# Propagation Logique pour les Formules Booléennes Quantifiées

Igor Stephan

# Propagation logique pour les formules booléennes quantifiées

**Igor Stéphan**

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

**Résumé**

Cet article propose un nouvel ensemble de règles de propagation pour les formules booléennes quantifiées basées sur les littéraux et générées automatiquement grâce aux certificats pour les formules booléennes quantifiées. Cet ensemble de règles de propagation est comparé avec l'ensemble de règles de propagation déjà proposé pour les contraintes booléennes quantifiées, le système QUBOS et l'ensemble de règles de la méthode de Stålmarck. Nous esquissons une implantation en le langage CHR.

**Abstract**

This paper proposes a new set of propagation rules for Quantified Boolean Formulae based on literals and generated automatically thanks to Quantified Boolean Formulae certificates. This set of propagation rules is compared with the already proposed quantified Boolean propagation rule system, QUBOS system and Stålmarck's method. We also outline an implementation in CHR language.

## 1 Introduction

Le problème de validité pour les formules booléennes quantifiées 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 formules booléennes quantifiées 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 formules booléennes quantifiées. C'est pourquoi, l'étude de la validité des formules booléennes quantifiées est importante.

Étant donné qu'une formule booléenne quantifiée peut être ramenée à une formule booléenne (non quantifiée), la première solution semble être de réaliser cette transformation puis d'appliquer un algorithme de test de satisfiabilité. L'inconvénient majeur d'une telle approche est la taille de la formule booléenne générée qui est dans le pire des cas exponentielle par rapport à la celle de la formule booléenne quantifiée initiale. La plupart des travaux récents portant sur les procédures de décision pour la validité des formules booléennes quantifiées [22, 16, 13, 19] sont des extensions de la procédure de recherche dite de Davis-Putnam [14] pour le test de satisfiabilité des formules booléennes. D'autres procédures de décision sont basées soit sur le principe de résolution [23] (comme la Q-resolution [11] qui étend le principe de résolution pour les formules booléennes [14] aux formules booléennes quantifiées ou bien Quantor [10] qui combine efficacement la Q-resolution avec l'expansion), soit sur des algorithmes d'élimination de quantificateurs [20], soit encore sur des approches combinant le symbolique et l'utilisation de solveurs SAT (par exemple, à base de diagrammes binaires de décision [6] ou de skolémisation [8]). La plupart des systèmes précédents ont opté pour une résolution à partir de formules booléennes quantifiées sous forme prénexe et normale conjonctive. Dans [7] est proposée le système QUBOS : une approche par réduction de la portée des quantificateurs, propagation et élimination des quantificateurs sur la QBF initiale (qui n'a été mise ni en forme prénexe, ni en forme normale conjonctive) ; cette méthode est à rapprocher de celle explorée dans [21] qui applique des transformations qui préservent la validité de la formule. Enfin, dans [12] est proposée une approche basée sur la propagation de contraintes booléennes quantifiées qui étend la consistance d'arc aux contraintes quantifiées. Il existe aussi des algorithmes efficaces pour des fragments syntaxiques de l'ensemble des formules booléennes quantifiées. Par exemple, l'algorithme polynomial pour 2CNF-QBF [2] ou des heuristiques pour

les formules quantifiées Horn renommables [18].

Dans [5], une méthodologie est proposée pour construire un ensemble de règles de propagation pour des problèmes de satisfaction de contraintes basés sur des contraintes prédéfinies et données explicitement. La propagation de contraintes booléennes (cf. [4] pour un historique concis) et la propagation de contraintes quantifiées booléennes [12] vérifient cette définition. En général, les systèmes de propagation de contraintes s'appuient implicitement sur la décomposition par l'introduction de variables existentiellement quantifiées. Cette décomposition ne permet pas de capturer l'ensemble des simplifications possibles dues aux propriétés des connecteurs dans le treillis booléen. Les définitions et propriétés décrites dans ces articles sont en terme de domaine et d'arc-consistance (quantifiée). Nous nous intéressons à ces résultats plus en terme de logique propositionnelle, d'équivalence logique et de propagation par substitution comme dans la méthode de Stålmarck [24]. C'est pourquoi, nous parlerons de propagation logique plutôt que de propagation booléenne pour marquer cette distinction.

Dans [9] est introduite la notion de certificat pour une formule booléenne quantifiée. Un certificat est une application extraite d'une formule booléenne quantifiée qui permet d'en tester ou d'en générer les modèles. Nous montrerons que des certificats peuvent être extraites automatiquement les règles pour la propagation logique pour les formules booléennes quantifiées.

L'article est organisé ainsi : après des préliminaires syntaxiques et sémantiques sur les formules booléennes quantifiées en section 2, nous déroulons en section 3 un exemple pour donner l'intuition des résultats que nous décrivons en section 4 : un ensemble de règles de propagation logique basées sur les littéraux pour les formules booléennes quantifiées. Pour ce faire, nous introduisons d'abord la décomposition par introduction de littéraux quantifiées existentiellement ; puis nous décrivons la génération automatique, à partir des certificats, des règles de propagation logique basées sur les littéraux ; enfin nous proposons un algorithme complet incluant cet ensemble de règles et s'appuyant sur la sémantique des quantificateurs. Dans la section 5, nous comparons notre approche avec le système de propagation booléenne quantifiée décrit dans [12], le système QUBOS [7] et la méthode de Stålmarck [24]. La section 6 décrit sommairement une implantation dans le langage CHR sous Prolog. Enfin, nous concluons en section 7 par une présentation de nos travaux futurs.

## 2 Préliminaires

**Formules booléennes quantifiées.** Les valeurs booléennes sont notées $\mathbf{v}$ et $\mathbf{f}$. L'ensemble des symboles (ou variables) propositionnel(le)s est noté $PV$. 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 l'équivalence. Un littéral est une variable booléenne ou la négation de celle-ci. Si $l$ est un littéral alors sa variable sous-jacente est notée $|l|$ et son complémentaire $\overline{l}$ (i.e. $|v| = v$, $|\neg v| = v$, $\overline{v} = \neg v$ et $\overline{\neg v} = v$). La satisfaction propositionnelle est notée $\models$ 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. Par convention, des quantificateurs différents lient des variables différentes. Si une variable $x$ n'est pas dans la portée d'un quantificateur $qx$ alors elle est libre. L'ensemble des variables libres d'une QBF $F$ est noté $VL(F)$. Si $VL(F) = \emptyset$ alors la QBF $F$ est close. 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 propositionnelle $x$ par la formule $F$. Cette substitution est étendue aux littéraux : si $l = \neg x$ alors $G[l \leftarrow F] = G[x \leftarrow \neg F]$. Un lieur est une chaîne de caractère $q_1x_1 \ldots q_nx_n$ avec $x_1, \ldots, x_n$ des variables distinctes et $q_1 \ldots q_n$ des quantificateurs. Nous écrivons $qx_1 \ldots x_n$ pour une quelconque permutation de $qx_1 \ldots qx_n$. Une QBF $QF$ est en forme prénexe si $F$ est une formule booléenne (appelée matrice) et sous forme normale conjonctive si $F$ est une formule booléenne en forme normale conjonctive (i.e. une conjonction de disjonctions de littéraux).

**Sémantique des QBF.** La sémantique des symboles booléens est définies de manière habituelle. En particulier, de la structure de treillis booléen un certain nombres de simplifications peuvent être appliquées (nous nous intéressons plus particulièrement à la disjonction par la suite, mais tous les résultats sont transposables pour les autres connecteurs) :

| | | | |
|---|---|---|---|
| (1) | $(\bot \vee \bot) \equiv \bot$ | (2) | $(\bot \vee \top) \equiv \top$ |
| (3) | $(\top \vee \bot) \equiv \top$ | (4) | $(\top \vee \top) \equiv \top$ |
| (5) | $(\bot \vee y) \equiv y$ | (6) | $(\top \vee y) \equiv \top$ |
| (7) | $(x \vee \bot) \equiv x$ | (8) | $(x \vee \top) \equiv \top$ |
| (9) | $(x \vee x) \equiv x$ | (10) | $(x \vee \overline{x}) \equiv \top$ |

Ces règles peuvent être appliquées itérativement jusqu'à ce qu'un (unique) point fixe soit atteint. La sémantique des quantificateurs est la suivante : pour toute variable booléenne $y$ et toute QBF $F$,

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

et

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

Une QBF est valide si $F \equiv \top$. Si $y$ est une variable quantifiée existentiellement précédée par les variables quantifiées universellement $x_1, \ldots, x_n$, nous notons $\hat{y}_{x_1,\ldots,x_n}$ sa fonction de Skolem de $\{\mathbf{v}, \mathbf{f}\}^n$ dans $\{\mathbf{v}, \mathbf{f}\}$. Un modèle pour une QBF $F$ est une séquence $s$ de fonctions de Skolem qui satisfait la formule[1]. 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 : $\hat{y}_z(\mathbf{v}) = \mathbf{v}$, $\hat{y}_z(\mathbf{f}) = \mathbf{f}$, $\hat{x}_z(\mathbf{v}) = \mathbf{f}$ et $\hat{x}_z(\mathbf{f}) = \mathbf{f}$. Dans [26, 25], 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 $\forall z \exists y \exists x((x \vee y) \leftrightarrow z) \not\cong \top$. Nous ne nous intéresserons par la suite qu'aux QBF closes. Une QBF est valide si et seulement s'il existe une séquence de fonctions de Skolem qui satisfasse la formule. Selon les théorèmes :

$$\exists x \exists y F \equiv \exists y \exists x F, \forall x \forall y \equiv \forall y \forall x F, \exists x \forall y F \not\equiv \forall y \exists x F \quad (1)$$

pour une QBF $F$ quelconque, la QBF induit un ordre sur les classes d'équivalence constituées des quantificateurs identiques contigus que nous noterons $\leq$. Toute QBF a une QBF sous forme prénexe qui lui est équivalente.

**Certificat pour les QBF.** Dans [9] a été introduite la notion de certificat pour une QBF close sous forme prénexe (notion définie simultanément, sous un autre nom, dans [25]). Un certificat $\{x \mapsto (\Phi_x^+, \Phi_x^-)\}_{x \in V_\exists}$ est une application de l'ensemble des variables existentielles $V_\exists$ d'une QBF vers des couples de formules booléennes $(\Phi_x^+, \Phi_x^-)$ constituées uniquement sur les variables qui précèdent la variable dans le lieur. Le certificat peut être extrait d'une QBF prénexe $QF$ par une extension de l'algorithme QMRES [20] d'élimination de quantificateurs. D'un certificat $\{x \mapsto (\Phi_x^+, \Phi_x^-)\}_{x \in V_\exists}$ peut être extraite une QBF

$$Q \bigwedge_{x \in V_\exists} (x \vee \Phi_x^+) \wedge (\neg x \vee \Phi_x^-)$$

Cette QBF est équivalente (dans le sens de la préservation la validité) à la QBF $QF$ dont le certificat est issu, mais lui est aussi équivalente dans le sens de la préservation des fonctions de Skolem et donc des modèles [25]. Ce certificat permet de vérifier qu'un ensemble de fonctions de Skolem est un modèle [9] ou d'énumérer un à un les modèles [9, 25]. Par exemple, le certificat associé à la QBF $F = \forall z \exists y \exists x((x \vee y) \leftrightarrow z)$ est l'application $\{y \mapsto (\top, z), x \mapsto ((\neg z \vee y), z)\}$ de ce certificat peut être extraite la QBF

$$F' = \forall z \exists y \exists x((y \vee \top) \wedge (\neg y \vee z) \wedge (x \vee (\neg z \vee y)) \wedge (\neg x \vee z))$$

cette QBF vérifie les deux propriétés suivantes : $F' \equiv F$ et $F' \cong F$.

**Décomposition par introduction de variables quantifiées existentiellement.** La décomposition par introduction de variables quantifiées existentiellement introduit des variables quantifiées existentiellement pour faire apparaître les résultats intermédiaires de l'application des opérateurs. Cette décomposition préserve la validité de la QBF initiale. Par exemple, la QBF

$$\forall z \exists y \exists x((x \vee y) \leftrightarrow \neg z)$$

est décomposée en la QBF

$$\forall z \exists y \exists x \exists u \exists v(((x \vee y) \leftrightarrow u) \wedge (\neg z \leftrightarrow v) \wedge ((u \leftrightarrow v) \leftrightarrow \top)).$$

Selon les théorèmes suivants :

$$\left.\begin{array}{l}(\forall x\ (F \wedge G)) \equiv ((\forall x\ F) \wedge (\forall x\ G)) \\ (\exists x\ (F \wedge G)) \equiv ((\exists x\ F) \wedge G)\ x \text{ non libre dans } G \\ (\exists x\ (F \wedge G)) \models ((\exists x\ F) \wedge (\exists x\ G))\end{array}\right\} \quad (2)$$

cette QBF ne peut être valide que si les QBF suivantes le sont aussi :

$$\forall z \exists v(\neg z \leftrightarrow v), \exists y \exists x \exists u((x \vee y) \leftrightarrow u), \forall u \exists v((u \leftrightarrow v) \leftrightarrow \top).$$

Ce principe de décomposition est à la base des systèmes de propagation de contraintes booléennes [4, 3] et de l'algorithme de Stålmarck [24] pour la vérification du caractère tautologique d'une formule propositionnelle. Les règles de simplification deviennent alors des règles de simplification/propagation par substitution. Dans le cadre des QBF le lieur est important et est donc rajouté pour donner ce que nous appellerons un *schéma d'équivalence*. Les règles de simplification évoquées plus haut sont donc réécrites pour faire apparaître la variable existentielle introduite ainsi que le lieur. Par exemple, la règle de simplification (9) est réécrite en le schéma d'équivalence $qx \exists z((x \vee x) \leftrightarrow z)$ avec pour propagation $[z \leftarrow x]$. Ici l'ordre des quantificateurs est capital puisque la QBF $\exists z \forall x((x \vee x) \leftrightarrow z)$ n'est pas valide. Dans le schéma d'équivalence le lieur n'est pas une sous chaîne du lieur de la QBF mais précise l'ordre des quantificateurs ; ici $qx \leq \exists z$ mais les classes d'équivalences associées ne sont pas nécessairement contiguës. Il est à noter que la règle (10) ne peut pas être une règle d'un système de propagation booléenne basée sur la décomposition par introduction de variables puisque les littéraux ne sont pas autorisés dans les règles d'un tel système.

1. Cette notion de "modèle" diffère de celle en logique classique qui concerne les variables libres mais est justifiée par le fait qu'un modèle (booléen) pour une formule booléenne non quantifiée correspond exactement au modèle (QBF) de sa clôture existentielle.

## 3 Un exemple

Nous déroulons un exemple pour introduire de manière intuitive les résultats de la partie suivante. Cet exemple est issu de [24] mais légèrement transformé. Soit la QBF $\forall p \exists q F$ avec

$$F = ((((p \rightarrow p) \rightarrow p) \rightarrow (p \rightarrow q)) \rightarrow (((p \rightarrow q) \rightarrow p) \rightarrow q))$$

Au lieu d'appliquer la décomposition par introduction de variables existentiellement quantifiées, nous introduisons des littéraux existentiellement quantifiés. Nous obtenons alors la QBF équivalente

$$\begin{array}{rl} \forall p \exists q F \equiv \forall p \exists q \exists z_1 \ldots z_7 ( & \\ & ((\overline{p} \vee p) \leftrightarrow \overline{z_5}) \wedge ((z_5 \vee p) \leftrightarrow \overline{z_6}) \\ \wedge & ((\overline{p} \vee q) \leftrightarrow z_4) \wedge ((z_6 \vee z_4) \leftrightarrow \overline{z_7}) \\ \wedge & ((\overline{p} \vee q) \leftrightarrow \overline{z_1}) \wedge ((z_1 \vee p) \leftrightarrow \overline{z_2}) \\ \wedge & ((z_2 \vee q) \leftrightarrow z_3) \wedge ((z_7 \vee z_3) \leftrightarrow \top)) \end{array}$$

La QBF $\forall p \exists q F$ est valide seulement si, par les théorèmes (2), les QBF

$$\begin{array}{l} \forall p \exists z_5 ((\overline{p} \vee p) \leftrightarrow \overline{z_5}), \forall p \exists z_5 \exists z_6 ((z_5 \vee p) \leftrightarrow \overline{z_6}), \\ \forall p \exists q \exists z_4 ((\overline{p} \vee q) \leftrightarrow z_4), \exists z_6 \exists z_4 \exists z_7 ((z_6 \vee z_4) \leftrightarrow \overline{z_7}), \\ \forall p \exists q \exists z_1 ((\overline{p} \vee q) \leftrightarrow \overline{z_1}), \forall p \exists z_1 \exists z_2 ((z_1 \vee p) \leftrightarrow \overline{z_2}), \\ \forall p \exists z_2 \exists z_3 ((z_2 \vee q) \leftrightarrow z_3), \exists z_7 \exists z_3 ((z_7 \vee z_3) \leftrightarrow \top) \end{array}$$

le sont aussi.

Mais ici, la seule valeur possible pour $z_5$ est $\bot$, et donc dans la décomposition de $\forall p \exists q F$ aussi et ainsi de suite nous pouvons propager les substitutions ($[z_5 \leftarrow \bot]$) et $[z_6 \leftarrow \overline{p}]$. A ce moment du calcul, plus rien ne peut être déduit, nous devons donc appliquer maintenant la sémantique du quantificateur universel ; nous obtenons deux formules (l'une en substituant $p$ par $\top$ et l'autre en substituant $p$ par $\bot$) qui doivent être toutes les deux valides :

$$\begin{array}{rl} \exists q F[p \leftarrow \top] \equiv \exists q \exists z_1 \ldots \exists z_4 \exists z_7 ( & \\ & ((\overline{\top} \vee q) \leftrightarrow z_4) \wedge ((\overline{\top} \vee z_4) \leftrightarrow \overline{z_7}) \\ \wedge & ((\overline{\top} \vee q) \leftrightarrow \overline{z_1}) \wedge ((z_1 \vee \top) \leftrightarrow \overline{z_2}) \\ \wedge & ((z_2 \vee q) \leftrightarrow z_3) \wedge ((z_7 \vee z_3) \leftrightarrow \top)) \end{array}$$

et

$$\begin{array}{rl} \exists q F[p \leftarrow \bot] \equiv \exists q \exists z_1 \ldots \exists z_4 \exists z_7 ( & \\ & ((\overline{\bot} \vee q) \leftrightarrow z_4) \wedge ((\overline{\bot} \vee z_4) \leftrightarrow \overline{z_7}) \\ \wedge & ((\overline{\bot} \vee q) \leftrightarrow \overline{z_1}) \wedge ((z_1 \vee \bot) \leftrightarrow \overline{z_2}) \\ \wedge & ((z_2 \vee q) \leftrightarrow z_3) \wedge ((z_7 \vee z_3) \leftrightarrow \top)) \end{array}$$

Dans $\exists q F[p \leftarrow \top]$ nous avons les substitutions suivantes qui démontrent que $\exists q F[p \leftarrow \top]$ est valide :

$$[z_2 \leftarrow \bot], [z_3 \leftarrow q], [z_4 \leftarrow q], [z_1 \leftarrow \overline{q}], [z_7 \leftarrow \overline{q}]$$

De même, dans $\exists q F[p \leftarrow \bot]$ nous avons les substitutions suivantes qui démontrent que $\exists q F[p \leftarrow \bot]$ est valide :

$$[z_4 \leftarrow \top], [z_7 \leftarrow \bot], [z_3 \leftarrow \top], [z_1 \leftarrow \bot], [z_2 \leftarrow \top].$$

Nous remarquons que dans les substitutions qui précèdent (pour les deux QBF), la variable $q$ n'est substituée par aucune valeur ; nous pouvons immédiatement en déduire la validité de la QBF $\forall p \forall q F$ (c'est d'ailleurs ce qui est démontré dans [24]).

## 4 Propagation logique basée sur les littéraux pour les QBF

Cette partie décrit le cœur de notre contribution : un ensemble de règles de propagation logique basées sur les littéraux pour les QBF. Nous introduisons d'abord la décomposition par introduction de littéraux existentiellement quantifiés ; puis nous décrivons la génération automatique des règles de propagation logique basées sur littéraux pour les QBF grâce aux certificats ; enfin nous proposons un algorithme complet incluant cet ensemble de règles et s'appuyant sur la sémantique des quantificateurs.

### 4.1 Décomposition par introduction de littéraux existentiellement quantifiés

La décomposition classique par introduction de variables existentiellement quantifiées pour les formules booléennes conserve la négation comme un connecteur de la formule booléenne générée. Ainsi, un schéma d'équivalence tel que $x \vee \overline{x} \equiv z$ avec $[z \leftarrow \top]$ pour substitution ne peut être utilisé. Nous proposons une décomposition basée sur les littéraux au lieu des variables pour être en mesure d'introduire de telles règles dans notre système de propagation. La négation disparaît donc des connecteurs présents dans la formule décomposée. La fonction $\delta$ ci-dessous décompose une formule booléenne par introduction de littéraux existentiellement quantifiés ($\circ$ est un connecteur binaire quelconque, le résultat des fonctions $\delta^+$ et $\delta^-$ sont des couples (variable, décomposition), $\pi_i(c)$ pour $i = 1$ (resp. $i = 2$) est la première projection (resp. seconde projection) du couple $c$).

$$\begin{array}{l} \delta(F) = \pi_2(\delta^+(F)) \wedge (\pi_1(\delta^+(F)) \leftrightarrow \top) \\ \delta^+(x) = (x, \top), x \in PV \\ \delta^-(x) = (\overline{x}, \top), x \in PV \\ \delta^+(\neg A) = \delta^-(A), \\ \delta^-(\neg A) = \delta^+(A), \\ \delta^+(A \circ B) = (z, \ \pi_2(\delta^+(A)) \wedge \pi_2(\delta^+(B)) \\ \qquad\qquad \wedge ((\pi_1(\delta^+(A)) \circ \pi_1(\delta^+(B))) \leftrightarrow z)) \\ \delta^-(A \circ B) = (z, \ \pi_2(\delta^+(A)) \wedge \pi_2(\delta^+(B)) \\ \qquad\qquad \wedge ((\pi_1(\delta^+(A)) \circ \pi_1(\delta^+(B))) \leftrightarrow \overline{z})) \end{array}$$

Si $QF$ est une QBF prénexe, $D = \delta(F)$ la décomposition de $F$ et $X = FV(QD)$ l'ensemble des nouvelles variables existentiellement quantifiées introduites par la fonction $\delta$ alors la QBF $Q \exists X D$ est le résultat de la

décomposition par introduction de littéraux existentiellement quantifiés de $QF$ [2]. Par exemple, la QBF $\forall z \exists y \exists x((x \vee y) \leftrightarrow \neg z)$ est maintenant décomposée en $\forall z \exists y \exists x \exists u(((x \vee y) \leftrightarrow u) \wedge ((u \leftrightarrow \overline{z}) \leftrightarrow \top))$.

Selon les théorèmes (2) si $Q \bigwedge ((x \circ y) \leftrightarrow z)$ est une décomposition de la QBF $F$ alors

- $F$ est valide si et seulement si $Q \bigwedge ((x \circ y) \leftrightarrow z)$ est valide ;
- si $F$ est valide alors toutes les QBF $Q((x \circ y) \leftrightarrow z)$ sont aussi valides.

La décomposition introduit autant de nouvelles variables existentielles qu'il y a de connecteurs binaires dans la formule.

### 4.2 L'ensemble des règles de propagation logique basées sur les littéraux pour les QBF

L'ensemble de règles de propagation logique basées sur les littéraux pour les QBF présenté dans cet article (figures 1, 2 et 3) est généré automatiquement par énumération des schémas d'équivalence possibles comme décrit dans la partie suivante [3]. Cet ensemble de règles est appliqué itérativement jusqu'à ce qu'un (unique) point fixe soit atteint. L'ensemble des schémas d'équivalence est divisé en deux ensembles de taille égale : l'ensemble des schémas contradictoires issus de schémas d'équivalence non valides et l'ensemble des autres schémas. Une règle de contradiction est générée pour chaque schéma contradictoire. Une règle de contradiction arrête le calcul du point fixe et renvoie la non validité de la QBF (puisqu'au moins une des équivalences de la conjonction de la décomposition est non valide). Les autres schémas sont de quatre types différents :

- certains schémas d'équivalence sont tautologiques : dans ce cas une règle de simplification pour tautologie est générée qui élimine l'équivalence de la décomposition (le numéro de cette règle a pour indice supérieur le symbole $\models$) ;
- certains schémas d'équivalence sont seulement contingents (i.e. ils ne sont pas tautologiques mais ne donnent pas de substitution pour les variables) : dans ce cas aucune règle n'est générée (le numéro de ce schéma a pour indice supérieur le symbole ?) ;
- certains schémas d'équivalence déterminent toutes les variables du schéma par substitution : dans ce cas une règle de simplification/propagation est générée qui élimine l'équivalence de la décomposition et propage les substitutions (le numéro de la règle a pour indice supérieur le symbole $\Rightarrow$) ;
- certains schémas d'équivalence déterminent seulement une partie des variables par substitution et après cette propagation sont toujours contingents : dans ce cas une règle de propagation est générée qui propage les substitutions (le numéro de la règle a pour indice supérieur le symbole $\Rightarrow$?).

| | Schéma d'équivalence | | Propagation |
|---|---|---|---|
| $(11)^{\Rightarrow}$ | $\exists \vert y \vert$ | $\bot \vee y \leftrightarrow \bot$ | $[y \leftarrow \bot]$ |
| $(12)^{\Rightarrow}$ | $\exists \vert y \vert$ | $\bot \vee y \leftrightarrow \top$ | $[y \leftarrow \top]$ |
| $(13)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \bot \leftrightarrow \bot$ | $[x \leftarrow \bot]$ |
| $(14)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \bot \leftrightarrow \top$ | $[x \leftarrow \top]$ |
| $(15)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee x \leftrightarrow \bot$ | $[x \leftarrow \bot]$ |
| $(16)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee x \leftrightarrow \top$ | $[x \leftarrow \top]$ |
| $(17)^{\Rightarrow}$ | $\exists \vert x \vert \vert y \vert$ | $x \vee y \leftrightarrow \bot$ | $[x \leftarrow \bot], [y \leftarrow \bot]$ |
| $(18)^{\Rightarrow}$ | $\exists \vert x \vert \forall \vert y \vert$ | $x \vee y \leftrightarrow \top$ | $[x \leftarrow \top]$ |
| $(19)^{\Rightarrow}$ | $\exists \vert y \vert \forall \vert x \vert$ | $x \vee y \leftrightarrow \top$ | $[y \leftarrow \top]$ |
| $(20)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \top \leftrightarrow x$ | $[x \leftarrow \top]$ |
| $(21)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \top \leftrightarrow \overline{x}$ | $[x \leftarrow \bot]$ |
| $(22)^{\Rightarrow}$ | $q \vert z \vert \exists \vert y \vert$ | $\bot \vee y \leftrightarrow z$ | $[y \leftarrow z]$ |
| $(23)^{\Rightarrow}$ | $\exists \vert y \vert$ | $\top \vee y \leftrightarrow y$ | $[y \leftarrow \top]$ |
| $(24)^{\Rightarrow}$ | $\exists \vert y \vert$ | $\top \vee y \leftrightarrow \overline{y}$ | $[y \leftarrow \bot]$ |
| $(25)^{\Rightarrow}$ | $\exists \vert z \vert q \vert y \vert$ | $\top \vee y \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $(26)^{\Rightarrow}$ | $q \vert z \vert \exists \vert x \vert$ | $x \vee \bot \leftrightarrow z$ | $[x \leftarrow z]$ |
| $(27)^{\Rightarrow}$ | $\exists \vert z \vert q \vert x \vert$ | $x \vee \top \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $(28)^{\Rightarrow}$ | $q \vert z \vert \exists \vert x \vert$ | $x \vee x \leftrightarrow z$ | $[x \leftarrow z]$ |
| $(29)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \overline{x} \leftrightarrow x$ | $[x \leftarrow \top]$ |
| $(30)^{\Rightarrow}$ | $\exists \vert x \vert$ | $x \vee \overline{x} \leftrightarrow \overline{x}$ | $[x \leftarrow \bot]$ |
| $(31)^{\Rightarrow}$ | $\exists \vert z \vert q \vert x \vert$ | $x \vee \overline{x} \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $(32)^{\Rightarrow}$ | $\exists \vert x \vert \vert y \vert$ | $x \vee y \leftrightarrow \overline{x}$ | $[x \leftarrow \bot][y \leftarrow \top]$ |
| $(33)^{\Rightarrow}$ | $\exists \vert x \vert \forall \vert y \vert$ | $x \vee y \leftrightarrow x$ | $[x \leftarrow \top]$ |
| $(34)^{\Rightarrow}$ | $\exists \vert y \vert \forall \vert x \vert$ | $x \vee y \leftrightarrow x$ | $[y \leftarrow \bot]$ |
| $(35)^{\Rightarrow}$ | $\exists \vert x \vert \forall \vert y \vert$ | $x \vee y \leftrightarrow y$ | $[x \leftarrow \bot]$ |
| $(36)^{\Rightarrow}$ | $\exists \vert y \vert \forall \vert x \vert$ | $x \vee y \leftrightarrow y$ | $[y \leftarrow \top]$ |
| $(37)^{\Rightarrow}$ | $\exists \vert x \vert \vert y \vert$ | $x \vee y \leftrightarrow \overline{y}$ | $[x \leftarrow \top][y \leftarrow \bot]$ |
| $(38)^{\Rightarrow}$ | $\exists \vert y \vert \vert z \vert \forall \vert x \vert$ | $x \vee y \leftrightarrow z$ | $[y \leftarrow \top][z \leftarrow \top]$ |
| $(39)^{\Rightarrow}$ | $\exists \vert y \vert \forall \vert z \vert \exists \vert x \vert$ | $x \vee y \leftrightarrow z$ | $[y \leftarrow \top][x \leftarrow z]$ |
| $(40)^{\Rightarrow}$ | $\exists \vert x \vert \vert z \vert \forall \vert y \vert$ | $x \vee y \leftrightarrow z$ | $[x \leftarrow \top][z \leftarrow \top]$ |
| $(41)^{\Rightarrow}$ | $\exists \vert x \vert \forall \vert z \vert \exists \vert y \vert$ | $x \vee y \leftrightarrow z$ | $[x \leftarrow \bot][y \leftarrow z]$ |
| $(1)^{\Rightarrow ?}$ | $\exists \vert z \vert \forall \vert x \vert \exists \vert y \vert$ | $x \vee y \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $(2)^{\Rightarrow ?}$ | $\exists \vert z \vert \forall \vert y \vert \exists \vert x \vert$ | $x \vee y \leftrightarrow z$ | $[z \leftarrow \top]$ |

FIG. 1 – *Règles de simplification/propagation pour la disjonction.*

Nous ne rapportons ici que les résultats pour la disjonction. Nous obtenons 16 règles de simplification par tautologie (résumées en 10 règles dans la figure 3), 58 règles de simplification/propagation (résumées en les 10 règles de la partie sur les préliminaires et les 30

2. comme lors d'une décomposition par introduction de variables existentiellement quantifiées, l'équivalence $z \leftrightarrow \top$ est immédiatement propagée.

3. Le programme Prolog est accessible sur notre page web : `http://www.info.univ-angers.fr/pub/stephan/Research/QBF/`

premières règles de la figure 1), 2 règles de propagation (les deux dernières règles de la figure 1), 28 schémas d'équivalence contingents (résumés en 15 schémas dans la figure 4) et 104 schémas d'équivalence contradictoires (résumés en 67 schémas dans la figure 2).

| Schéma d'équivalence | | Schéma d'équivalence | |
|---|---|---|---|
| | $\bot\vee\bot\leftrightarrow\top$ | $\forall\|z\|$ | $\bot\vee\bot\leftrightarrow z$ |
| | $\bot\vee\top\leftrightarrow\bot$ | $\forall\|z\|$ | $\bot\vee\top\leftrightarrow z$ |
| $\forall\|y\|$ | $\bot\vee y\leftrightarrow\bot$ | $\forall\|y\|$ | $\bot\vee y\leftrightarrow\top$ |
| $q\|y\|$ | $\bot\vee y\leftrightarrow\overline{y}$ | $q\|y\|\forall\|z\|$ | $\bot\vee y\leftrightarrow z$ |
| $q\|z\|\forall\|y\|$ | $\bot\vee y\leftrightarrow z$ | | $\top\vee\bot\leftrightarrow\bot$ |
| $\forall\|z\|$ | $\top\vee\bot\leftrightarrow z$ | | $\top\vee\top\leftrightarrow\bot$ |
| $\forall\|z\|$ | $\top\vee\top\leftrightarrow z$ | $q\|y\|$ | $\top\vee y\leftrightarrow\bot$ |
| $\forall\|y\|$ | $\top\vee y\leftrightarrow y$ | $\forall\|y\|$ | $\top\vee y\leftrightarrow\overline{y}$ |
| $q\|y\|\forall\|z\|$ | $\top\vee y\leftrightarrow z$ | $\forall\|z\|q\|y\|$ | $\top\vee y\leftrightarrow z$ |
| $\forall\|x\|$ | $x\vee\bot\leftrightarrow\bot$ | $\forall\|x\|$ | $x\vee\bot\leftrightarrow\top$ |
| $q\|x\|$ | $x\vee\bot\leftrightarrow\overline{x}$ | $q\|x\|\forall\|z\|$ | $x\vee\bot\leftrightarrow z$ |
| $q\|z\|\forall\|x\|$ | $x\vee\bot\leftrightarrow z$ | $q\|x\|$ | $x\vee\top\leftrightarrow\bot$ |
| $\forall\|x\|$ | $x\vee\top\leftrightarrow x$ | $\forall\|x\|$ | $x\vee\top\leftrightarrow\overline{x}$ |
| $q\|x\|\forall\|z\|$ | $x\vee\top\leftrightarrow z$ | $\forall\|z\|q\|x\|$ | $x\vee\top\leftrightarrow z$ |
| $\forall\|x\|$ | $x\vee x\leftrightarrow\bot$ | $\forall\|x\|$ | $x\vee x\leftrightarrow\top$ |
| $q\|x\|$ | $x\vee x\leftrightarrow\overline{x}$ | $q\|x\|\forall\|z\|$ | $x\vee x\leftrightarrow z$ |
| $q\|z\|\forall\|x\|$ | $x\vee x\leftrightarrow z$ | $q\|x\|$ | $x\vee\overline{x}\leftrightarrow\bot$ |
| $\forall\|x\|$ | $x\vee\overline{x}\leftrightarrow x$ | $\forall\|x\|$ | $x\vee\overline{x}\leftrightarrow\overline{x}$ |
| $q\|x\|\forall\|z\|$ | $x\vee\overline{x}\leftrightarrow z$ | $\forall\|z\|q\|x\|$ | $x\vee\overline{x}\leftrightarrow z$ |
| $\exists\|x\|\forall\|y\|$ | $x\vee y\leftrightarrow\bot$ | $\forall\|x\|q\|y\|$ | $x\vee y\leftrightarrow\bot$ |
| $\exists\|y\|\forall\|x\|$ | $x\vee y\leftrightarrow\bot$ | $\forall\|y\|q\|x\|$ | $x\vee y\leftrightarrow\bot$ |
| $\forall\|x\|\|y\|$ | $x\vee y\leftrightarrow\top$ | $\forall\|x\|\|y\|$ | $x\vee y\leftrightarrow x$ |
| $q\|x\|\forall\|y\|$ | $x\vee y\leftrightarrow\overline{x}$ | $\forall\|x\|\exists\|y\|$ | $x\vee y\leftrightarrow\overline{x}$ |
| $\exists\|y\|\forall\|x\|$ | $x\vee y\leftrightarrow\overline{x}$ | $\forall\|y\|q\|x\|$ | $x\vee y\leftrightarrow\overline{x}$ |
| $\forall\|x\|\|y\|$ | $x\vee y\leftrightarrow y$ | $\exists\|x\|\forall\|y\|$ | $x\vee y\leftrightarrow\overline{y}$ |
| $\forall\|x\|q\|y\|$ | $x\vee y\leftrightarrow\overline{y}$ | $\exists\|y\|\forall\|x\|$ | $x\vee y\leftrightarrow\overline{y}$ |
| $\forall\|y\|q\|x\|$ | $x\vee y\leftrightarrow\overline{y}$ | $\exists\|x\|q\|y\|\forall\|z\|$ | $x\vee y\leftrightarrow z$ |
| $\forall\|x\|\exists\|y\|\forall\|z\|$ | $x\vee y\leftrightarrow z$ | $\forall\|x\|\|y\|\|z\|$ | $x\vee y\leftrightarrow z$ |
| $\exists\|x\|\forall\|z\|\forall\|y\|$ | $x\vee y\leftrightarrow z$ | $\forall\|x\|\exists\|z\|\forall\|y\|$ | $x\vee y\leftrightarrow z$ |
| $\forall\|x\|q\|z\|\exists\|y\|$ | $x\vee y\leftrightarrow z$ | $\exists\|y\|q\|x\|\forall\|z\|$ | $x\vee y\leftrightarrow z$ |
| $\forall\|y\|\exists\|x\|\forall\|z\|$ | $x\vee y\leftrightarrow z$ | $\exists\|y\|\forall\|z\|\forall\|x\|$ | $x\vee y\leftrightarrow z$ |
| $\forall\|y\|\exists\|z\|\forall\|x\|$ | $x\vee y\leftrightarrow z$ | $\forall\|y\|q\|z\|\exists\|x\|$ | $x\vee y\leftrightarrow z$ |
| $\exists\|z\|\forall\|x\|q\|y\|$ | $x\vee y\leftrightarrow z$ | $\forall\|z\|\exists\|x\|\forall\|y\|$ | $x\vee y\leftrightarrow z$ |
| $\forall\|z\|\exists\|y\|\forall\|x\|$ | $x\vee y\leftrightarrow z$ | | |

FIG. 2 – *Schémas d'équivalence pour les règles de contradiction*

La preuve que le point fixe de l'application itérative de ces règles est toujours atteint et est unique est basé sur les résultats de [3]. L'argument de la preuve ne peut être basé sur la réduction des domaines comme classiquement pour les systèmes de propagation de contraintes puisque toute règle ne fait pas décroître strictement le nombre de valeurs booléennes possibles pour les variables d'une règle. A la place, nous utilisons un argument classique en logique basé sur la complexité de la formule.

| | Schéma d'équivalence | |
|---|---|---|
| $(1)^{\models}$ | | $\bot\vee\bot\leftrightarrow\bot$ |
| $(2)^{\models}$ | | $\bot\vee\top\leftrightarrow\top$ |
| $(3)^{\models}$ | | $\top\vee\bot\leftrightarrow\top$ |
| $(4)^{\models}$ | | $\top\vee\top\leftrightarrow\top$ |
| $(5)^{\models}$ | $q\|x\|$ | $x\vee\top\leftrightarrow\top$ |
| $(6)^{\models}$ | $q\|x\|$ | $x\vee\overline{x}\leftrightarrow\top$ |
| $(7)^{\models}$ | $q\|y\|$ | $\top\vee y\leftrightarrow\top$ |
| $(8)^{\models}$ | $q\|x\|$ | $x\vee x\leftrightarrow x$ |
| $(9)^{\models}$ | $q\|y\|$ | $\bot\vee y\leftrightarrow y$ |
| $(10)^{\models}$ | $q\|x\|$ | $x\vee\bot\leftrightarrow x$ |

FIG. 3 – *Règles de simplification par tautologie pour la disjonction.*

| | Schéma d'équivalence | |
|---|---|---|
| $(1)^{?}$ | $\exists\|x\|\|y\|\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(2)^{?}$ | $\exists\|x\|\forall\|y\|\exists\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(3)^{?}$ | $\forall\|x\|\exists\|y\|\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(4)^{?}$ | $\exists\|y\|\forall\|x\|\exists\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(5)^{?}$ | $\forall\|y\|\exists\|x\|\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(6)^{?}$ | $\forall\|x\|\|y\|\exists\|z\|$ | $x\vee y\leftrightarrow z$ |
| $(7)^{?}$ | $\forall\|z\|\exists\|x\|\|y\|$ | $x\vee y\leftrightarrow z$ |
| $(8)^{?}$ | $\exists\|x\|\|y\|$ | $x\vee y\leftrightarrow\top$ |
| $(9)^{?}$ | $\forall\|x\|\exists\|y\|$ | $x\vee y\leftrightarrow\top$ |
| $(10)^{?}$ | $\forall\|y\|\exists\|x\|$ | $x\vee y\leftrightarrow\top$ |
| $(11)^{?}$ | $q\|x\|\exists\|y\|$ | $x\vee y\leftrightarrow x$ |
| $(12)^{?}$ | $q\|y\|\exists\|x\|$ | $x\vee y\leftrightarrow x$ |
| $(13)^{?}$ | $\exists\|x\|\|y\|$ | $x\vee y\leftrightarrow y$ |
| $(14)^{?}$ | $\forall\|x\|\exists\|y\|$ | $x\vee y\leftrightarrow y$ |
| $(15)^{?}$ | $\forall\|y\|\exists\|x\|$ | $x\vee y\leftrightarrow y$ |

FIG. 4 – *Schémas d'équivalence contingents pour la disjonction.*

### 4.3 Génération automatique des règles de propagation basées sur les littéraux

Le calcul du certificat pour un schéma d'équivalence (avec le littéral $x$ (resp. $\overline{x}$) considéré comme la variable $x$ (resp. la formule $\neg x$)) permet de déduire automatiquement la règle associée si elle existe. Durant ce calcul deux cas peuvent apparaître : le certificat démontre que le schéma d'équivalence n'est pas valide alors une règle de contradiction est générée, ou bien le certificat

démontre que l'équivalence est valide et dans ce cas le certificat lui-même permet de déduire si la règle est une règle de simplification pour tautologie, une règle de simplification/propagation, une règle de propagation ou une équivalence contingente.

- Si le certificat est vide ou $\{x \mapsto (\top, \top)\}$ alors l'équivalence est tautologique et une règle de simplification par tautologie est générée.
- Si le certificat contient un couple $(x \mapsto (\top, \bot))$ alors $x$ est équivalent à $\bot$ et une règle de propagation qui contient la substitution $[x \leftarrow \bot]$ est générée. Réciproquement, si le certificat contient le couple $(x \mapsto (\bot, \top))$ alors $x$ est équivalent à $\top$ et une règle de propagation contenant une substitution $[x \leftarrow \top]$ est générée.
- Si le certificat contient le couple $(x \mapsto (\overline{y}, y))$ alors une règle de propagation contenant une substitution $[y \leftarrow x]$ est générée. Réciproquement si le certificat contient le couple $(x \mapsto (y, \overline{y}))$ alors une règle de propagation contenant une substitution $[y \leftarrow \overline{x}]$ est générée.

Si tous les littéraux d'une équivalence sont déterminés par substitution alors la règle est une règle de simplification (qui est aussi une règle de propagation). Une règle de propagation n'est pas nécessairement une règle de simplification puisque le schéma d'équivalence peut propager une valeur booléenne pour un littéral et être en même temps contingent. Par exemple, $\{x \mapsto (\top, \top)\}$ est le certificat de l'équivalence $\exists x(x \vee \top \leftrightarrow \top)$ et est ainsi un schéma tautologique et génère donc la règle de simplification par tautologie $(5)^{\vDash}$ ; de même $\{y \mapsto (\top, \top), z \mapsto (\neg x \wedge \neg y, x \vee y)\}$ est le certificat de l'équivalence $\forall x \exists y \exists z((x \vee y) \leftrightarrow z)$ et est ainsi le schéma contingent $(3)^{?}$ et ne génère donc aucune règle ; $\{x \mapsto (\neg z, z)\}$ est le certificat de l'équivalence $\forall z \exists x((x \vee x) \leftrightarrow z)$ et génère la règle de simplification/propagation $(28)^{\Rightarrow}$ avec la substitution $[x \leftarrow z]$ car

$$\begin{array}{ll} & \forall z \exists x((x \vee x) \leftrightarrow z) \\ \equiv & \forall z \exists x((x \vee \neg z) \wedge (\neg x \vee z)) \quad \text{par le certificat} \\ \equiv & \forall z \exists x(x \leftrightarrow z) \end{array}$$

$\{z \mapsto (\bot, \top), y \mapsto (x, \top)\}$ est le certificat de l'équivalence $\exists z \forall x \exists y((x \vee y) \leftrightarrow z)$ et donc génère la règle de propagation $(1)^{\Rightarrow ?}$ avec la substitution $[z \leftarrow \top]$ puisque toutes les variables ne sont pas déterminées par substitution et le schéma d'équivalence $\forall x \exists y((x \vee y) \leftrightarrow \top)$ demeure contingent [4].

### 4.4 Un algorithme pour atteindre la complétude

Il est immédiat que la seule application des règles décrites dans cet article est incomplète pour décider si une QBF est valide ou non. Dans le cas de la propagation de contraintes booléennes, la complétude est atteinte par une boucle à deux étapes : propagation puis énumération sur une des variables restantes. Puisque toutes les variables sont quantifiées existentiellement, le choix de la variable est libre et il n'est guidé que par la recherche d'efficacité. Il n'en est pas de même pour les QBF puisque les quantificateurs sont ordonnés et qu'ils ne peuvent pas dans la plupart des cas être permutés.

Dans le cas des algorithmes de recherche, qui éliminent le quantificateur le plus externe d'abord, seulement un des quantificateurs de la classe d'équivalence (induite par l'ordre) la plus externe peut être choisi.

Soit $qxD$ la décomposition de la QBF $F$, $x$ une variable de la classe d'équivalence la plus externe et $z \in \{\top, \bot\}$. Par la sémantique du quantificateur universel, si $q = \forall$ alors $F \equiv (D[x \leftarrow \top] \wedge D[x \leftarrow \bot])$. Ainsi $F$ est valide si et seulement si $D[x \leftarrow \top]$ et $D[x \leftarrow \bot]$ le sont. En particulier, si $D[x \leftarrow z]$ n'est pas valide alors il est inutile de calculer $D[x \leftarrow \overline{z}]$ et $F$ n'est pas valide. Réciproquement, par la sémantique du quantificateur existentiel, si $q = \exists$ alors $F \equiv (D[x \leftarrow \top] \vee D[x \leftarrow \bot])$. Ainsi $F$ est valide si et seulement si $D[x \leftarrow \top]$ ou $D[x \leftarrow \bot]$ le sont. En particulier, si $D[x \leftarrow z]$ est valide alors il est inutile de calculer $D[x \leftarrow \overline{z}]$ et $F$ l'est aussi. Nous reconnaissons ici la règle du dilemme de la méthode de Stålmarck [24].

Il ne nous semble pas facile d'utiliser un algorithme d'élimination de quantificateur de l'intérieur vers l'extérieur de la formule comme dans QMRES [20] pour atteindre la complétude car les quantificateurs introduits par la décomposition sont des quantificateurs existentiels et qu'ils introduisent par leur sémantique une disjonction qui casse la décomposition.

## 5 Comparaisons

**Propagation booléenne quantifiée.** Dans [12] est proposée une extension de la consistance d'arc pour les contraintes quantifiées (appelée "consistance d'arc quantifiée"). Cette extension est appliquée aux QBF et un ensemble de règles est décrit. Cet ensemble de règles est la contrepartie dans l'univers de la propagation de contraintes des règles suivantes : de $(1)^{\Rightarrow}$ à $(8)^{\Rightarrow}$, de $(11)^{\Rightarrow}$ à $(14)^{\Rightarrow}$, de $(17)^{\Rightarrow}$ à $(19)^{\Rightarrow}$, de $(22)^{\Rightarrow}$ à $(25)^{\Rightarrow}$, de $(38)^{\Rightarrow}$ à $(41)^{\Rightarrow}$, $(1)^{\Rightarrow ?}$ et $(2)^{\Rightarrow ?}$. Les autres règles de simplification/propagation sont hors de la portée de la consistance d'arc quantifiée. De ce point de vue l'ensemble de règles proposé dans cet article est plus puissant que celui proposé dans [12].

**Le système QUBOS** Dans [7] est proposé le système QUBOS qui applique à une QBF initiale les règles de

4. Tous les certificats sont accessibles sur notre site web.

simplification décrites dans l'introduction, la réduction de la portée et l'élimination des quantificateurs par les théorèmes (1) et (2), une propagation élémentaire basée sur une extension de la propagation unitaire ainsi qu'une propagation des littéraux monotones (pour une formule en forme normale négative[5]). Une fois que tous les quantificateurs existentiels (resp. universels) ont été éliminés, éventuellement par des phases d'expansion, via la définition de la sémantique des quantificateurs, si l'application itérative des règles de simplification n'a pas été suffisante, la QBF restante est mise sous forme normale conjonctive (resp. sa négation) et transmise à un solveur SAT. Les règles de QUBOS liées à la localité des quantificateurs ne sont pas présentes dans notre approche parce que la décomposition ignore la profondeur des quantificateurs ; mais les deux approches ne sont pas incompatibles et l'insertion de l'ensemble de nos règles de contraintes (qui peuvent être vues comme globales) semble une source d'amélioration possible pour le système QUBOS.

**La méthode de Stålmarck.** Dans [24] est présentée la méthode de Stålmarck pour le problème TAUT (i.e. tester si une formule est une tautologie). TAUT est un problème de complexité co-NP complet et donc correspond au fragment des QBF prénexes avec uniquement des variables universellement quantifiées. La méthode de Stålmarck tente de prouver que la matrice de la QBF ne peut pas être équivalente à $\bot$. Cette méthode utilise d'abord la même décomposition que celle décrite dans les préliminaires et ensuite applique un ensemble de règles. Cette méthode traduit les négations ($\neg x$) en ($x \rightarrow \bot$) et ne couvre que l'ensemble des formules construites sur $\rightarrow$ et $\bot$. Ainsi, l'ensemble initial des règles de Stålmarck sont pour l'implication ; nous les transformons (cf. Figure 5) pour les comparer avec les nôtres. La règle $(32)^{\Rightarrow}$ couvre la règle $r_6$ mais elle est plus fine puisqu'elle substitue aussi la variable $y$ ; les règles $(9)^{\Rightarrow}$, $(16)^{\Rightarrow}$, $(28)^{\Rightarrow}$ et $(37)^{\Rightarrow}$ ne sont couvertes par aucune règle de la méthode de Stålmarck et peuvent y être ajoutées dans une extension de la méthode de Stålmarck basée sur les littéraux. La méthode de Stålmarck inclut aussi des schémas d'équivalence contradictoire (appelés "triplets terminaux"), ils sont aussi exprimés en fonction de l'implication et donnés en figure 5 pour comparaison. Nos schémas d'équivalence contradictoires couvrent plus de cas que ceux de Stålmarck et les suivants peuvent être ajoutés dans une extension de la méthode de Stålmarck basée sur les littéraux : $\exists|x|\ \ x \vee \overline{x} \leftrightarrow \bot$, $\exists|y|\ \ \bot \vee y \leftrightarrow \overline{y}$, $\exists|x|\ \ x \vee x \leftrightarrow \overline{x}$ et $\exists|x|\ \ x \vee \bot \leftrightarrow \overline{x}$.

La méthode de Stålmarck prouve qu'une formule propositionnelle est une tautologie en prouvant qu'il est impossible de la falsifiée ; nous ne pouvons faire de même pour une QBF dans le cas général car il existe des QBF qui n'ont pas de modèle mais dont la matrice a au moins un modèle booléen (par exemple, l'équivalence $\forall x \forall y \forall z (x \vee y \leftrightarrow z)$).

5. une formule uniquement constituée de littéraux et les opérateurs de conjonction et disjonction

| Les règles de la méthode de Stålmarck | | | |
|---|---|---|---|
| | Schémas | | Substitutions |
| $r_1$ | $\exists xy$ | $x \vee y \leftrightarrow \bot$ | $[x \leftarrow \bot][y \leftarrow \bot]$ |
| $r_2$ | $\exists xz$ | $x \vee \top \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $r_3$ | $\exists yz$ | $\top \vee y \leftrightarrow z$ | $[z \leftarrow \top]$ |
| $r_4$ | $\exists yz$ | $\bot \vee y \leftrightarrow z$ | $[z \leftarrow y]$ |
| $r_5$ | $\exists xz$ | $x \vee \bot \leftrightarrow z$ | $[z \leftarrow x]$ |
| $r_6$ | $\exists xy$ | $x \vee y \leftrightarrow \overline{x}$ | $[x \leftarrow \top]$ |
| $r_7$ | $\exists xz$ | $x \vee \overline{x} \leftrightarrow z$ | $[z \leftarrow \top]$ |

| Schémas terminaux | |
|---|---|
| | $\bot \vee \bot \leftrightarrow \top$ |
| $\exists y$ | $\top \vee y \leftrightarrow \bot$ |
| $\exists x$ | $x \vee \top \leftrightarrow \bot$ |

FIG. 5 – *L'ensemble des règles et schémas terminaux de la méthode de Stålmarck*

## 6 Esquisse d'une implantation en CHR

**Le langage CHR.** Le langage des Constraints Handling Rules (CHR) [15] est un langage de règles de réécriture destiné à l'implantation de solveurs de contraintes dans un langage hôte que nous choisirons ici comme étant Prolog. Le langage CHR permet à l'utilisateur de définir déclarativement des contraintes, le langage hôte réalisant le reste des calculs. Le langage CHR est un langage de règles gardées à gardes multiples dont la syntaxe (simplifiée) d'une règle est la suivante :

- une règle de simplification est de la forme :

$$H_1, \ldots, H_i <=> G_1, \ldots, G_j | B_1, \ldots, B_k$$

- ou bien une règle de propagation est de la forme :

$$H_1, \ldots, H_i ==> G_1, \ldots, G_j | B_1, \ldots, B_k$$

avec $i > 0$, $j \geq 0$, $k \geq 0$, $H_1, \ldots, H_i$ une suite non vide de contraintes CHR, $G_1, \ldots, G_j$ une suite de contraintes gérées par l'hôte et $B_1, \ldots, B_k$ une suite de contraintes. La sémantique déclarative est la suivante :

- pour une règle de simplification, c'est l'équivalence suivante :

$$\forall X (G_1 \wedge \ldots \wedge G_j \rightarrow (H_1 \wedge \ldots \wedge H_i \leftrightarrow (\exists Z\ B_1 \wedge \ldots \wedge B_k)))$$

– pour une règle de propagation, c'est l'implication suivante :

$$\forall X(G_1 \wedge \ldots \wedge G_j \rightarrow (H_1 \wedge \ldots \wedge H_i \rightarrow (\exists Z\ B_1 \wedge \ldots \wedge B_k)))$$

Plus informellement, la règle de simplification remplace les contraintes de la tête par celles du corps tandis que la règle de propagation ajoute les contraintes du corps tout en conservant celles de la tête.

**Du certificat d'une QBF à la règle CHR.** D'un schéma d'équivalence et de son certificat associé nous pouvons déduire la règle CHR. Nous ne traitons ici à nouveau que la contrainte sur la disjonction implantée en la contrainte CHR `'qbf+'`.

Un schéma d'équivalence $Q(x \vee y \leftrightarrow z)$ non valide est tel que $(Q(x \vee y \leftrightarrow z) \leftrightarrow \bot)$ ; nous en déduisons une règle CHR de simplification

```
'qbf+'(LX, LY, LZ) <=>
   condition(LX,LY,LZ) | fail
```

avec le prédicat `condition` assurant la garde sur le lieur et les égalités possibles des littéraux $x$, $y$ et $z$ entre-eux ou avec les constantes booléennes qui sont codées `0` pour $\bot$ et `1` pour $\top$. Par exemple, l'équivalence $\forall x \forall y(x \vee y \leftrightarrow \overline{y})$ est non valide, nous en déduisons donc une règle CHR :

```
'qbf+'(LX, LY, LZ) <=>
   universel?(LX), universel?(LY),
   lit_opp(LY,LZ) |
   fail.
```

avec le prédicat `existentiel?` (resp. `universel?`) étant tel que `existentiel?(L)` (resp. `universel?(L)`) est vrai si le littéral `L` est quantifié existentiellement (resp. universellement) et le prédicat `lit_opp` étant tel que `lit_opp(L,L_)` est vrai si le littéral `L` est le complémentaire du littéral `L_`. Un schéma d'équivalence tautologique $Q(x \vee y \leftrightarrow z)$ est tel que $(Q((x \vee y) \leftrightarrow z) \leftrightarrow \top)$ ; nous en déduisons une règle CHR de simplification

```
'qbf+'(LX, LY, LZ) <=>
   condition(LX,LY,LZ) | true.
```

Les règles de simplification/propagation s'implantent en des règles de simplification CHR. Par exemple, la règle de simplification/propagation $(32)^{\Rightarrow}$ pour le schéma d'équivalence $\exists|x|\exists|y|((x \vee y) \leftrightarrow \overline{x})$ s'implante en la règle de simplification CHR :

```
'qbf+'(LX, LY, LZ) <=>
   existentiel?(LX), existentiel?(LY),
   lit_opp(LX,LZ), (LX < LY) |
   LX <- 0, LY <- 1.
```

avec le prédicat `<` qui respecte l'ordre du lieur et le prédicat `'<-'` qui réalise la substitution sur les littéraux.

Les règles de propagation s'implantent en des règles de propagation CHR. Par exemple, la règle de propagation $(1)^{\Rightarrow ?}$ pour le schéma $\exists|z|\forall|x|\exists|y|((x \vee y) \leftrightarrow z)$ s'implante en la règle de propagation CHR :

```
'qbf+'(LX, LY, LZ) ==>
   existentiel?(LZ),
   universel?(LX), existentiel?(LY),
   (LZ < LX), (LX < LY) | LZ <- 1.
```

Les littéraux sont obtenus grâce à des variables Prolog attribuées qui permettent d'attacher des attributs aux variables. Les variables attribuées ont été introduites dans [17] pour pouvoir attaché à des variables des informations (par exemple un domaine) ou des comportements et sont particulièrement utiles pour implanter des systèmes à propagation de contraintes. L'élimination des quantificateurs entre deux phases de propagation de contraintes s'effectue par affectation puis retour-arrière. La génération de l'ensemble des règles est obtenue grâce à la table de vérité.

## 7 Travaux à venir

L'algorithme RuleMiner [1] est un algorithme pour générer des ensembles de règles de propagation pour des contraintes définies en extension sur des domaines finis. Cet algorithme nous semble pouvoir générer un ensemble plus compacte de règles, grâce à un ordre sur les règles et un ensemble de règles plus large et en cela, nous semble aussi plus puissante que la méthodologie évoquée dans l'introduction et décrite dans [5]. Ainsi nous nous intéresserons à l'impacte de l'algorithme RuleMiner sur notre ensemble de règles.

Les premiers résultats expérimentaux sur l'implantation en CHR ont été menés sur des instances structurées issues de la modélisation d'additionneurs n-bits [7]. Les résultats sont très décevants : ils ne permettent pas de décider, même pour des additionneurs de faible taille, de la validité des formules. A cela nous voyons trois raisons majeures : l'ensemble trop important de règles du système de contraintes, la manière dont le quantificateur est éliminé entre deux phases de propagation et le choix du langage d'implantation. L'application de l'algotihme RuleMiner peut nous offrir une solution pour le premier point. Pour le deuxième point, qui nous apparait le plus important, nous pensons que notre approche à caractère global doit être couplée avec une approche plus locale à la QUBOS. Enfin, pour le troisième et dernier point, nous développons actuellement une implantation réalisant cette hybridation que nous venons d'évoquer.

## Références


[1] S. Abdennadher and C. Rigotti. Automatic generation of propagation rules for finite domains. In *Principles and Practice of Constraint Programming*, pages 18–34, 2000.

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

[3] K.R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2):179–210, 1999.

[4] K.R. Apt. Some remarks on boolean constraint propagation. In *New trends in constraints*, volume 1865. Springer-Verlag, 2000.

[5] K.R. Apt and E. Monfroy. Constraint programming viewed as rule-based programming. *Theory and Practice of Logic Programming (TPLP)*, 1(6):713–750, 2001.

[6] Gilles Audemard and Lakhdar Saïs. A symbolic search based approach for quantified boolean formulas. In *SAT*, 2005.

[7] Abdelwaheb Ayari and David Basin. Qubos: Deciding quantified boolean logic using propositional satisfiability solvers. In *Formal Methods in Computer-Aided Design, Fourth International Conference, FMCAD 2002*. Springer-Verlag, 2002.

[8] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR04)*, number 3452 in LNCS. Springer, 2005.

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

[10] A. Biere. Resolve and expand. In *SAT*, 2004.

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

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

[13] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An algorithm to evaluate quantified boolean formulae and its experimental evaluation. *Journal of Automated Reasoning*, 28(2):101–142, 2002.

[14] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, July 1960.

[15] T. Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming*, 37(1-3):95–138, 1998.

[16] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for quantified boolean logic satisfiability. *Artificial Intelligence*, 145:99–120, 2003.

[17] C. Holzbaur. Specification of constraint based inference mechanism through extended unification, dissertation, dept. of medical cybernetics & ai, university of vienna, 1990.

[18] F. Letombe. Une heuristique de branchement dirigée vers les formules horn renommables quantifiées. In *Proceedings of the 7èmes Rencontres Jeunes Chercheurs en Intelligence Artificielle (RJCIA'05)*, pages 183–196, 2005.

[19] R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *TABLEAUX*, pages 160–175, 2002.

[20] G. Pan and M.Y. Vardi. Symbolic decision procedures for qbf. In *International Conference on Principles and Practice of Constraint Programming*, 2004.

[21] D.A. Plaisted, A. Biere, and Y. Zhu. A staisfiability procedure for quantified boolean formulae. *Discrete Applied Mathematics*, 130:291–328, 2003.

[22] J. Rintanen. Partial implicit unfolding in the davis-putnam procedure for quantified boolean formulae. In *Workshop on Theory and Applications of QBF, Int. Join Conference on Automated Reasoning*, Sienna, Italia, 2001.

[23] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965.

[24] Mary Sheeran and Gunnar Stålmarck. A tutorial on Stålmarck's proof procedure for propositional logic. In G. Gopalakrishnan and P. Windley, editors, *Formal Methods in Computer-Aided Design*, volume 1522, pages 82–99. Springer-Verlag, Berlin, 1998.

[25] 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.

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