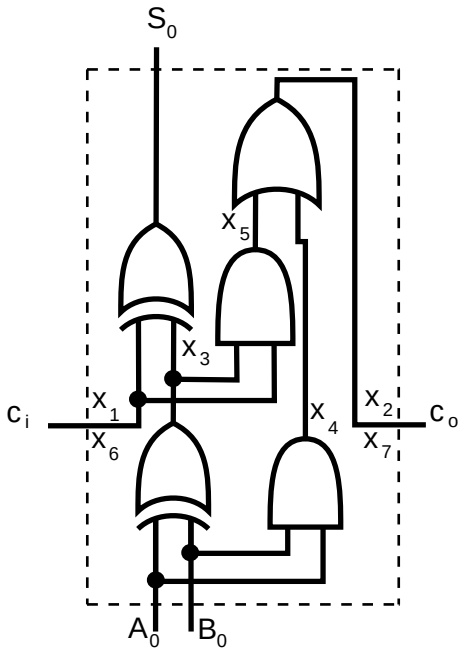
La suite de la mise sous forme prénexe est classique, toujours sans prendre en compte les résultats intermédiaires x_k.

4.2. *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 *et al.*, 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) :

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



$(x_1 \leftrightarrow c_i)$,
$(x_6 \leftrightarrow c_i)$,
$(x_2 \leftrightarrow c_o)$,
$(x_7 \leftrightarrow c_o)$,
$(x_3 \leftrightarrow (A_0 \oplus B_0))$,
$(x_4 \leftrightarrow (A_0 \wedge B_0))$,
$(x_5 \leftrightarrow (x_1 \wedge x_3))$,
$(S_0 \leftrightarrow (x_1 \oplus x_3))$,
$(x_2 \leftrightarrow (x_4 \vee x_5))$,

Figure 2. *Une implantation de l'additionneur 1-bit.*

Les formules add_{struc} et add_{comp} contiennent des variables existentielles, qui représentent des résultats intermédiaires, entre autres les retenues (cf. figure 2). 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.

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

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 la bi-implication).

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

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 de l'équivalences logiques [19], puis mise sous forme prénexe selon l'algorithme classique.

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

Il s'agit de la version proposée dans (Da Mota *et al.*, 2010). 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 la bi-implication. Afin d'étudier le cas général , nous considérons les x_k comme n'étant pas des résultats intermédiaires. Aussi nous introduisons des variables intermédiaires v_k à l'aide du théorème 3. La QBF ϕ_e est la QBF initiale ϕ_1 mise sous forme prénexe de cette façon:

$$\begin{array}{l}\phi_e = \forall c_i \forall c_o \forall A_0 \forall B_0 \forall S_0 \exists v_0 \exists v_1 \\ (v_0 \leftrightarrow [\exists x_1 \exists x_2 (((C_1(x_1) \wedge C_2(x_2)) \wedge \\ \quad (\exists x_3 \exists x_4 \exists x_5 (C_3(x_3) \wedge (F_1(x_3, x_1) \wedge (C_4(x_4) \wedge \\ \quad (C_5(x_5, x_1, x_3) \wedge F_2(x_2, x_5, x_4)))))))])) \wedge \\ (v_1 \leftrightarrow [\exists x_6 \exists x_7 ((C_1(x_6) \wedge C_2(x_7)) \wedge (F_3(x_6, x_7) \wedge F_4(x_6)))]) \wedge \\ (v_0 \leftrightarrow v_1)\end{array}$$

La variable v_0 représente la définition structurelle, alors que la variable v_1 représente la définition comportementale. La suite de la mise sous forme prénexe est faite à l'aide de la méthode classique, sans prendre en compte les résultats intermédiaires:

$$\begin{array}{l}\phi_e = \forall c_i \forall c_o \forall A_0 \forall B_0 \forall S_0 \exists v_0 \exists v_1 \\ \exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \exists x_7 \forall y_1 \forall y_2 \forall y_3 \forall y_4 \forall y_5 \forall y_6 \forall y_7 \\ (([C_1(y_1) \wedge C_2(y_2) \wedge C_3(y_3) \wedge (F_1(y_3, y_1) \wedge C_4(y_4) \wedge C_5(y_5, y_1, y_3) \wedge \\ \quad F_2(y_2, y_5, y_4)] \rightarrow v_0) \wedge (v_0 \rightarrow [C_1(x_1) \wedge C_2(x_2) \wedge C_3(x_3) \wedge \\ \quad F_1(x_3, x_1) \wedge C_4(x_4) \wedge C_5(x_5, x_1, x_3) \wedge F_2(x_2, x_5, x_4)])) \wedge \\ ((v_1 \rightarrow [C_1(x_6) \wedge C_2(x_7) \wedge F_3(x_6, x_7) \wedge F_4(x_6)]) \wedge \\ \quad ([C_1(y_6) \wedge C_2(y_7) \wedge F_3(y_6, y_7) \wedge F_4(y_6)] \rightarrow v_1)) \wedge \\ (v_0 \leftrightarrow v_1)\end{array}$$

Une fois mise sous forme prénexe, les formules ϕ_e et ϕ_i ont une taille comparable.

5. Résultats expérimentaux

5.1. *Chaîne de bi-implications*

Nous avons développé un algorithme en Prolog, qui permet de générer les instances des chaînes de bi-implications (cf. section 4.1), puis d'effectuer la mise sous FNC. Notons que les instances sont invalides.

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

Tableau 1. *Tailles des QBF transformées pour les chaînes de bi-implications.*

Le tableau 1 montre les préfixes et le nombre de clauses (NC_f et NC_t), pour différentes valeurs de n, de la FNC des chaînes de bi-implications ψ_n avec la méthode de la section 3.3 et 3.4 ($\psi_{n,f}$) et avec l'extraction des résultats intermédiaires ($\psi_{n,t}$) avec la méthode présentée dans (Da Mota *et al.*, 2010). Nous n'avons pas donné les préfixes des chaînes de bi-implications 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 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 la méthode qui traite le cas général permet de réduire exponentiellement la taille de la formule mise sous FNC par rapport à la méthode classique. Elle permet ainsi de mettre sous forme prénexe de façon efficace en taille l'ensemble des formules de QBF à l'instar de la méthode par extraction de résultats intermédiaires qui ne traite qu'un cas particulier, mais de façon plus compacte.

Nous allons maintenant mesurer l'impact de notre proposition sur le temps de résolution des problèmes par différentes procédures. Les tests ont été effectués sur un Intel(R) Xeon(R) à 2.50GHz avec 4Go de mémoire vive disponible. Les résultats sont résumés dans les figures 3,4 et 5. Chaque procédure a un temps limité à 3600 secondes. 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.*, 2004a) qui étend aux QBF la procédure de Davis-Logemann-Loveland (Davis *et al.*, 1962) et QuBE6.5 une version plus récente qui utilise un préprocesseur (Bubeck *et al.*, 2007) intégrant la Q-résolution. Pour ce problème, QuBE6.5 donne toujours de meilleurs résultats que QuBE-BJ1.2.

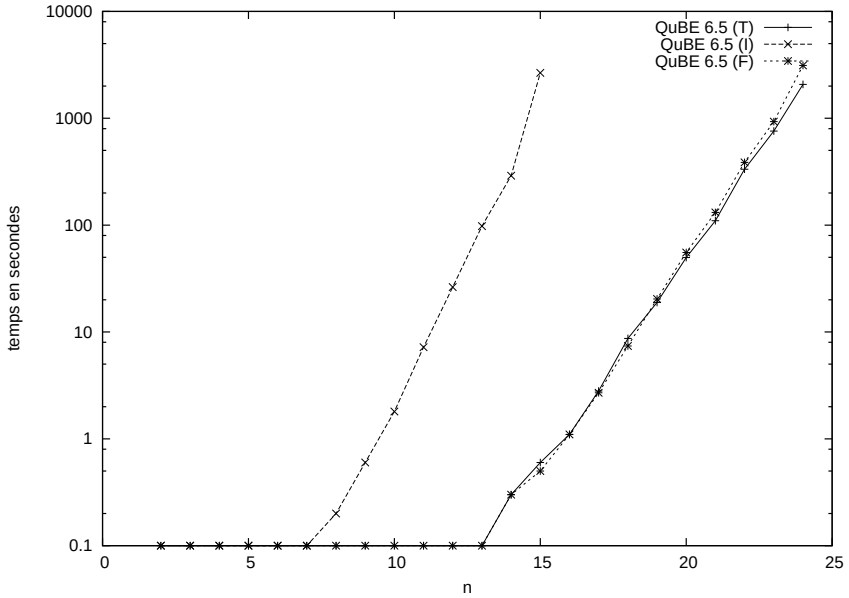


Figure 3. *Temps de calcul pour QuBE6.5 avec les chaînes de bi-implications mises sous forme prénexe de façon classique (I), par extraction des variables intermédiaires (T) et à l'aide du théorème 3 avec fusion de quantificateurs (F). Le temps en ordonnée est en échelle logarithmique.*

Sur cet exemple, la méthode de la section 3.3 donne d'aussi bons résultats que la méthode traitant particulièrement les résultats intermédiaires, Or ces deux méthodes ne sont pas incompatibles. Nous avons juste voulu traiter la chaîne de bi-implications de façon générale. Par souci de clarté, nous n'avons pas consigné 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 le tableau 2, demandent une étude supplémentaire. En effet, les trois procédures utilisent une structure de quantificateurs (Giunchiglia *et al.*, 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

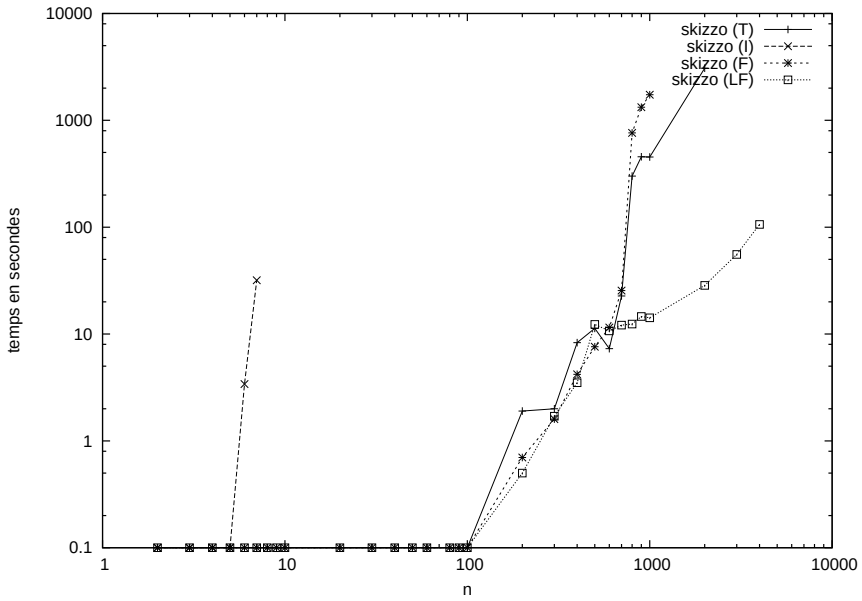


Figure 4. *Temps de calcul pour sKizzo avec les chaînes de bi-implications mises sous forme prénexe de façon classique (I), par extraction des variables intermédiaires (T), à l'aide du théorème 3 avec fusion de quantificateurs (F) et à l'aide du théorème 1 avec fusion de quantificateurs en désactivant la construction de l'arbre de quantificateurs (LF). Les deux axes sont en échelle logarithmique.*

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 le tableau 2.

n		20	30	100	200	800	1000	2000	4000
$\psi_{n,e}$	(sans fusion)	199,4	M	-	-	-	-	-	-
$\psi_{n,f}$	(avec fusion)	<0,1	<0,1	<0,1	0,5	12,4	14,2	28,5	106,2

Tableau 2. *Résultats pour sKizzo, avec ou sans la fusion, sans construction de l'arbre de quantificateurs.*

Les résultats sont même meilleurs qu'avec la construction de l'arbre des quantificateurs. Une explication probable vient des formules: chaque maillon possède 3 variables libres, chaque variable libre est présente dans 3 maillons et 3 variables libres ne sont présentes ensembles que dans un unique maillon. Aussi, la construction de l'arbre choisit arbitrairement des quantificateurs afin de réduire leur portée. Cet exemple montre que ce n'est pas toujours la meilleure solution et cela rejoint les résultats de (Lonsing *et al.*, 2009). Plus surprenant, ce sont ces résultats (avec fusion et

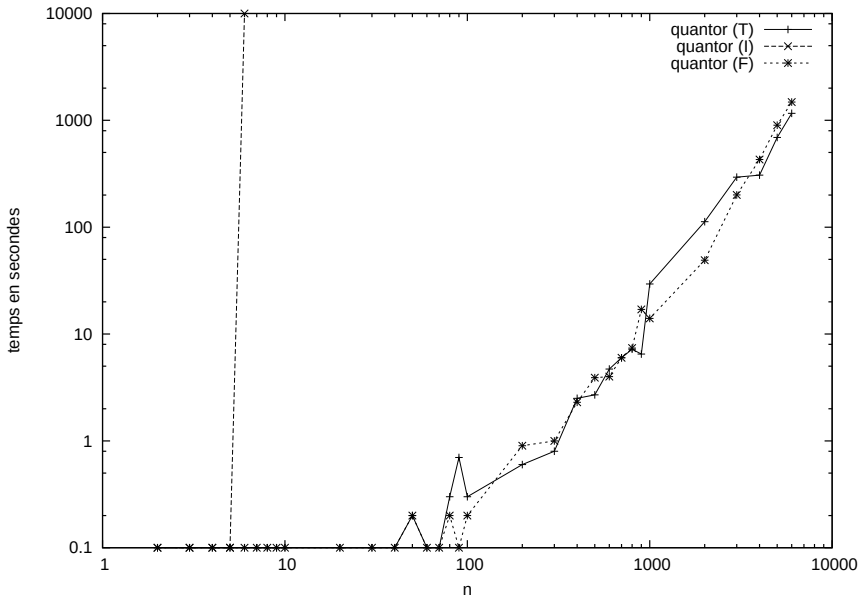


Figure 5. *Temps de calcul pour quantor avec les chaînes de bi-implications mises sous forme prénexe de façon classique (I), par extraction des variables intermédiaires (T) et à l'aide du théorème 3 avec fusion de quantificateurs (F). Les deux axes sont en échelle logarithmique.*

sans reconstruction de l'arbre de quantificateurs) qui sont bien meilleurs que ceux de sKizzo (avec ou sans reconstruction de l'arbre de quantificateur) ou quantor avec $\psi_{n,t}$, alors que nous pensions que $\psi_{n,t}$ serait la référence. En extrayant chaque maillon entièrement, il ne reste plus que les variables des définitions des maillons dans la chaîne de bi-implication. (cf. fin de la section 4.1). Alors qu'après l'extraction des résultats intermédiaires, ce sont toujours des formules qui sont dans les maillons. Nous pouvons constater sur la courbe (LF) de la figure 4, un point d'inflexion vers $n = 500$. Or n représente le nombre de maillons, donc aussi le nombre de variables universelles qui fusionnent lors de la mise sous forme prénexe. Nous pensons qu'à partir de $n = 500$, Le nombre d'occurrences de la variable résultat de la fusion dépasse un certain seuil au niveau d'une heuristique de choix de variables dans sKizzo. Ce choix permet probablement d'élaguer plus vite l'espace de recherche. Avec sKizzo, notre méthode générique est plus efficace sur ce problème spécifique, à condition de désactiver la construction de l'arbre de quantificateur (cf. figure 4).

5.2. *L'additionneur n-bits*

Nous avons développé un algorithme en Prolog, qui permet de générer les instances de l'additionneur et d'appliquer la méthode décrite dans la section 3.3, puis d'effectuer la mise sous FNC. La table 3 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 (Da Mota *et al.*, 2010).

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 le tableau 3, 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
$NQ_i(NQ_i^{\forall})$	281(16)	541(68)	801(100)	1061(132)	1321(164)
$NQ_t(NQ_t^{\forall})$	147(14)	283(26)	419(38)	555(50)	691(62)
NC_t	383	739	1095	1451	1807

Tableau 3. *Tailles des QBF initiales et transformées pour l'additionneur.*

Les tests ont été effectués sur un Intel(R) pentium(R) 4 à 2.80GHz avec 600Mo de mémoire vive disponible, puis ont été effectués sur un Intel(R) Xeon(R) à 2.50GHz avec 4Go de mémoire vive. Les résultats sont résumés dans les tableaux 4 et 5, 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. QuBE n'a pas réussi à résoudre le problème avec les QBF $\phi_{n,i}$ et $\phi_{n,e}$ pour $n > 3$.

Dans (Da Mota *et al.*, 2010) nous remarquons que pour cet exemple, la Q-résolution semble travailler à l'encontre des autres méthodes. Notre explication était 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. Nous avons effectué des tests en désactivant la Q-résolution de sKizzo. Sur une machine avec plus de mémoire, les résultats sont différents, même si désactiver la Q-résolution permet d'obtenir un gain de temps significatif pour les QBF $\phi_{n,t}$ et $\phi_{n,e}$. Il semble que pour ce problème la Q-résolution nécessite une grande quantité de mémoire. L'élimination de variable à l'aide de la Q-résolution peut poten-

	sKizzo sans Q-résolution			QuBE 6.5	QuBE BJ1.2	Quantor		
n	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$	$\phi_{n,t}$	$\phi_{n,t}$	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$
4	0,2	**0,1**	**0,1**	0,7	**0,1**	5,3	**0,1**	**0,1**
6	1,3	0,2	**0,1**	61,4	9,8	T	5,8	1,8
8	3,3	0,6	**0,1**	T	714,5	M	122,9	106,0
10	3,2	1,0	**0,1**	T	T	M	M	M
12	4,8	1,7	**0,1**	T	T	M	M	M
14	5,3	2,1	**0,3**	T	T	M	M	M
16	6,5	3,8	**0,3**	T	T	M	M	M
18	14,0	6,2	**0,5**	T	T	M	M	M
20	28,9	7,8	**0,4**	T	T	M	M	M
22	19,6	10,3	**1,0**	T	T	M	M	M

Tableau 4. *Résultats obtenus avec le Xeon(R) 2,5GHz/4Go, pour différentes procédures, avec les QBF initiales et les QBF transformées.*

tiellement générer un grand nombre de clauses et l'utilisation intensive de la mémoire a aussi un coût en temps. La table 5 résume les tests effectués sur les 2 machines différentes. Bien que la vitesse des processeurs soient différentes, cela ne suffit pas à expliquer les temps de calculs. En effet, d'une machine à l'autre, les temps ne sont pas proportionnels à la vitesse du processeur. Nous nous appuyons sur ces résultats pour conforter l'hypothèse que la Q-résolution nécessite une grande quantité de mémoire pour ce problème, ce qui impacte directement le temps de calcul.

	sKizzo avec Q-résolution Pentium(R) 4 - 600Mo			sKizzo avec Q-résolution Xeon(R) - 4Go		
n	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$	$\phi_{n,i}$	$\phi_{n,e}$	$\phi_{n,t}$
4	0,3	0,2	**0,1**	**0,1**	**0,1**	**0,1**
6	1,0	0,9	**0,1**	0,4	0,4	**0,1**
8	5,5	2,9	0,4	2,0	1,2	**0,2**
10	23,7	67,8	6,0	7,5	10,7	**1,6**
12	170,9	507,8	143,7	14,1	23,2	**11,9**
14	915,6	T	T	16,3	74,0	**15,6**
16	T	T	T	33,5	74,4	**29,3**
18	T	T	T	19,3	49,4	**14,4**
20	T	T	T	82,5	78,5	**21,7**
22	T	T	T	35,7	122,9	**14,0**

Tableau 5. *Résultats de sKizzo pour l'additionneur sur différentes machines.*

6. Conclusion

Dans nos travaux précédents (Da Mota *et al.*, 2010), nous nous occupions de rechercher et d'extraire les résultats intermédiaires. Dans cet article, nous introduisons des résultats intermédiaires afin de traiter, efficacement en espace, la mise sous forme

prénexe de toutes les formules du langage QBF. 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. Notre méthode de mise sous forme prénexe permet une réduction exponentielle de la taille de la formule et du nombre de variables par rapport à la méthode classique. Les procédures de calcul peuvent ainsi résoudre beaucoup plus d'instances de ce problème. De par la structure du problème, la méthode utilisant la fusion de quantificateurs permet d'obtenir les meilleurs résultats en désactivant la reconstruction de l'arbre de quantificateurs par sKizzo.

Nous avons repris le cas pratique de 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 qu'avec la mise sous forme prénexe classique, bien qu'il possède plus de variables et plus de clauses. Nous avons ignorer les résultats intermédiaires présents et nous en avons créé afin de les extraire, dans le but d'étudier uniquement le cas général. Il est intuitif d'obtenir de moins bons résultats qu'avec l'extraction directe des motifs. Toutefois, les méthodes peuvent être combinées: il est possible de rechercher, en premier, avec un sur-coût négligeable, les motifs de résultats intermédiaires déjà présents afin d'effectuer un remplacement. Nous apportons des éléments supplémentaires de réponses au sujet de la Q-résolution.

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 intéressant de spécifier un format pour des QBF sous forme clausale non prénexe, 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. Cela permettrait d'éviter la reconstruction en interne d'un arbre de quantificateurs. Dans un second temps, il serait intéressant de travailler directement sur la formulation originale, 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 plus facilement l'impact des heuristiques telle que la Q-résolution sur ces différentes catégories.

7. Bibliographie

Ayari A., Basin D. A., « Bounded Model Construction for Monadic Second-Order Logics », *in* E. A. Emerson, A. P. Sistla (eds), *CAV*, vol. 1855 of *Lecture Notes in Computer Science*, Springer, p. 99-112, 2000.

Ayari A., Basin D. A., « Qubos: Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers », *Formal Methods in Computer-Aided Design, Fourth International Conference, FMCAD 2002*, Springer-Verlag, 2002.

Ayari A., Basin D. A., Friedrich S., « Structural and Behavioral Modeling with Monadic Logics », *ISMVL*, p. 142-151, 1999.

Bacchus F., Walsh T. (eds), *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings*, vol. 3569 of *Lecture Notes in Computer Science*, Springer, 2005.

Benedetti M., « Quantifier Trees for QBFs », *in* Bacchus *et al.* (2005), p. 378-385, 2005a.

Benedetti M., « sKizzo: A Suite to Evaluate and Certify QBFs », *in* R. Nieuwenhuis (ed.), *CADE*, vol. 3632 of *Lecture Notes in Computer Science*, Springer, p. 369-376, 2005b.

Benedetti M., Mangassarian H., « QBF-Based Formal Verification: Experience and Perspectives », *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 5, p. 133-191, 2008.

Besnard P., Hunter A., Woltran S., « Encoding deductive argumentation in quantified Boolean formulae », *Artificial Intelligence*, vol. 173, n° 15, p. 1406 - 1423, 2009.

Biere A., « Resolve and Expand », *in* H. H. Hoos, D. G. Mitchell (eds), *SAT (Selected Papers)*, vol. 3542 of *Lecture Notes in Computer Science*, Springer, p. 59-70, 2004.

Bryant R. E., Lahiri S. K., Seshia S. A., « Convergence Testing in Term-Level Bounded Model Checking », *in* D. Geist, E. Tronci (eds), *CHARME*, vol. 2860 of *Lecture Notes in Computer Science*, Springer, p. 348-362, 2003.

Bubeck U., Büning H. K., « Bounded Universal Expansion for Preprocessing QBF », *in* Marques-Silva *et al.* (2007), p. 244-257, 2007.

Büning H. K., Karpinski M., Flögel A., « Resolution for Quantified Boolean Formulas », *Inf. Comput.*, vol. 117, n° 1, p. 12-18, 1995.

Da Mota B., Stéphan I., Nicolas P., « Une nouvelle stratégie de mise sous forme prénexe pour des formules booléennes quantifiées avec bi-implication », *Information-Interaction-Intelligence i3 2009 n°2*, 2010.

Davis M., Logemann G., Loveland D. W., « A machine program for theorem-proving », *Commun. ACM*, vol. 5, n° 7, p. 394-397, 1962.

de la Tour T. B., « An Optimality Result for Clause Form Translation », *J. Symb. Comput.*, vol. 14, n° 4, p. 283-302, 1992.

Demaine E. D., « Playing Games with Algorithms: Algorithmic Combinatorial Game Theory », *in* J. Sgall, A. Pultr, P. Kolman (eds), *MFCS*, vol. 2136 of *Lecture Notes in Computer Science*, Springer, p. 18-32, 2001.

Dershowitz N., Hanna Z., Katz J., « Bounded Model Checking with QBF », *in* Bacchus *et al.* (2005), p. 408-414, 2005.

Egly U., « On Different Structure-Preserving Translations to Normal Form », *J. Symb. Comput.*, vol. 22, n° 2, p. 121-142, 1996.

Egly U., Eiter T., Klotz V., Tompits H., Woltran S., « Computing Stable Models with Quantified Boolean Formulas: Some Experimental Results », *in* A. Provetti, T. C. Son (eds), *Answer Set Programming*, 2001a.

Egly U., Eiter T., Tompits H., Woltran S., « Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas », *AAAI/IAAI*, AAAI Press / The MIT Press, p. 417-422, 2000.

Egly U., Seidl M., Tompits H., Woltran S., Zolda M., « Comparing Different Prenexing Strategies for Quantified Boolean Formulas », *in* Giunchiglia *et al.* (2004b), p. 214-228, 2003.

Egly U., Seidl M., Woltran S., « A solver for QBFs in negation normal form », *Constraints*, vol. 14, n° 1, p. 38-79, 2009.

Egly U., Tompits H., « Proof-complexity results for nonmonotonic reasoning », *ACM Trans. Comput. Log.*, vol. 2, n° 3, p. 340-387, 2001b.

Gent I., Rowley A., Encoding Connect-4 using Quantified Boolean Formulae, Technical Report n° APES-68-2003, APES Research Group, July, 2003.

Giunchiglia E., Narizzano M., Tacchella A., « QuBE++: An Efficient QBF Solver », *in* A. J. Hu, A. K. Martin (eds), *FMCAD*, vol. 3312 of *Lecture Notes in Computer Science*, Springer, p. 201-213, 2004a.

Giunchiglia E., Narizzano M., Tacchella A., « Quantifier structure in search based procedures for QBFs », *Proceedings of the conference on Design, automation and test in Europe (DATE'06)*, p. 812-817, 2006.

Giunchiglia E., Tacchella A. (eds), *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, vol. 2919 of *Lecture Notes in Computer Science*, Springer, 2004b.

Ling A. C., Singh D. P., Brown S. D., « FPGA Logic Synthesis Using Quantified Boolean Satisfiability », *in* Bacchus *et al.* (2005), p. 444-450, 2005.

Lonsing F., Biere A., « Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers », *Electr. Notes Theor. Comput. Sci.*, vol. 251, p. 83-95, 2009.

Mangassarian H., Veneris A. G., Safarpour S., Benedetti M., Smith D., « A performance-driven QBF-based iterative logic array representation with applications to verification, debug and test », *in* G. G. E. Gielen (ed.), *ICCAD*, IEEE, p. 240-245, 2007.

Marques-Silva J., Sakallah K. A. (eds), *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, vol. 4501 of *Lecture Notes in Computer Science*, Springer, 2007.

Mneimneh M. N., Sakallah K. A., « Computing Vertex Eccentricity in Exponentially Large Graphs: QBF Formulation and Solution », *in* Giunchiglia *et al.* (2004b), p. 411-425, 2003.

Pan G., Vardi M. Y., « Symbolic Decision Procedures for QBF », *in* M. Wallace (ed.), *CP*, vol. 3258 of *Lecture Notes in Computer Science*, Springer, p. 453-467, 2004.

Plaisted D. A., Greenbaum S., « A Structure-Preserving Clause Form Translation », *Journal of Symbolic Computation*, vol. 2, n° 3, p. 293-304, 1986.

Rintanen J., « Constructing Conditional Plans by a Theorem-Prover », *J. Artif. Intell. Res. (JAIR)*, vol. 10, p. 323-352, 1999.

Scholl C., Becker B., « Checking Equivalence for Partial Implementations », *DAC*, ACM, p. 238-243, 2001.

Staber S., Bloem R., « Fault Localization and Correction with QBF », *in* Marques-Silva *et al.* (2007), p. 355-368, 2007.

Stockmeyer L., « The polynomial-time hierarchy », *Theoretical Computer Science*, vol. 3, p. 1-22, 1977.

Tseitin G. S., « On the complexity of derivation in propositional calculus », *in* A. O. Slisenko (ed.), *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, Consultants Bureau, New York, p. 115-125, 1970.