Base littérale et certificat pour les formules booléennes quantifiées

Igor Stéphan, Benoit da Mota

Base littérale et certificat pour les formules booléennes quantifiées

Igor Stéphan **Benoit Da Mota**

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

Résumé

Nous étudions dans cet article une nouvelle forme normale, celle des *bases littérales*, pour les formules booléennes quantifiées (prénexes). Un des aspects majeurs d'une base littérale est que le certificat (qui permet de vérifier a posteriori qu'une décision positive sur une formule booléenne quantifiée est correcte ou non) en est un cas particulier. Notre principal résultat est un algorithme orienté recherche qui calcule un certificat associé à une formule booléenne quantifiée dans le cas où elle est valide. Un autre aspect majeur d'une base littérale est qu'elle peut être vue comme étant le résultat d'une compilation de la formule booléenne quantifiée qui respecte des propriétés d'optimalité en terme de construction de la solution et de minimalité en terme de modèles propositionnels de la matrice.

Abstract

We study in this article a new normal form, the *literal bases*, for the (prenex) quantified Boolean formulae. One of the major aspects of literal bases is that the notion of certificate (which allows to verify a posteriori that validity of a quantified Boolean formula is correct or not) is one special case. Our main result is a top-down search algorithm which computes the certificate associated to a quantified Boolean formula in case of its validity. Another major aspect of literal bases is that a literal base may be seen as a the result of the compilation of the quantified Boolean formula which respects some properties of optimality (in term of generation of solutions) and minimality (in term of number of propositional models of the matrix).

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. Les QBF sont aussi utiles pour représenter des stratégies dans un jeu fini à deux joueurs. Dans un tel jeu les adversaires jouent alternativement en valuant les variables d'une formule qui doit être valide si un des deux joueurs est sûr de gagner. Dans ce style d'application, une procédure de décision n'est pas suffisante et une solution au problème de recherche associé à la QBF est nécessaire. De plus, lorsque l'algorithme de décision répond valide ou non-valide, il n'y a aucun moyen de vérifier que l'algorithme a correctement répondu : dans le cadre de la logique propositionnelle, le résultat associé à la décision positive est le modèle qui est alors facilement vérifiable. Une possibilité est de construire un arbre représentant la solution (appelé politique [6] ou stratégie [3]) mais il est alors dans le pire et la majeur partie des cas de taille exponentielle. Dans [1] une solution à ces deux problèmes est proposée : le **sat**-certificat qui est une représentation pour une séquence des fonctions booléennes qui valide la formule. Si la notion de **sat**-certificat est exprimée dans [1] indépendemment de tout algorithme, elle l'est dans le formalisme des diagrammes de décision binaire [4] pour des formules sous forme normale conjonctive. De plus son extraction n'est explicitée que dans le cadre du solver sKizzo [2] basé sur la skolémisation symbolique et le raisonnement symbolique ; or la plupart des implémentations [9] sont basées sur des algorithmes orientés recherche à la Davis, Logemann et Loveland [8][1]. Dans [12], un algorithme d'élimination des quantificateurs est proposé pour résoudre les

[1] Il est à noter que sKizzo permet aussi le calcul via un algorithme DLL et en extrait des **sat**-certificats

QBF (et non "seulement" décider de la validité). Cet algorithme génère une QBF équivalente non seulement au sens de la validité mais aussi par la préservation des solutions de la formule. Le résultat de l'application de l'algorithme peut être vu comme le compilé de la QBF initiale. Le format de cette QBF générée est très proche de celui des **sat**-certificats.

Après des préliminaires et un exemple introductif qui a pour but de révéler les intuitions, nous présentons les résultats suivants :

- une description de cette forme commune pour les **sat**-certificats et le compilé d'une QBF selon [12] : les bases littérales,
- une interprétation des **sat**-certificats en terme de QBF selon les bases littérales,
- des opérations internes ainsi que des propriétés sur bases littérales ;

le tout amenant

- à une extension de tout algorithme orienté recherche pour la construction des **sat**-certificats et
- à une extension de tout algorithme orienté recherche pour la compilation d'une QBF valide en une base littérale respectant un critère d'optimalité et dont l'interprétation sous forme de QBF est telle que la matrice respecte un critère de minimalité.

2 Préliminaires

Les formules booléennes quantifiées. Les valeurs booléennes sont notées **vrai** et **faux**. L'ensemble des symboles (ou variables) propositionnel(le)s est noté $\mathcal{V}$. Les symboles $\bot$ et $\top$ sont les constantes booléennes. Le symbole $\wedge$ est utilisé pour la conjonction, $\vee$ pour la disjonction, $\neg$ pour la négation, $\rightarrow$ pour l'implication et $\leftrightarrow$ pour la bi-implication. L'ensemble des formules propositionnelles est dénoté **PROP**. Un littéral est une variable booléenne ou la négation de celle-ci. Une formule booléenne est sous forme normale conjonctive (FNC) - resp. disjonctive (FND) - si c'est une conjonction de disjonctions de littéraux - resp. disjonction de conjonctions de littéraux ; la fonction fnc - resp. fnd - de **PROP** dans **PROP** calcule la FNC - resp. FND - d'une formule quelconque. Une substitution est une fonction de l'ensemble des variables dans l'ensemble **PROP**. Nous définissons la substitution d'une variable 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 propositionnelle x par la formule F. Une valuation v est une fonction de $\mathcal{V}$ dans **BOOL** et l'interprétation, notée v^* est son extension dans **PROP**. A une valuation v est associé une substitution $\vec{v}$ ainsi : si $v(x) = \mathbf{vrai}$ alors $[x \leftarrow \top]$ est dans $\vec{v}$, si $v(x) = \mathbf{faux}$ alors $[x \leftarrow \bot]$ est dans $\vec{v}$. La satisfaction propositionnelle est notée $\models$ ($v \models F$ signifie que $v^*(F) = \mathbf{vrai}$, la formule propositionnelle F est satisfaite par la valuation v et v est un modèle de F) et l'équivalence logique $\equiv$. Le symbole $\exists$ est utilisé pour la quantification existentielle et $\forall$ pour la quantification universelle (q est utilisé pour noter un quantificateur quelconque). Toute formule booléenne est aussi une formule booléenne quantifiée (QBF). Si F est une QBF et x est une variable booléenne alors $(\exists x\ F)$ et $(\forall x\ F)$ sont des QBF. Un lieur est une chaîne de caractères $q_1 x_1 \dots q_n x_n$ avec $x_1, \dots, x_n$ des variables distinctes et $q_1 \dots q_n$ des quantificateurs ; le lieur vide est dénoté ϵ ; par convention, des quantificateurs différents lient des variables différentes. Une formule booléenne quantifiée constituée d'un lieur et d'une formule booléenne appelée matrice est une QBF prénexe ; si toute variable apparaissant dans une QBF possède une occurrence dans le lieur elle est dite close. Nous nous restreignons par la suite aux QBF prénexes et closes. Une QBF est en FNC - resp. FND - si sa matrice l'est.

La sémantique des QBF. La sémantique des symboles booléens est définie de manière habituelle. La sémantique des quantificateurs est la suivante : pour toute variable booléenne y et toute QBF $qyQF$,

$$\exists y QF = (QF[y \leftarrow \top] \vee QF[y \leftarrow \bot])$$

et

$$\forall y QF = (QF[y \leftarrow \top] \wedge QF[y \leftarrow \bot]).$$

Une QBF est valide si $F \equiv \top$. Si y est une variable quantifiée existentiellement précédée par les variables quantifiées universellement $x_1, \dots, x_n$, nous notons $\hat{y}_{x_1 \dots x_n}$ sa fonction de Skolem de $\{\mathbf{vrai}, \mathbf{faux}\}^{\mathbf{n}}$ dans $\{\mathbf{vrai}, \mathbf{faux}\}$. Un modèle pour une QBF F est une séquence s de fonctions de Skolem qui satisfait la formule [5]. Par exemple, la QBF $\exists y \exists x \forall z((x \vee y) \leftrightarrow z)$ n'est pas valide tandis que $\forall z \exists y \exists x((x \vee y) \leftrightarrow z)$ l'est avec pour séquence possible de fonctions de Skolem : $\hat{y}_z(\mathbf{vrai}) = \mathbf{vrai}$, $\hat{y}_z(\mathbf{faux}) = \mathbf{faux}$, $\hat{x}_z(\mathbf{vrai}) = \mathbf{faux}$ et $\hat{x}_z(\mathbf{faux}) = \mathbf{faux}$. Un modèle (booléen) pour une formule booléenne non quantifiée (i.e. une formule propositionnelle) correspond exactement au modèle (QBF) de sa clôture existentielle. Une QBF est valide si et seulement s'il existe une séquence de fonctions de Skolem qui satisfasse la formule. Dans [12, 11], une nouvelle relation d'équivalence sur les QBF, notée $\cong$, a été introduite ; elle porte sur la préservation de l'ensemble des modèles (et non plus seulement sur la validité). Par exemple, $\forall z \exists y \exists x((x \vee y) \leftrightarrow z) \equiv \top$ mais $\forall z \exists y \exists x((x \vee y) \leftrightarrow z) \ncong \top$.

sat-certificat. A une fonction de Skolem $\hat{y}_{x_1 \dots x_n}$ faisant partie d'une séquence de fonctions de Sko-

lem qui satisfait une QBF, nous associons deux formules propositionnelles P_y and N_y définies uniquement sur les variables universellement quantifiées $\{x_1, \ldots, x_n\}$ ainsi : $v^*(P_y) =$ **vrai** si et seulement si $\hat{y}_{x_1 \ldots x_n}(v(x_1), \ldots, v(x_n)) =$ **vrai** et $v^*(N_y) =$ **vrai** si et seulement si $\hat{y}_{x_1 \ldots x_n}(v(x_1), \ldots, v(x_n)) =$ **faux** (v une valuation quelconque). A une séquence de fonctions de Skolem $\hat{y_1}; \ldots; \hat{y_m}$ satisfaisant une QBF F, $y_1, \ldots, y_m$ les variables existentiellement quantifiées, est associé un **sat**-certificat i.e. une séquence de paires de formules $(P_1, N_1); \ldots; (P_m, N_m)$. Cette définition est légèrement différente de celle de [1] : dans notre définition les formules P_i et N_i, $1 \leq i \leq m$, sont associées exactement aux fonctions de Skolem tandis que, dans [1], elles sont associées à des fonctions booléennes qui recouvrent des ensembles de fonctions de Skolem. Dans cet article, nous nous plaçons pour des raisons de clarté dans la plus grande généralité : un algorithme quelconque orienté recherche, sans présupposer de mécanismes de déduction qui rendraient certaines variables sans valuation à la fin du calcul.

3 Exemple introductif

Nous développons un exemple introductif pour soutenir l'intuition des algorithmes et résultats qui seront présentés dans les sections suivantes.

Soit la formule propositionnelle

$$F = ((c \vee b) \wedge (b \rightarrow ((c \rightarrow d) \wedge (c \vee (a \leftrightarrow \neg d)))))$$

Alors F peut être décomposée selon la séquence de variables $a; b; c; d$ en les formules :

$$\begin{array}{rl} F \equiv & (\neg a \vee \top) \wedge (a \vee \top) \wedge (\neg b \vee \top) \wedge (b \vee \top) \wedge \\ & (\neg c \vee \top) \wedge (c \vee \top) \wedge (\neg d \vee X^{\neg}) \wedge (d \vee X) \end{array}$$

ou bien

$$F \equiv \overbrace{(\neg a \vee \top) \wedge (a \vee \top)}^{1} \wedge \overbrace{(\neg b \vee \top) \wedge (b \vee \top)}^{2} \wedge \underbrace{(\neg c \vee \top)}_{3} \wedge \underbrace{(c \vee b)}_{4} \wedge \underbrace{(\neg d \vee X^{\neg})}_{5} \wedge \underbrace{(d \vee X)}_{6}$$

avec $X = (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (a \vee b \vee c)$

et $X^{\neg} = (\neg a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c) \wedge (a \vee b \vee c)$.

La première (élémentaire) a été obtenue par associativité directement à partir de la forme normale conjonctive[2] de F :

$$\begin{array}{l} fnc(F) = \\ \quad (\neg a \vee \neg b \vee \neg c \vee d) \wedge (\neg a \vee b \vee c \vee d) \wedge (a \vee \neg b \vee \neg c \vee d) \wedge \\ \quad (a \vee \neg b \vee c \vee d) \wedge (a \vee b \vee c \vee d) \wedge (\neg a \vee \neg b \vee c \vee \neg d) \wedge \\ \quad (\neg a \vee b \vee c \vee \neg d) \wedge (a \vee b \vee c \vee \neg d) \end{array}$$

[2]Ce n'est pas la forme normale la plus simplifiée.

La seconde est plus intéressante car elle exhibe les propriétés suivantes :

1. pour toute valeur de vérité de a, la formule admet (au moins) un modèle ;
2. pour toute valeur de vérité de b, la formule admet (au moins) un modèle ;
3. si c est interprété à **vrai**, la formule admet (au moins) un modèle ;
4. si c est interprété à **faux**, pour que la formule admette un modèle il faut que b soit **vrai** ;
5. si d est interprété à **vrai**, pour que la formule admette un modèle il faut nécessairement que $X^{\neg}$ soit **vrai**.
6. si d est interprété à **faux**, pour que la formule admette un modèle il faut nécessairement que X soit **vrai** ;

Cette seconde décomposition peut être obtenue à partir de la forme normale disjonctive[2] de F :

$$\begin{array}{l} fnd(F) = \\ \quad (a \wedge b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d) \vee (a \wedge \neg b \wedge c \wedge d) \vee \\ \quad (a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge b \wedge c \wedge d) \vee (\neg a \wedge b \wedge \neg c \wedge d) \vee \\ \quad (\neg a \wedge \neg b \wedge c \wedge d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d) \end{array}$$

et l'équivalence (spécialisation de la distributivité)

$$((\neg x \wedge A) \vee (x \wedge B)) \stackrel{dis}{\equiv} ((A \vee B) \wedge (\neg x \vee B) \wedge (x \vee A))$$

ainsi

$$\begin{array}{rll} fnd(F) & \stackrel{dis}{\equiv} & (Y \vee Y^{\neg}) \wedge (\neg d \vee Y^{\neg}) \wedge (d \vee Y) \\ & \stackrel{dis*}{\equiv} & (\neg a \vee \top) \wedge (a \vee \top) \wedge (\neg b \vee \top) \wedge (b \vee \top) \wedge \\ & & (\neg c \vee \top) \wedge (c \vee b) \wedge (\neg d \vee Y^{\neg}) \wedge (d \vee Y) \end{array}$$

avec $Y = (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge c) \vee (\neg a \wedge \neg b \wedge c)$ et

$$\begin{array}{rl} Y^{\neg} = & (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c) \vee \\ & (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \end{array}$$

Nous pouvons enfin retrouver la seconde décomposition en remarquant que $Y \equiv X$ et $Y^{\neg} \equiv X^{\neg}$.

Nous poursuivons cet exemple introductif en quantifiant la formule F ainsi : $F_{\forall a \exists b \forall c \exists d} = \forall a \exists b \forall c \exists d F$.

A nouveau

$$\begin{array}{l} F_{\forall a \exists b \forall c \exists d} \cong \forall a \exists b \forall c \exists d \\ \quad (\neg a \vee \top) \wedge (a \vee \top) \wedge (\neg b \vee \top) \wedge (b \vee \top) \wedge \\ \quad (\neg c \vee \top) \wedge (c \vee b) \wedge (\neg d \vee X^{\neg}) \wedge (d \vee X) \end{array}$$

Mais cette décomposition n'a plus la propriété intéressante exhibée pour SAT : il n'y a pas de fonction de Skolem $\hat{b}_a$ qui satisfasse la QBF telle que $\hat{b}_a(.) =$ **faux**. Mais une telle décomposition est possible :

$$F_{\forall a\exists b\forall c\exists d} \cong \forall a\exists b\forall c\exists d$$
$$\overbrace{(\neg a\vee\top)\wedge(a\vee\top)}^{1} \wedge \overbrace{(\neg b\vee\top)}^{2} \wedge \overbrace{(b\vee\bot)}^{3} \wedge \overbrace{(\neg c\vee\top)\wedge(c\vee\top)}^{4} \wedge$$
$$\underbrace{(\neg d\vee(a\wedge b\wedge c)\vee(\neg a\wedge b\wedge c)\vee(\neg a\wedge b\wedge\neg c))}_{5} \wedge \underbrace{(d\vee(a\wedge b\wedge\neg c))}_{6}$$

Les cas 1 et 4 sont immédiats puisque les variables a et c sont quantifiées universellement ;

2. & 3. Pour une valeur de vérité de a donnée, si $\hat{b}_a(a) =$ **faux** il n'existe pas de séquence de fonctions de Skolem qui satisfasse la QBF et si $\hat{b}_a(a) =$ **vrai** il existe (au moins) une séquence de fonctions de Skolem qui satisfasse la QBF ;
5. Pour des valeurs de vérité de a et c données, si $\hat{d}_{ac}(a,c) =$ **vrai**, pour que la QBF admette au moins une solution il faut nécessairement que $(a\wedge b\wedge c)\vee(\neg a\wedge b\wedge c)\vee(\neg a\wedge b\wedge\neg c)$ soit **vrai** ;
6. Pour des valeurs de vérité de a et c données, si $\hat{d}_{ac}(a,c) =$ **faux**, pour que la QBF admette au moins une solution il faut nécessairement que $(a\wedge b\wedge\neg c)$ soit **vrai** ;

De cette dernière décomposition s'obtient le certificat : $(\top,\bot);((a\wedge c)\vee(\neg a\wedge c)\vee(\neg a\wedge\neg c),(a\wedge\neg c))$ dans lequel nous retrouvons l'unique séquence $\hat{b}_a;\hat{d}_{ac}$ de fonctions de Skolem qui satisfasse la QBF avec :

$$\hat{b}_a(\mathbf{vrai}) = \mathbf{vrai}, \hat{\mathbf{b}}_{\mathbf{a}}(\mathbf{faux}) = \mathbf{vrai},$$
$$\hat{d}_{ac}(\mathbf{vrai},\mathbf{vrai}) = \mathbf{vrai}, \hat{\mathbf{d}}_{\mathbf{ac}}(\mathbf{vrai},\mathbf{faux}) = \mathbf{faux},$$
$$\hat{d}_{ac}(\mathbf{faux},\mathbf{vrai}) = \mathbf{vrai}, \hat{\mathbf{d}}_{\mathbf{ac}}(\mathbf{faux},\mathbf{faux}) = \mathbf{vrai}.$$

4 Base littérale

Les formes normales disjonctive et conjonctive, aussi bien pour SAT que pour QBF, suivent l'aspect horizontal d'une table de vérité (dont les valuations des symboles propositionnels forment les colonnes) : elles reprennent syntaxiquement respectivement la disjonction des modèles et la négation de la disjonction des valuations qui falsifient la formule. Si dans SAT, les formes normales disjonctive et conjonctive reflètent exactement l'ensemble des modèles, il n'en est pas de même dans QBF : des conjonctions de littéraux (resp. disjonctions de littéraux) peuvent faire parti de la forme normale disjonctive (respectivement conjonctive) de la matrice sans pour autant faire parti d'une solution à la QBF. Les formes décrites dans [1] (le **sat**-certificat pour la validité d'une QBF) et [12] (le calcul d'une QBF équivalente conservant toutes les solutions) sont dans l'aspect vertical de la table de vérité (et donc selon une séquence pour les variables). En nous inspirant de ces formes, nous définissons la notion de base littérale pour les QBF, nous en donnons l'interprétation selon les QBF puis nous définissons trois lois internes. Deux d'entre-elles transposent au niveau des bases littérales la disjonction et la conjonction, la troisième introduit dans les fonctions booléennes un nouvel argument. Nous introduisons aussi une propriété d'optimalité qui reflète la propriété exhibée dans l'exemple introductif et une propriété de minimalité pour les QBF ; le lien entre optimalité et minimalité sera explicité. Nous montrons alors comment dans le cadre de SAT (i.e. toutes les variables sont existentiellement quantifiées) une base littérale optimale peut être calculée, ceci en prélude aux algorithmes pour les QBF de la section suivante.

4.1 Base littérale et opérations

Nous explorons dans cette partie une forme normale, que nous appelons *base littérale*, qui suit l'aspect vertical de la table de vérité.

Définition 1 (Base littérale) *Une base littérale est un couple (Q,G) constitué*
- *soit de $Q=\epsilon$ et $G=\top$ ou $G=\bot$;*
- *soit d'un lieur $Q=q_1x_1\dots q_nx_n$, $n>0$, et d'une séquence de couples de formules (que nous nommerons les gardes) $G=(P_1,N_1);\dots;(P_n,N_n)$ telle que les formules P_k et N_k sont soit $\top$ ou $\bot$ soit uniquement construites sur les variables $\{x_1,\dots,x_{k-1}\}$.*

Nous notons $\mathcal{B}_Q$ l'ensemble des bases littérales pour un lieur Q et définissons la fonction grds qui extrait les gardes de la base littérale et qui est telle que $grds((Q,G))=G$.

De par la définition précédente :
- si $Q=\epsilon$ alors $\mathcal{B}_\epsilon=\{(\epsilon,\top),(\epsilon,\bot)\}$;
- si $Q=qx$ alors
$$\mathcal{B}_{qx}=\{(qx,(\top,\top)),(qx,(\top,\bot)),(qx,(\bot,\top)),(qx,(\bot,\bot))\}$$

Définition 2 (Interprétation d'une base littérale) *L'interprétation d'une base littérale B, fonction à valeur dans l'ensemble des QBF et notée B^*, est définie par :*
- *si $B=(\epsilon,F)$ alors $B^*=F$;*
- *si $B=(q_1x_1\dots q_nx_n,(P_1,N_1);\dots;(P_n,N_n))$, $n>0$, alors*
$$B^* = q_1x_1\dots q_nx_n \bigwedge_{k\leq n}((\neg x_k\vee P_k)\wedge(x_k\vee N_k))$$

Théorème 1 (Complétude des bases littérales) *Soit QF une QBF alors il existe une base littérale $B\in\mathcal{B}_Q$ telle que $B^*\cong QF$.*

Argument de la preuve : Ce théorème est immédiat car toute formule est équivalente à une formule sous forme normale conjonctive dont la dernière variable de la séquence peut être mise en facteur. □

Exemple 1 (Suite de l'exemple introductif) *Soit* $Q = \exists a\exists b\exists c\exists d$ *alors* $B = (Q, (\top,\top);(\top,\top);(\top,\top);(X^{\neg},X))$ *et* $B' = (Q, (\top,\top);(\top,\top);(\top,b);(Y^{\neg},Y))$ *sont telles que* $B, B' \in \mathcal{B}_Q$ *et* $B^* \cong \exists a\exists b\exists c\exists dF$ *et* $B'^* \cong \exists a\exists b\exists c\exists dF$.

Nous transcrivons la conjonction et la disjonction dans le formalisme des bases littérales en deux lois de composition interne respectivement $\otimes$ et $\oplus$. L'interprétation des bases littérales ayant pour colonne vertébrale la conjonction, l'opérateur $\otimes$ est immédiat ; par contre l'opérateur $\oplus$ est plus complexe puisqu'il intègre la distributivité. Nous ajoutons un opérateur indicé par une variable x qui est un opérateur pour combiner les **sat**-certificats et introduire dans les fonctions booléennes le nouvel argument x : cet opérateur est directement inspiré de la manière dont on démontre que le fragment propositionnel constitué uniquement sur la conjonction, la disjonction et la négation est complet.

Définition 3 (Opérateurs $\otimes$, $\oplus$ et $\circ$) *Soit un lieur* $Q = q_1x_1 \dots q_nx_n$ *et* $B, B' \in \mathcal{B}_Q$. *Les opérateurs* $\otimes, \oplus : \mathcal{B}_Q \times \mathcal{B}_Q \to \mathcal{B}_Q$ *sont définis par :*

$\otimes$ *:*
$$\begin{array}{l}(Q,(P_1,N_1);\dots;(P_n,N_n))\otimes\\ (Q,(P'_1,N'_1);\dots;(P'_n,N'_n))\\ =\ (Q,\ ((P_1\wedge P'_1),(N_1\wedge N'_1));\dots;\\ \qquad ((P_n\wedge P'_n),(N_n\wedge N'_n))\end{array}$$

$\oplus$ *: si* $Q = \epsilon$ *alors* $(B \oplus B') = (B \vee B')$ *sinon*

$$\begin{array}{l}(Q,(P_1,N_1);\dots;(P_n,N_n))\oplus\\ (Q,(P'_1,N'_1);\dots;(P'_n,N'_n))\\ =\ (Q,\ ((P_1\vee P'_1),(N_1\vee N'_1));\\ \qquad (\mathcal{P}_2\wedge(P'_2\vee\mathcal{X})\wedge(P_2\vee\mathcal{X}'),\\ \qquad \mathcal{N}_2\wedge(N'_2\vee\mathcal{X})\wedge(N_2\vee\mathcal{X}'));\dots;\\ \qquad (\mathcal{P}_n\wedge(P'_n\vee\mathcal{X})\wedge(P_n\vee\mathcal{X}'),\\ \qquad \mathcal{N}_n\wedge(N'_n\vee\mathcal{X})\wedge(N_n\vee\mathcal{X}')))\end{array}$$

avec $\mathcal{X} = ((\neg x_1 \vee P_1)\wedge(x_1\vee N_1))$,
$\mathcal{X}' = ((\neg x_1 \vee P'_1)\wedge(x_1\vee N'_1))$
et avec $Q' = q_2x_2 \dots q_nx_n$

$$\begin{array}{l}(Q',(P_2,N_2);\dots;(P_n,N_n))\oplus\\ (Q',(P'_2,N'_2);\dots;(P'_n,N'_n)) =\\ \quad (Q',(\mathcal{P}_2,\mathcal{N}_2);\dots;(\mathcal{P}_n,\mathcal{N}_n))\end{array}$$

L'opérateur $\circ_x : \mathcal{B}_Q \times \mathcal{B}_Q \to \mathcal{B}_{\forall xQ}$ *défini par :*

$$\begin{array}{l}(Q,(P_1,N_1);\dots;(P_n,N_n))\circ_x\\ (Q,(P'_1,N'_1);\dots;(P'_n,N'_n))\\ =(\ \forall xQ,\\ \quad (\top,\top);\\ \quad (((\neg x\vee P_1)\wedge(x\vee P'_1)),((\neg x\vee N_1)\wedge(x\vee N'_1)));\dots;\\ \quad (((\neg x\vee P_n)\wedge(x\vee P'_n)),((\neg x\vee N_n)\wedge(x\vee N'_n))))\end{array}$$

Dans la définition précédente, lorsque $n = 1$,

$$\begin{array}{l}(q_1x_1,(P_1,N_1))\oplus(q_1x_1,(P_1^{\neg},N_1^{\neg}))\\ =\ (q_1x_1,((P_1\vee P_1^{\neg}),(N_1\vee N_1^{\neg})))\end{array}$$

ce qui définit le cas d'arrêt de l'opération $\oplus$.

Il est important de remarquer que l'opérateur $\circ$ n'est pas commutatif.

Exemple 2 (Suite de l'exemple introductif) *Soient* $Q = \forall a\exists b\forall c\exists d$ *et les bases littérales*

$$\begin{array}{l}B_1 = (Q,\ (\bot,\top);(\neg a,\neg a);(\neg a,(\neg a\wedge b));\\ \qquad (\neg a\wedge(b\vee c),\neg a\wedge\neg b\wedge c))\end{array}$$

et

$$\begin{array}{l}B_2 = (Q,\ (\top,\bot);(a,a);(a,(a\wedge b));\\ \qquad ((a\wedge c)),(a\wedge(b\leftrightarrow\neg c)))\end{array}$$

Après simplifications algébriques

$$(B_1 \oplus B_2) = (Q,(\top,\top);(\top,\top);(\top,b);(Y^{\neg},Y))$$

Soient $B_3 = (\exists b\forall c\exists d,(\top,\bot);(\top,\top);(c,\neg c))$ *et* $B_4 = (\exists b\forall c\exists d,(\top,\bot);(\top,\top);(\top,\bot))$ *alors*

$$\begin{array}{c}(B_3 \circ_a B_4) = (Q,(\top,\top);(\top,\bot);(\top,\top);\\ ([(a\wedge c)\vee(\neg a\wedge c)\vee(\neg a\wedge\neg c)],(a\wedge\neg c)))\end{array}$$

Le théorème suivant confirme la sémantique des opérateurs $\oplus$ et $\otimes$.

Théorème 2 *Soient* Q *un lieur et* $B, B' \in \mathcal{B}_Q$ *telles que* $B^* = QF$ *et* $B'^* = QF'$ *alors* $(B \otimes B')^* = QF_{\otimes}$ *avec* $F_{\otimes} \equiv (F \wedge F')$ *et* $(B \oplus B')^* = QF_{\oplus}$ *avec* $F_{\oplus} \equiv (F \vee F')$.

Argument de la preuve : La première équivalence est immédiate par définition ; la seconde se démontre par récurrence grâce à la distributivité. □

Le théorème suivant nous permettra dans la section suivante de reconstruire des certificats pour des algorithmes orientés recherche puisqu'il recombine dans le cas d'un quantificateur universel les certificats obtenus par la substitution dans la formule de la variable universelle par $\top$ ou $\bot$. Dans la définition du **sat**-certificat n'est fait mention ni du lieur, ni de gardes pour les variables universelles ; nous définissons donc la fonction *certificat* qui extrait d'une base littérale uniquement les gardes pour les variables existentielles.

Théorème 3 *Soit* $\forall xQF$ *une QBF. Si* $certificat(B_\top)$ *est un* ***sat****-certificat pour* $QF[x \leftarrow \top]$ *et* $certificat(B_\bot)$ *est un* ***sat****-certificat pour* $QF[x \leftarrow \bot]$ *alors* $certificat(B_\top \circ_x B_\bot)$ *est un* ***sat****-certificat pour* $\forall xQF$.

Argument de la preuve : Le théorème se démontre en remarquant que dans $(B_\top \circ_x B_\bot)$ *si* x *est substitué par* $\top$ *le* ***sat****-certificat de* $B_\top$ *est retrouvé et que si* x *est substitué par* $\bot$ *le* ***sat****-certificat de* $B_\bot$ *est retrouvé.* □

4.2 Optimalité et minimalité

A une QBF correspond un ensemble de bases littérales. Dans le cas de SAT, ce qui rend plus intéressante la base littérale construite grâce à la forme normale disjonctive par rapport à celle obtenue grâce à la forme normale conjonctive est sa plus grande efficacité à détecter au plus tôt les valuations qui ne conduisent pas à un modèle ; nous définissons la propriété d'optimalité pour caractériser ces bases littérales.

Définition 4 (Optimalité d'une base littérale) *Soit $q_1x_1 \ldots q_nx_nF$ une QBF valide et une base littérale $B = (q_1x_1 \ldots q_nx_n, (P_1, N_1); \ldots; (P_n, N_n))$ telle que $B^* \cong q_1x_1 \ldots q_nx_nF$. La base littérale B est optimale (pour $q_1x_1 \ldots q_nx_nF$) si pour tout i, $1 \leq i \leq n$, v une valuation pour les variables de $x_1 \ldots x_{i-1}$.*

- *si $v \models P_i$ alors il existe (au moins) une séquence de fonctions de Skolem qui satisfasse la QBF $q_{i+1}x_{i+1} \ldots q_nx_n\vec{v}(F)[x_i \leftarrow \top]$;*
- *si $v \models N_i$ alors il existe (au moins) une séquence de fonctions de Skolem qui satisfasse la QBF $q_{i+1}x_{i+1} \ldots q_nx_n\vec{v}(F)[x_i \leftarrow \bot]$.*

Cette propriété d'optimalité est en lien avec la propriété de minimalité qui exprime que la matrice de la QBF ne contient que les modèles nécessaires à la construction de toutes les solutions à la QBF.

Définition 5 (Minimalité d'un QBF) *Une QBF est minimale si tout modèle de la matrice fait partie d'une séquence de fonctions de Skolem qui la satisfasse.*

Ce qui rend particulièrement intéressante la propriété d'optimalité d'une base littérale est exprimé dans le théorème suivant.

Théorème 4 *Soit B une base littérale optimale alors B^* est une QBF minimale.*

Argument de la preuve : Si l'interprétation de la base littérale n'est pas minimale alors il existe un modèle pour la matrice de l'interprétation qui ne fait pas partie d'une séquence de fonctions de Skolem qui satisfait la formule donc nécessairement il existe une séquence de gardes satisfaites par ce modèle mais au moins une de ces gardes n'aurait pas dû l'être puisque l'interprétation de la base littérale n'est pas minimale donc la base littérale n'est pas optimale. □

Dans le cas de la base littérale exprimant un **sat**-certificat les propriétés d'optimalité et de minimalité sont aisément obtenues.

La réciproque du théorème 4 est fausse comme le démontre l'exemple suivant.

Exemple 3 *La base littérale suivante*

$$(\forall a \exists b \forall c \exists d, (\top, \top); (\top, \bot); (\top, \top); \\ ((a \wedge b \wedge c) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c), (a \wedge b \wedge \neg c)))$$

issue de l'exemple introductif est optimale et son interprétation est bien minimale tandis que la base littérale suivante toujours issue du même exemple introductif n'est pas optimale mais son interprétation est minimale :

$$(\forall a \exists b \forall c \exists d, (\top, \top); (\top, \top); (\top, \top); \\ ((a \wedge b \wedge c) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c), (a \wedge b \wedge \neg c)))$$

4.3 Mise sous forme de base littérale

Il est possible, dans le cas SAT, d'obtenir à partir de la forme normale disjonctive une base littérale qui ait les propriétés mises en exergue dans l'exemple introductif.

Définition 6 (Fonction msf_bl) *Soit la fonction γ qui associe à un littéral positif la constante $\bot$ et à un littéral négatif la constante $\top$. Soit la fonction Γ construit à partir d'un lieur $\exists x_1 \ldots \exists x_n$ et d'une conjonction constituée de littéraux sur ces variables $(l_1, \ldots, l_n)$ une base littérale ainsi :*

$$\Gamma(\exists x_1 \ldots \exists x_n, l_1 \wedge \ldots \wedge l_n) = \\ (\exists x_1 \ldots \exists x_n, (\gamma(\overline{l_1}), \gamma(l_1)); \ldots; (\gamma(\overline{l_n}), \gamma(l_n)))$$

Nous étendons alors cette fonction à une forme normale disjonctive grâce à la loi interne $\oplus$ ($1 \leq i \leq n, 1 \leq j \leq m, l_{ij}$ des littéraux sur les variables x_i) :

$$msf\_bl(\exists x_1 \ldots \exists x_n, \bigvee_{1 \leq j \leq m} \bigwedge_{1 \leq i \leq n} l_{ij}) = \\ \bigoplus_{1 \leq j \leq m} \Gamma(\exists x_1 \ldots \exists x_n, \bigwedge_{1 \leq i \leq n} l_{ij})$$

Exemple 4 (Suite de l'exemple 2) *Nous rappelons que*

$$fnd(F) = \\ (a \wedge b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d) \vee (a \wedge \neg b \wedge c \wedge d) \vee \\ (a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge b \wedge c \wedge d) \vee (\neg a \wedge b \wedge \neg c \wedge d) \vee \\ (\neg a \wedge \neg b \wedge c \wedge d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d)$$

Nous réalisons le calcul en deux parties :

$$(\Gamma(Q, a \wedge b \wedge c \wedge d) \oplus \Gamma(Q, a \wedge b \wedge \neg c \wedge \neg d)) \oplus \\ (\Gamma(Q, a \wedge \neg b \wedge c \wedge d) \oplus \Gamma(Q, a \wedge \neg b \wedge c \wedge \neg d)) = \\ (Q, \quad (\bot, \top); (\neg a, \neg a); (\neg a, (\neg a \wedge b)); \\ (\neg a \wedge (b \vee c), \neg a \wedge \neg b \wedge c))$$

et

$$(\Gamma(Q, \neg a \wedge b \wedge c \wedge d) \oplus \Gamma(Q, \neg a \wedge b \wedge \neg c \wedge d)) \oplus \\ (\Gamma(Q, \neg a \wedge \neg b \wedge c \wedge d) \oplus \Gamma(Q, \neg a \wedge \neg b \wedge c \wedge \neg d)) = \\ (Q, \quad (\top, \bot); (a, a); \\ (a, (a \wedge b)); ((a \wedge c), (a \wedge (b \leftrightarrow \neg c))))$$

Nous reconnaissons alors les bases littérales B et B' de l'exemple 2 donc

$$\bigoplus_{(\bigwedge_{k\leq n} l_k)\in fnd(F)} \Gamma(Q, \bigwedge_{k\leq n} l_k) = (B \oplus B')$$

et nous avons bien

$$\begin{array}{ll} & \exists a\exists b\exists c\exists d F \cong \exists a\exists b\exists c\exists d\ fnd(F) \\ \cong & (\bigoplus_{(\bigwedge_{k\leq n} l_k)\in fnd(F)} \Gamma(Q, \bigwedge_{k\leq n} l_k))^* \\ = & msf\_bl(\exists a\exists b\exists c\exists d, fnd(F)) \end{array}$$

Tirant parti de l'associativité et de la commutativité de l'opération $\oplus$ (héritées de la disjonction par le théorème 2) et surtout du cas particulier où tous les quantificateurs sont existentiels, la fonction msf_bl est suffisante pour SAT.

Théorème 5 (Correction et optimalité) *Soit F une formule propositionnelle sur un ensemble de variables $\{x_1, \ldots, x_n\}$ et la base littérale $B = msf\_bl(\exists x_1 \ldots \exists x_n, fnd(F))$ alors $B^* \cong \exists x_1 \ldots \exists x_n F$ et si la formule F est satisfiable alors la base littérale B est optimale pour la QBF $\exists x_1 \ldots \exists x_n F$.*

La démonstration de ce théorème est triviale grâce au théorème 2.

La fonction msf_bl n'est pas suffisante pour les QBF en général qui demande le respect de l'ordre du lieur. Nous présentons donc un algorithme récursif *search_bl_sat* ayant les mêmes propriétés pour le cas de SAT que la fonction msf_bl mais qui s'étendra plus aisément dans la section prochaine aux QBF quelconques. Le cœur de cet algorithme est orienté recherche à la Davis, Logemann et Loveland [8].

Le théorème suivant montre que l'algorithme *search_bl_sat* réalise en fait une mise sous forme normale disjonctive.

Théorème 6 ($msf\_bl = search\_bl\_sat$) *Soit F une formule propositionnelle sur un ensemble de variables $\{x_1, \ldots, x_n\}$.*

$$msf\_bl(\exists x_1 \ldots \exists x_n, fnd(F)) = search\_bl\_sat(\exists x_1 \ldots \exists x_n, F).$$

La démonstration par induction de ce théorème est directe.

5 Bases littérales et certificats QBF pour les algorithmes orientés recherche

Dans cette section, nous présentons nos principaux résultats sur les algorithmes orientés recherche pour les QBF :

Algorithme 1 *search_bl_sat*

Entrée: Q : un lieur purement existentiel
Entrée: F : une formule propositionnelle sur Q
Sortie: une base littérale

```
si Q = ∃x alors
 selon F faire
  cas ⊤ : retourner (∃x, (⊤, ⊤))
  cas ⊥ : retourner ⊥
  cas x : retourner (∃x, (⊤, ⊥))
  cas ¬x : retourner (∃x, (⊥, ⊤))
 fin selon
sinon
 Q = ∃xQ'
 bl+ := search_bl_sat(Q', F[x ← ⊤])
 bl− := search_bl_sat(Q', F[x ← ⊥])
 si bl+ = ⊥ et bl− = ⊥ alors retourner ⊥ fin si
 si bl+ = ⊥ alors
 retourner (Q, (⊥, ⊤) ; grds(bl−)) fin si
 si bl− = ⊥ alors
 retourner (Q, (⊤, ⊥) ; grds(bl+)) fin si
 retourner  (Q, (⊤, ⊥) ; grds(bl+))⊕
            (Q, (⊥, ⊤) ; grds(bl−))
fin si
```

- l'extension à tout algorithme orienté recherche du calcul d'un **sat**-certificat pour une QBF ;
- l'extension à tout algorithme orienté recherche du calcul d'une base littérale dont l'interprétation est équivalente (au sens de la conservation des fonctions de Skolem) à la QBF.

5.1 Certificats QBF pour les algorithmes orientés recherche

Nous présentons l'algorithme $search\_certif\_qbf$ qui calcule un **sat**-certificat pour une QBF selon un algorithme orienté recherche.

L'algorithme $search\_certif\_qbf$ détermine d'abord si le lieur est réduit à un unique quantificateur. Dans le cas où il n'y a qu'un seul quantificateur, s'il est existentiel quatre cas sont possibles, correspondant dans l'ordre de l'algorithme à : $\exists x \top \equiv \exists x x$, $\exists x \bot \equiv \bot$, $\exists x x \cong \exists((\neg x \vee \top) \wedge (x \vee \bot))$ et $\exists x \neg x \cong ((\neg x \vee \bot) \wedge (x \vee \top))$; si le quantificateur est universel alors si $F \equiv \top$ alors $\forall x F \equiv \top$ sinon $\forall x x \equiv \forall x \neg x \equiv \forall x \bot \equiv \bot$. Dans le cas où il n'y a pas qu'un seul quantificateur, étant dans un algorithme orienté recherche, le premier est considéré[3]. Si le quantificateur est existentiel alors il suffit qu'un des deux appels récursifs pour la substitution par $\top$ (resp. $\bot$) ne soit pas à $\bot$ pour retourner un **sat**-certificat $(Q, (\top, \bot); grds(bl^+))$ (resp.

[3] plus précisément n'importe quelle variable du premier bloc de quantificateurs identiques

Algorithme 2 *search_certif_qbf*

Entrée: Q : le lieur d'une QBF
Entrée: F : la matrice d'une QBF
Sortie: un **sat**-certificat si la QBF est valide et $\bot$ sinon

```
si Q = qx alors
 si q = ∃ alors
  selon F faire
   cas ⊤ : retourner (∃x, (⊤, ⊥))
   cas ⊥ : retourner ⊥
   cas x : retourner (∃x, (⊤, ⊥))
   cas ¬x : retourner (∃x, (⊥, ⊤))
  fin selon
 sinon si F≡⊤ alors
  retourner (∀x, (⊤, ⊤))
 sinon
  retourner ⊥
 fin si
sinon
 Q = qxQ'
 bl+ := search_certif_qbf(Q', F[x ← ⊤])
 si bl+ = ⊥ alors
  si q = ∃ alors
   bl− := search_certif_qbf(Q', F[x ← ⊥])
   si bl− = ⊥ alors
    retourner ⊥
   sinon
    retourner (Q, (⊥, ⊤) ; grds(bl−))
   fin si
  sinon
   retourner ⊥
  fin si
 sinon
  si q = ∃ alors
   retourner (Q, (⊤, ⊥) ; grds(bl+))
  sinon
   bl− := search_certif_qbf(Q', F[x ← ⊥])
   si bl− = ⊥ alors
    retourner ⊥
   sinon
    retourner (bl+ ∘x bl−)
   fin si
  fin si
 fin si
fin si
```

$(Q, (\bot, \top); grds(bl^-)))$ qui exprime que x doit être nécessairement à **vrai** (resp. à **faux**). Si le quantificateur est universel alors si le résultat d'un des deux appels récursifs est $\bot$ c'est qu'il n'y a pas de solution et le résultat retourné est $\bot$ sinon les fonctions de Skolem exprimées dans les **sat**-certificats bl^+ et bl^- doivent être recombinées pour intégrer le nouvel argument x par $(bl^+ \circ_x\ bl^-)$ avant que ceci soit retourné comme **sat**-certificat.

Théorème 7 (Correction *search_certif_qbf***)** *Pour toute QBF QF, $search\_certif\_qbf(Q, F)$ retourne le **sat**-certificat si la QBF est valide et $\bot$ sinon.*

Argument de la preuve : Il est immédiat que l'algorithme retourne une base littérale différente de $\bot$ si la QBF est valide et $\bot$ si ce n'est pas le cas. De même, de part l'interprétation d'une base littérale, le cas d'induction pour le quantificateur existentiel est immédiat. Pour le cas d'induction pour le quantificateur universel, c'est le théorème 3 qui est appliqué. □

Dans [1], un algorithme pour la vérification d'un couple (QBF, **sat**-certificat), sa complexité (coNP-complète) et sa correction sont explicités.

5.2 Bases littérales pour les algorithmes orienté recherche

Nous présentons l'algorithme *search_bl_qbf* qui calcule la base littérale optimale pour des algorithmes orientés recherche dont l'interprétation est équivalente (au sens de la conservation des fonctions de Skolem) à la QBF. Le calcul d'une base littérale par cet algorithme peut être vu comme un processus de compilation. Nous présentons cet algorithme comme une extension directe de l'algorithme *search_sat_qbf*.

Le théorème suivant exprime que l'algorithme *search_bl_qbf* calcule à partir d'une QBF une base littérale dont l'interprétation lui est équivalente au sens de la conservation des modèles.

Théorème 8 (Correction de *search_bl_qbf***)** *Soit QF une QBF et la base littérale $B = search\_bl\_qbf(Q, F)$ alors $B^* \cong QF$.*

Argument de la preuve : Soient $bl^+)^* = Q'F^+$, $bl^-)^* = Q'F^-$, $(Q, (\top, \bot)\ ;\ grds(bl^+))^* = QF_+$,$(Q, (\bot, \top)\ ;\ grds(bl^-))^* = QF_-$ et $QF_\oplus = ((Q, (\top, \bot)\ ;\ grds(bl^+)) \oplus (Q, (\bot, \top)\ ;\ grds(bl^-)))^*$ Par hypothèse d'induction, $(bl^+)^* \cong Q'F[x \leftarrow \top]$ et $(bl^-)^* \cong Q'F[x \leftarrow \bot]$. Par le théorème 2, $F_\oplus \equiv (F_+ \vee F_-)$ et comme $F_+ = (\neg x \vee \top) \wedge (x \vee \bot) \wedge F^+$ et $F_- = (\neg x \vee \bot) \wedge (x \vee \top) \wedge F^-$, $F_\oplus[x \leftarrow \top] \equiv F^+$ et $F_\oplus[x \leftarrow \bot] \equiv F^-$. Donc $Q'F_\oplus[x \leftarrow \top] \cong Q'F[x \leftarrow \top]$ et $Q'F_\oplus[x \leftarrow \bot] \cong Q'F[x \leftarrow \bot]$. Donc

$((Q,(\top,\bot)\ ;\ grds(bl^+)) \oplus (Q,(\bot,\top)\ ;\ grds(bl^-)))^* \cong QF$ □

Le théorème suivant exprime que l'algorithme *search_bl_qbf* calcule à partir d'une QBF une base littérale optimale pour la QBF.

Théorème 9 (Optimalité de *search_bl_qbf*) *Soit QF une QBF et la base littérale $B = search\_bl\_qbf(Q,F)$ alors la base littérale B est optimale pour la formule QF.*

Ce théorème est immédiat par construction.

Algorithme 3 *search_bl_qbf*

Entrée: Q : le lieur d'une QBF
Entrée: F : la matrice d'une QBF
Sortie: une base littérale

```
si Q = qx alors
 si q = ∃ alors
  selon F faire
   cas ⊤ : retourner (∃x,(⊤,⊤))
   cas ⊥ : retourner ⊥
   cas x : retourner (∃x,(⊤,⊥))
   cas ¬x : retourner (∃x,(⊥,⊤))
  fin selon
 sinon
  si F = ⊤ alors retourner (∀x,(⊤,⊤))
  sinon retourner ⊥ fin si
 fin si
sinon
 Q = qxQ'
 bl+ := search_bl_qbf(Q',F[x ← ⊤])
 bl- := search_bl_qbf(Q',F[x ← ⊥])
 si q = ∃ alors
  si bl+ = ⊥ et bl- = ⊥
  alors retourner ⊥ fin si
  si bl+ = ⊥
  alors retourner (Q,(⊥,⊤) ; grds(bl-)) fin si
  si bl- = ⊥
  alors retourner (Q,(⊤,⊥) ; grds(bl+)) fin si
  retourner  (Q,(⊤,⊥) ; grds(bl+))⊕
             (Q,(⊥,⊤) ; grds(bl-))
 sinon
  si bl+ = ⊥ ou bl- = ⊥ alors
   retourner ⊥
  sinon
   retourner  (Q,(⊤,⊥) ; grds(bl+))⊕
              (Q,(⊥,⊤) ; grds(bl-))
  fin si
 fin si
fin si
```

6 Résultats expérimentaux

Les algorithmes décrits dans les sections précédentes ont été développés en Prolog pour s'assurer (en plus des preuves) de leur fonctionnement et sont accessibles sur notre site web. Seul l'algorithme *search_certif_qbf* a été intégré dans une application développée en C/C++. Pour nos tests, nous avons utilisé, pour algorithme orienté recherche à étendre, un algorithme de propagation booléenne quantifiée [13] dont l'implémentation est en C/C++ et dont nous avons ôté toutes les optimisations de la recherche qui modifient l'ordre des variables (nous n'avons conservé que la monotonie). Les **sat**-certificats ont été implémentés grâce aux diagrammes de décision binaires [4] (BDD). Les BDD sont utilisés pour représenter de façon compacte et explicite des fonctions booléennes. La librairie CUDD [10] permet entre autre la manipulation de BDD et offre un large assortiment d'opérations sur les BDD et de méthodes de réordonnancement des variables. Les BDD utilisés sont réduits et ordonnés (ROBDD), ainsi toutes les fonctions booléennes représentées possède le même ordre sur les variables (BDD ordonnés) et deux BDD représentant la même fonction possède une unique représentation canonique (BDD réduits). L'algorithme étendu étant pour des formules non-CNF, nous avons testé sur la série de benchmarks au format $QBF\_1.0$ de la compétition QBFEVAL'08[4]. Malheureusement, cette série de benchmarks n'a pour alternance de quantificateurs qu'uniquement $\exists\forall$ et le **sat**-certificat s'apparente alors à un modèle pour SAT. Nous rapportons dans le tableau ci-dessous les résultats les plus significatifs des temps en secondes pour le test de validité uniquement, pour le test de validité plus la construction du **sat**-certificat ainsi que le sur-coût en pourcentage ($\Delta\%$) ; pour les autres benchmarks résolus, le temps de calcul est trop négligeable pour que les différences soient significatives.

Benchmark	recherche	recherche+ certificat	$\Delta\%$
counter4_16	656.950s	691.942s	5.3%
ring4_5	206.846s	208.461s	0.8%
ring5_4	60.262s	61.774s	2.5%
ring6_4	42.108s	46.137s	9.5%
semaphore4_4	16.501s	18.522s	12.2%

Le benchmark counter4_16 est particulièrement intéressant car il est le seul à être non valide : dans ce cas les calculs liés à la construction du **sat**-certificat se sont révélés inutiles et le sur-coût est de 5.3%.

[4] `http ://www.qbflib.org/`

7 Conclusion

Cette étude répond au soucis de mieux comprendre les certificats pour les QBF et de les intégrer dans les algorithmes orientés recherche. Nous avons décrit quelques résultats pratiques simplement pour démontrer le caractère réalisable de la démarche. Nous avons présenté la notion de base littérale pour les QBF ainsi qu'un ensemble d'opérations et de propriétés qui s'y rapportent ; les principales applications de cette notion sont dans le cadre des algorithmes orientés recherche pour les QBF :

- l'extension à tout algorithme orienté recherche du calcul d'un **sat**-certificat pour une QBF ;
- l'extension à tout algorithme orienté recherche du calcul d'une base littérale dont l'interprétation est équivalente (au sens de la conservation des fonctions de Skolem) à la QBF.

Dans [1] la construction du **sat**-certificat se fait par une analyse a posteriori d'une trace relativement difficile à suivre[5] tandis que notre construction se fait au court de l'exécution de l'algorithme orienté recherche : ceci est notre principal différence et notre principal apport pour ce qui concerne les **sat**-certificats. En ce qui concerne le processus de compilation sous-jacent à l'algorithme $search\_certif\_qbf$, il serait sans doute intéressant de le mettre en perspective avec celui décrit pour les QBF de complexité uniquement $\forall\exists$ de [6]. Nous souhaitons aussi mettre ce travail en perspective avec celui sur la compilation des bases de connaissance propositionnelles [7] : en particulier, le langage de compilation des P_k et N_k de la définition 1 n'est pas évoqué et est bien sûr d'un intérêt crucial dans une étude plus poussée. Enfin nous souhaitons comparer notre approche avec d'autres qui consisteraient à compiler directement les séquences de fonctions de Skolem.

Sur le plan pratique, ce travail peut naturellement être étendu par

- une expérimentation plus poussée sur des QBF de niveau de complexité plus élevé,
- une intégration dans un algorithme orienté recherche basé sur des formules CNF,
- une adaptation de la librairie CUDD pour intégrer le retour arrière inhérent aux algorithmes orientés recherche.

Références

[1] M. Benedetti. Extracting Certificates from Quantified Boolean Formulas. In *Proceedings of 9th International Joint Conference on Artificial Intelligence (IJCAI05)*, 2005.

[2] M. Benedetti. sKizzo : a Suite to Evaluate and Certify QBFs. In *Proceedings of the 20th International Conference on Automated Deduction (CADE05)*, 2005.

[3] L. Bordeaux. Boolean and interval propagation for quantified constraints. In *First International Workshop on Quantification in Constraint Programming*, 2005.

[4] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, 1986.

[5] H. K. Büning, K. Subramani, and Xishun Zhao. Boolean functions as models for quantified boolean formulas. *Journal of Automated Reasoning*, 39(1) :49–75, 2007.

[6] S. Coste-Marquis, H. Fargier, J. Lang, D. Le Berre, and P. Marquis. Representing policies for quantified boolean formulae. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 286–296, 2006.

[7] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17 :229–264, 2002.

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

[9] E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause/term resolution and learning in the evaluation of quantified boolean formulas. *Journal of Artificial Intelligence Research*, 26 :371–416, 2007.

[10] Fabio Somenzi. CUDD : CU decision diagram package release 2.3.0. university of colorado at boulder, 1998.

[11] I. Stéphan. Algorithmes d'élimination de quantificateurs pour le calcul des politiques des formules booléennes quantifiées. In *Premières Journées Francophones de Programmation par Contraintes*, 2005.

[12] I. Stéphan. Finding models for quantified boolean formulae. In *First International Workshop on Quantification in Constraint Programming*, 2005.

[13] I. Stéphan. Boolean propagation based on literals for quantified boolean formulae. In *17th European Conference on Artificial Intelligence*, 2006.

[5] sKizzo offre des outils pour générer et analyser une telle trace en vue de calculer un **sat**-certificat dans un algorithme orienté recherche quelconque et non uniquement dans sKizzo