La contrainte d'équivalence dans QCSP(QBF)

Vincent Barichard, Igor Stéphan

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

Résumé

Le problème de validité d'une QBF dont la matrice propositionnelle est quelconque peut être vu comme un QCSP. Dans ce cadre, nous introduisons une nouvelle contrainte : la contrainte d'équivalence et prouvons qu'elle est plus puissante que la contrainte d'égalité. Pour illustrer sa pertinence, nous nous penchons sur QCSP(QBF) qui est un QCSP instancié au problème de validité des QBF avec formule propositionnelle quelconque. Nous rapportons un ensemble de résultats expérimentaux obtenus dans le cadre d'un environnement générique pour le développement de systèmes de contraintes.

Abstract

The QBF validity problem with any propositional formula may be seen as a QCSP. In this frame, we introduce a new constraint : the equivalence constraint and prove that it is stronger than equality constraint. To illustrate its relevance, we focus on QCSP(QBF) which is a QCSP instanciated to the QBF validity problem with any propositional formula. We report some experiments in a generic constraint development environment.

1 Introduction

Le formalisme des *Formules Booléennes Quantifiées* (ou QBF)[26] qui décrit une « hiérarchie » de langages a été introduit initialement comme un outil théorique dont le but était de décrire des classes de complexité au-delà de la classe NP. Mais depuis [8] qui s'est intéressé en premier au problème de validité des QBF prénexes sous forme normale conjonctive (ou PFNC), l'ensemble des questions liées à ce formalisme est aussi devenu un domaine de l'Intelligence Artificielle car elles ont de multiples applications : la planification dans l'incertain [20], en tant que langage cible pour l'implantation de différentes questions autour de formalismes logiques [5] ou en vérification formelle [4] pour ne citer que les plus connus. Ce formalisme est une restriction de la logique du second ordre mais peut être vu plus intuitivement comme une extension de la logique propositionnelle où les symboles propositionnels sont quantifiés ; une QBF est constituée d'une chaîne $q_1x_1 \dots q_nx_n$, le lieur (avec $x_1, \dots, x_n$ des symboles propositionnels distincts et $q_1, \dots, q_n$ des quantificateurs) et une formule propositionnelle, la matrice. Comme de coutume, $\exists$ représente le quantificateur existentiel et $\forall$ le quantificateur universel. Par exemple, $\forall a \forall b \exists c(\neg(a \leftrightarrow c) \rightarrow (b \leftrightarrow c))$ est une QBF ($\neg$ représente la négation, $\rightarrow$ l'implication, $\leftrightarrow$ la bi-implication et a, b et c sont des symboles propositionnels). La sémantique des connecteurs logiques est préservée et celle des quantificateurs s'inspire de celle de la logique du premier ordre (x est un symbole propositionnel, ϕ une QBF, $\vee$ représente la disjonction, $\wedge$ représente la conjonction, les constantes propositionnelles $\top$ et $\bot$ représentent respectivement ce qui est toujours vrai et ce qui est toujours faux, $[\leftarrow]$ représente la substitution) :

$$\exists x\phi = ([x \leftarrow \bot](\phi) \vee [x \leftarrow \top](\phi))$$

et

$$\forall x\phi = ([x \leftarrow \bot](\phi) \wedge [x \leftarrow \top](\phi)).$$

Une QBF est valide si $\phi \equiv \top$ (avec le symbole « $\equiv$ » pour l'équivalence logique). Par exemple, la QBF $\forall z \exists y \exists x((x \vee y) \leftrightarrow z)$ est valide tandis que la QBF $\exists y \exists x \forall z((x \vee y) \leftrightarrow z)$ ne l'est pas. Le problème de satisfiabilité d'une formule propositionnelle (SAT) n'est autre que le problème de validité d'une QBF dont un lieur est uniquement constitué de quantificateurs existentiels. Ainsi, la plus grande part des procédures de décisions récentes pour le problème de validité des QBF (cf. [14]) sont basées sur l'algorithme de recherche dit de « Davis, Logemann et Loveland » [9] (ou DLL) pour SAT qui est une conséquence directe de la sémantique des quantificateurs existentiels. (D'autres procédures de décisions sont basées sur le principe de résolution telle que la Q-résolution [16] ou Quantor [6] ;

d'autres, tels QSAT [19] ou QMRES [18] sont basées sur un algorithme d'élimination des quantificateurs à la Fourier-Motzkin ; d'autres enfin, tel skizzo [3] sont basées sur la skolémisation).

La forme normale conjonctive (ou FNC) est un fragment privilégié pour SAT et les algorithmes de recherche car on peut détecter localement certaines formes d'insatisfiabilité. Puisque la plus grande part des procédures de décision pour QBF basées sur le DLL sont des extensions de procédures SAT [20, 12], elles sont aussi basées sur les QBF FNC. Mais ce fragment n'est pas adapté aux QBF car il ne reflète pas la dualité de la sémantique des quantificateurs qui est une conséquence directe de celle de la sémantique de la conjonction et de la disjonction. Ainsi, une forme mixte entre FND (forme normale disjonctive) et FNC [28, 21], une forme normale négative (FNN) [2, 17, 10], voire un fragment moins strict (par exemple [15] qui accepte toute formule sur $\{\neg, \wedge, \vee\}$) sont utilisés en entrée des procédures de décision modernes pour le problème de validité des QBF.

Si une QBF quelconque est considérée, par introduction de nouveaux symboles propositionnels [7] à la Tseitin [27] (voire de littéraux [24]), le problème de validité peut être vu comme un problème de *satisfaction de contraintes quantifiées* (QCSP) [11]. Par exemple, $\forall a \forall b \exists c(((a \rightarrow c) \rightarrow \neg(c \rightarrow a)) \rightarrow \neg((b \rightarrow c) \rightarrow \neg(c \rightarrow b)))$ est valide si et seulement si la décomposition suivante de la QBF est valide [1]

$$\begin{array}{ll} & \forall a \forall b \exists c \exists o_0 \exists o_1 \exists o_2 \exists o_3 \exists o_4 \exists o_5 \\ & \quad ((o_2 \rightarrow \overline{o_5}) \leftrightarrow \top) \wedge ((o_0 \rightarrow \overline{o_1}) \leftrightarrow o_2) \\ \wedge & \quad ((a \rightarrow c) \leftrightarrow o_0) \wedge ((c \rightarrow a) \leftrightarrow o_1) \\ \wedge & \quad ((o_3 \rightarrow \overline{o_4}) \leftrightarrow o_5) \wedge ((b \rightarrow c) \leftrightarrow o_3) \\ \wedge & \quad ((c \rightarrow b) \leftrightarrow o_4) \end{array}$$

seulement si chaque QBF (les *contraintes*, notées c_i) dans l'ensemble ci-dessous le sont aussi :

$$\left\{ \begin{array}{ll} c_1 : & \exists o_2 \exists o_5((o_2 \rightarrow \overline{o_5}) \leftrightarrow \top), \\ c_2 : & \exists o_0 \exists o_1 \exists o_2((o_0 \rightarrow \overline{o_1}) \leftrightarrow o_2), \\ c_3 : & \forall a \exists c \exists o_0((a \rightarrow c) \leftrightarrow o_0), \\ c_4 : & \forall a \exists c \exists o_1((c \rightarrow a) \leftrightarrow o_1), \\ c_5 : & \exists o_3 \exists o_4 \exists o_5((o_3 \rightarrow \overline{o_4}) \leftrightarrow o_5), \\ c_6 : & \forall b \exists c \exists o_3((b \rightarrow c) \leftrightarrow o_3), \\ c_7 : & \forall b \exists c \exists o_4((c \rightarrow b) \leftrightarrow o_4) \end{array} \right\}$$

La figure 1 montre des arbres de recherche dans lesquels chaque nœud (sauf la racine) est étiqueté par une substitution sur $\{\top, \bot\}$ et chaque arc est étiqueté par une propagation ; dans les deux arbres, les feuilles sont étiquetées par *succès* ou *échec*. Nous déroulons dans le premier temps une branche de l'arbre de recherche pour la figure 1.1. Si a est substitué par $\top$ alors, par propagation, le domaine booléen de o_1 est restreint aussi à $\top$ par la contrainte c_4, et selon la définition des CSP [1], une contrainte d'égalité entre o_0 et c par la contrainte c_3 ainsi qu'une contrainte d'inégalité entre o_2 et o_0 par la contrainte c_2 peuvent être ajoutées à l'ensemble des contraintes ; ce choix sur a et ses conséquences sont étiquetés par (1) dans l'arbre ainsi que dans le tableau des contraintes d'égalité et d'inégalité. Selon la sémantique, un échec revient sur le dernier choix fait pour un symbole existentiellement quantifié tandis qu'un succès revient sur le dernier choix fait pour un symbole universellement quantifié (s'il n'y en a plus, c'est un succès global). Poursuivant notre exemple, si b est substitué par $\top$ alors le domaine booléen de o_4 est restreint à $\top$ par la contrainte c_7, et une contrainte d'égalité entre o_3 et c par la contrainte c_6 ainsi qu'une contrainte d'inégalité entre o_5 et o_3 par la contrainte c_5 peuvent être ajoutées à l'ensemble des contraintes ; ceci étant étiqueté (2). Enfin, si c est substitué par $\bot$ alors le domaine de o_3 est restreint à $\bot$ par la contrainte c_6, le domaine de o_5 est restreint à $\top$ par la contrainte c_5, le domaine de o_0 est restreint à $\bot$ par la contrainte c_3 et le domaine de o_2 est restreint à $\top$ par la contrainte c_2 : ceci est consistant avec les contraintes d'égalités et d'inégalités mais inconsistant avec la contrainte c_1 (d'où l'échec) ; ceci étant étiqueté (3). La figure 1.2 montre aussi l'arbre de recherche mais pour un mécanisme de propagation qui prend en compte l'équivalence. Revenant sur le choix de a à $\top$, selon les propriétés booléennes, le domaine de o_1 est toujours restreint à $\top$ par la contrainte c_4, mais les deux équivalences $(o_0 \leftrightarrow c)$ et $(\overline{o_0} \leftrightarrow o_2)$ (et donc $(o_2 \leftrightarrow \overline{c})$ par transitivité) peuvent être considérées ; ceci étant étiqueté (4). En poursuivant l'exemple, si b est substitué par $\top$ alors le domaine de o_4 est toujours restreint à $\top$ par la contrainte c_7 et deux équivalences $(o_3 \leftrightarrow c)$ et $(\overline{o_3} \leftrightarrow o_5)$ (et donc $(o_5 \leftrightarrow \overline{c})$ par transitivité) peuvent être considérées. Bien que les domaines des symboles propositionnels o_2 et o_5 de la contrainte c_1 : $((o_2 \rightarrow \overline{o_5}) \leftrightarrow \top)$ n'aient pas changé, par l'application des équivalences $(o_2 \leftrightarrow \overline{c})$ et $(o_5 \leftrightarrow \overline{c})$, c est restreint par propagation à $\top$ (étant donné que c_1 est par application des équivalences $((\overline{c} \rightarrow c) \leftrightarrow \top)$) ; ceci étant étiqueté (5). Dans l'arbre de recherche par propagation sans équivalence, le nombre de nœuds explorés est de 11 tandis qu'il n'est que de 6 si nous prenons en compte l'équivalence. De plus, l'échec a été évité. Cet exemple démontre que la propagation prenant en compte une contrainte d'équivalence est plus puissante que la propagation sans équivalence.

Un CSP peut être conçu pour manipuler différents domaines (booléen, entier, flottant). Le terme

1. l'opérateur de complémentation $\overline{(.)}$ est tel que si x est un symbole propositionnel alors $\overline{x} = \neg x$, en particulier $\overline{\overline{x}} = x$; x et $\overline{x}$ sont des littéraux.

FIGURE 1 – Arbres de recherche

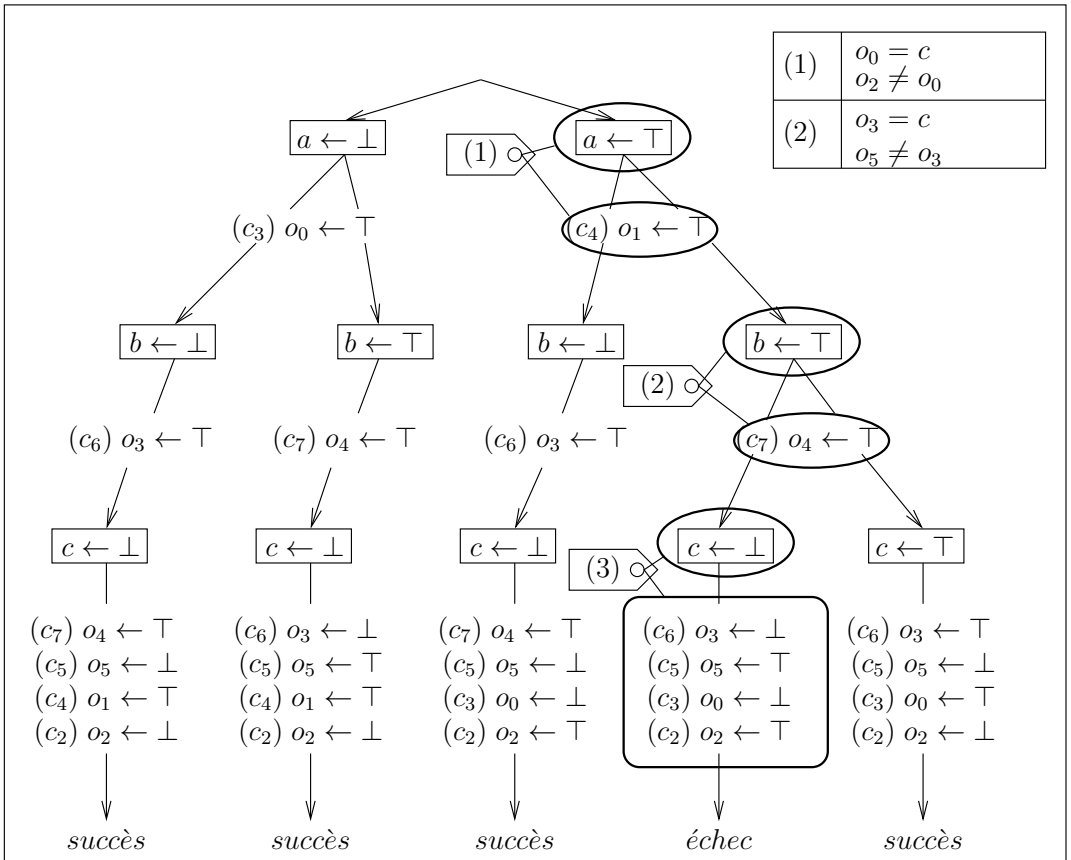

Figure 1.1 - Arbre de recherche *sans* contrainte d'équivalence

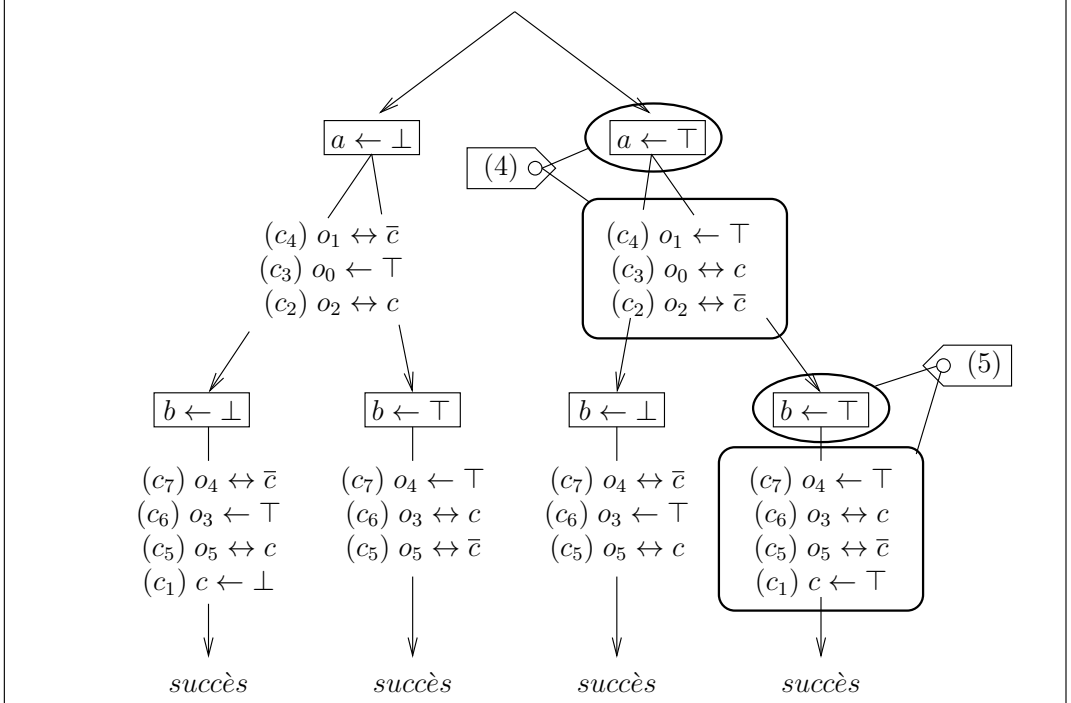

Figure 1.2 - Arbre de recherche *avec* contrainte d'équivalence

CSP(Bool) peut être employé pour dénommer les travaux autour du problème de satisfiabilité des formules booléennes. De même, nous utilisons le terme QCSP(Bool) pour caractériser les travaux autour du problème de validité des QBF. Dans ces derniers, les propagateurs maintiennent une consistance locale sur les domaines booléens ; les quantificateurs n'étant pris en charge que par l'algorithme de recherche. Pour améliorer la propagation, il a été proposé d'utiliser les propriétés spécifiques des QBF pour concevoir de nouveaux propagateurs [7] : nous nommons ces travaux QCSP(QBF). Pour aller encore plus loin, nous proposons d'ajouter la contrainte d'équivalence aux QCSP(QBF).

Une manière d'appliquer l'équivalence, en respectant l'ordre induit par le lieur, est de remplacer le symbole propositionnel par son équivalent soit par référence [24] soit par une réécriture des contraintes : dans les deux cas, nous nous situons hors du cadre classique des QCSP [11].

Dans le but de présenter notre nouvelle contrainte d'équivalence, nous faisons quelques rappels dans la section suivante sur le système de séquents S_{QBF} [25] qui la sous-tend. En nous basant sur ce système, nous utilisons un solveur QCSP qui est un solveur CSP instancié avec un algorithme de recherche quantifié et étendu pour prendre en compte un nouveau type de domaine : le domaine d'équivalence. Pour maintenir notre nouvelle contrainte, nous avons développé de nouveaux propagateurs basés sur notre nouveau domaine. Nous reportons quelques expériences dans le cadre de l'environnement générique pour le développement de systèmes de contraintes Gecode [22].

2 Préliminaire de théorie de la preuve

L'introduction de la contrainte d'équivalence s'appuie sur le calcul syntaxique S_{QBF} présenté en détail dans [25]. Ce système est basé sur la notion de relation de formules [23] (propositionnelle). Une relation de formule R pour une formule F est définie comme étant une relation d'équivalence sur l'ensemble des sous-formules avec pour contrainte que pour toute sous-formule A et B si $R(A, B)$ alors $R(\overline{A}, \overline{B})$. Une classe d'une relation de formules R contenant un élément A est notée $[A]_R$ (et plus simplement $[A]$ lorsque la relation de formules est clairement explicitée par le contexte). Ce formalisme est étendu aux QBF QM en ajoutant un lieur à la partition P des sous-formules de M et en écrivant la relation de formules $(Q; P)$.

La relation de formules la plus fine pour une QBF prénexe QM est la relation identité Id_{QM} qui est la partition des singletons des sous-formules de M associée au lieur Q. La relation de formules $R([A]_R = [B]_R)$ (notée plus simplement par la suite $R([A] = [B])$) pour une QBF prénexe QM est la relation de formules la plus fine pour QM telle qu'elle est un raffinement de R et $([A]_R = [B]_R)$.

Le système syntaxique S_{QBF} pour les QBF est un ensemble de règles $\frac{conclusion}{prémisse}$ qui opèrent sur des relations de formules formant un arbre de déduction dont la racine est la relation de formules $Id_{QM}([M] = [\top])$. Une relation de formules est *explicitement contradictoire* si une des conditions suivantes est vérifiée :

1. il existe une formule F dans la relation de formules telle que $[F] = [\overline{F}]$;
2. il existe une classe qui contient au moins deux symboles propositionnels universellement quantifiés du lieur ;
3. il existe une classe qui contient un symbole propositionnel universellement quantifié u du lieur et un symbole propositionnel existentiellement quantifié e tel que e est plus à gauche que u dans le lieur.

Les règles du système S_{QBF} ne sont présentées que pour le fragment $\{\rightarrow, \top\}$. Pour simplifier, dans la description de la prémisse et de la conclusion de la règle, seuls le lieur et les classes pertinentes sont notées, les classes invariantes ne sont pas décrites (la classe $[\top]$ est implicitement toujours présente).

Le système S_{QBF} est constitué de deux règles d'élimination du quantificateur existentiel :

$$\exists\top : \frac{(Q;[x]=[\top])}{(\exists x Q;[x])} \qquad \exists\bot : \frac{(Q;[\overline{x}]=[\top])}{(\exists x Q;[x])}$$

d'une règle d'élimination du quantificateur universel :

$$\forall : \frac{(Q;[\overline{x}]=[\top]) \quad (Q;[x]=[\top])}{(\forall x Q;[x])}$$

et neuf règles de fusion des classes :

$$1 : \frac{(Q;[\overline{F_X}]=[\top],[\overline{F_Y}]=[\top])}{(Q;[(F_X \rightarrow F_Y)]=[\overline{F_Y}])} \qquad 2 : \frac{(Q;[F_X]=[\top],[F_Y]=[\top])}{(Q;[(F_X \rightarrow F_Y)]=[F_X])}$$

$$3 : \frac{(Q;[F_X]=[\top],[\overline{F_Y}]=[\top])}{(Q;[(F_X \rightarrow F_Y)]=[\top])} \qquad 4 : \frac{(Q;[(F_X \rightarrow F_Y)]=[\overline{F_X}])}{(Q;[(F_X \rightarrow F_Y)],[\overline{F_Y}]=[\top])}$$

$$5 : \frac{(Q;[(F_X \rightarrow F_Y)]=[\overline{F_X}])}{(Q;[(F_X \rightarrow F_Y)],[F_X]=[\overline{F_Y}])} \qquad 6 : \frac{(Q;[(F_X \rightarrow F_Y)]=[\top])}{(Q;[(F_X \rightarrow F_Y)],[\overline{F_X}]=[\top])}$$

$$7 : \frac{(Q;[(F_X \rightarrow F_Y)]=[\top])}{(Q;[(F_X \rightarrow F_Y)],[F_Y]=[\top])} \qquad 8 : \frac{(Q;[(F_X \rightarrow F_Y)]=[\top])}{(Q;[(F_X \rightarrow F_Y)],[F_X]=[F_Y])}$$

$$9 : \frac{(Q;[(F_X \rightarrow F_Y)]=[F_Y])}{(Q;[(F_X \rightarrow F_Y)],[F_X]=[\top])}$$

Un *arbre de déduction* est défini inductivement, chaque nœud correspondant à une relation de formule non explicitement contradictoire. L'arbre croît à l'aide des règles du système S_{QBF}.

Un *axiome* est une relation de formules

1. non explicitement contradictoire ne contenant que deux classes et

2. qui est telle que l'on ne puisse construire un arbre de déduction avec l'axiome pour racine et contenant une relation de formules explicitement contradictoire.

Une *preuve* pour une QBF QM dans le système S_{QBF} est un arbre de déduction de racine $Id_{QM}([M] = [\top])$ tel que toute feuille soit un axiome.

La définition de l'axiome prévient la contradiction dans une classe. Cette définition se réduit uniquement à la condition 1 si l'utilisation de la règle d'élimination du quantificateur n'est utilisée que lorsque aucune autre règle ne peut plus l'être (car alors nécessairement des règles auraient déjà déduit la contradiction) ce qui sera le cas dans ce qui suit.

Le système S_{QBF} est correct et complet vis-à-vis de la sémantique des QBF [25].

La figure 2 développe l'arbre de preuve pour l'exemple introductif.

3 De QBF vers QCSP(QBF)

Dans le but de définir notre QCSP(QBF), dont le système S_{QBF} est la justification dans la théorie de la preuve, comme un CSP associé à un algorithme de recherche quantifié, nous avons besoin de définir un nouveau domaine pour les variables constitué d'un domaine booléen et un *domaine d'équivalence* avec des opérations qui respectent les propriétés liées à l'ordre des quantificateurs et de l'équivalence.

Définition 1 (domaine d'équivalence) *L'ensemble des polarités ordonnées est l'ensemble* $\mathbb{N}^{+-} = \{i^-, i^+ | i \in \mathbb{N}^*\}$.

- *Une opération de complémentation* $\overline{(.)}$ *sur* $\mathbb{N}^{+-}$ *est définie par* $\overline{i^+} = i^-$ *et* $\overline{i^-} = i^+$ *pour tout* $i \in \mathbb{N}^*$.
- *Une relation d'ordre* $\prec$ *sur* $\mathbb{N}^{+-}$ *est définie ainsi : pour tout* $i \in \mathbb{N}^*$, $i^- \prec (i+1)^-$, $i^- \prec (i+1)^+$, $i^+ \prec (i+1)^-$ *et* $i^+ \prec (i+1)^+$.
- *Une fonction* $[.]$ *de* $\mathbb{N}^{+-}$ *dans* $2^{\mathbb{N}^{+-}}$ *est définie ainsi : pour tout* $i \in \mathbb{N}^*$, $[i^-] = \{1^-, 1^+, \ldots, (i-1)^-, (i-1)^+, i^-\}$ *et* $[i^+] = \{1^-, 1^+, \ldots, (i-1)^-, (i-1)^+, i^+\}$.
- *Une fonction constDomEq de* $\{\exists, \forall\} \times \mathbb{N}^*$ *dans* $2^{\mathbb{N}^{+-}}$ *est définie ainsi :*

  si $q = \exists$ ***alors***
  $constDomEq(q, i) = [i^+]$
  sinon
  $constDomEq(q, i) = \{i^+\}$
  fin si

L'ensemble des domaines d'équivalence est l'ensemble $ED = \{[i^-], [i^+], \{i^-\}, \{i^+\} | i \in \mathbb{N}^*\}$. *Une opération de complémentation* $\overline{(.)}$ *sur* ED *est définie ainsi : pour tout* $d \in ED$, $\overline{d} = (d \setminus \max(d)) \cup \{\overline{\max(d)}\}$.

L'intuition du domaine d'équivalence pour une variable est comment celle-ci est connectée à sa classe et avec quelle polarité. Pour l'exemple introductif, les domaines d'équivalence initiaux des variables sont :

$$\begin{array}{l}
\mathrm{DomEq}(a) = \{1^+\}, \mathrm{DomEq}(b) = \{2^+\}, \\
\mathrm{DomEq}(c) = [3^+], \mathrm{DomEq}(o0) = [4^+], \\
\mathrm{DomEq}(o1) = [5^+], \mathrm{DomEq}(o3) = [6^+], \\
\mathrm{DomEq}(o4) = [7^+], \mathrm{DomEq}(o2) = [8^+], \\
\mathrm{DomEq}(o5) = [9^+].
\end{array}$$

Définition 2 (relations et opérateurs de filtrage du CSP(QBF)) *Le domaine d'une variable* v *est une paire* $(DomBool(v), DomEq(v))$ *dans* $2^{\{\top,\bot\}} \times ED$. *La table 1 définit quatre relations unaires* $(= \top)$, $(= \bot)$, $(\neq \top)$ *et* $(\neq \bot)$, *deux relations binaires* $\sim$ *et* $\not\sim$ *et une relation ternaire imp avec leurs opérateurs de filtrage respectifs.*

Les opérateurs de filtrage maintiennent un domaine d'équivalence de telle manière que si les conditions de l'explicitement contradictoire sont vérifiées alors celui-ci devient vide. Ces opérateurs de filtrage appliquent également les neuf règles de fusion du système S_{QBF}.

Nous définissons maintenant notre QCSP(QBF).

Définition 3 (QCSP(QBF)) *Un QCSP(QBF) est un quintuplet* $\langle \mathbf{V}, ordre, quant, \mathbf{D}, \mathbf{C} \rangle$ *avec*

$$\mathbf{D} = \{(\{\bot, \top\}, constDomEq(quant(v), ordre(v))) | v \in \mathbf{V}\}$$

Chaque symbole propositionnel de la QBF est associé à une variable et $\mathbf{C}$ *est un ensemble de contraintes sur les variables de* $\mathbf{V}$ *des relations* $(= \top)$, $(= \bot)$, $(\neq \top)$, $(\neq \bot)$, $\sim$, $\not\sim$ *et imp.*

Nous illustrons sur quelques exemples comment les domaines d'équivalence sont mises à jour lorsque les opérateurs de filtrage sont appliqués. Soit le lieur $\forall a \forall b \exists c \exists d$ et les domaines d'équivalence associés : $\{1^+\}, \{2^+\}, [3^+]$ et $[4^+]$. Soit la formule $(a \leftrightarrow b)$, en appliquant l'opérateur $(a \sim b)$ (cf table 1) nous obtenons deux nouveaux domaines $\mathrm{DomEq}(a) = \mathrm{DomEq}(b) = \emptyset$, un échec est constaté et la propagation est interrompue. Nous obtenons le même résultat si une variable exisentiellement quantifiée est mise en équivalence avec une variable universellement quantifiée telles que la première précède la seconde dans le lieur. Maintenant considérons la formule $((d \leftrightarrow \overline{c}) \wedge (c \leftrightarrow a))$. Les deux opérateurs de filtrage impliqués sont $(d \not\sim c)$ et $(c \sim a)$. Nous appliquons en premier la contrainte $(d \not\sim c)$ et mettons à jour le domaine d'équivalence de la variable d à $[3^-]$; alors nous appliquons l'opérateur $(c \sim a)$ et mettons à jour le domaine d'équivalence de c à jour à $\{1^+\}$; puisque le domaine de la variable c a changé, la propagation réveille la contrainte $(d \not\sim c)$ et mettons à jour le domaine d'équivalence de la variable d à

FIGURE 2 – Arbre de preuve pour la QBF $\forall a \forall b \exists c M$ avec $M = (((a \to c) \to \neg(c \to a)) \to \neg((b \to c) \to \neg(c \to b)))$, $F_0 = (a \to c)$, $F_1 = (c \to a)$, $F_4 = (c \to b)$, $F_3 = (b \to c)$, $F_2 = (F_0 \to \overline{F_1})$, $F_5 = (F_3 \to \overline{F_4})$ et $M = (F_2 \to \overline{F_5})$. Les classes fusionnées apparaissent dans la partie supérieure du séquent.

$$\dfrac{\dfrac{\dfrac{\dfrac{\nabla_\bot}{(\forall b \exists c; [M, \top, \overline{a}] = [F_0], [F_1, \overline{c}])}\,{}_9}{(\forall b \exists c; [M, \top, \overline{a}], [F_1] = [\overline{c}])}\,{}_6}{(\forall b \exists c; [M, \top] = [\overline{a}])}\,{}_4 \qquad \dfrac{\dfrac{\dfrac{\nabla_\top}{(\forall b \exists c; [M, \top, a, F_1], [F_0] = [c])}\,{}_4}{(\forall b \exists c; [M, \top, a] = [F_1])}\,{}_9}{(\forall b \exists c; [M, \top] = [a])}\,{}_7}{(\forall a \forall b \exists c; [M] = [\top])}\,\forall$$

avec $\nabla_\bot$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\varepsilon; [F_3, M, \top, \overline{a}, F_0, \overline{b}, \overline{F_5}, F_4, F_1, \overline{c}, \overline{F_2}])}{(\exists c; [F_3, M, \top, \overline{a}, F_0, \overline{b}] = [\overline{F_5}, F_4, F_1, \overline{c}, \overline{F_2}])}\,\exists\bot}{(\exists c; [F_3, M, \top, \overline{a}, F_0, \overline{b}], [\overline{F_5}] = [F_4, F_1, \overline{c}, \overline{F_2}])}\,{}_8}{(\exists c; [F_3] = [M, \top, \overline{a}, F_0, \overline{b}], [F_4, F_1, \overline{c}, \overline{F_2}])}\,{}_9}{(\exists c; [M, \top, \overline{a}, F_0, \overline{b}], [F_4] = [F_1, \overline{c}, \overline{F_2}])}\,{}_6}{(\exists c; [M, \top, \overline{a}, F_0] = [\overline{b}], [F_1, \overline{c}, \overline{F_2}])}\,{}_4 \qquad \dfrac{\dfrac{\dfrac{\dfrac{(\exists c; [M, \top, \overline{a}, F_0, b, F_4], [F_1, \overline{c}, \overline{F_2}, \overline{F_3}, F_5])}{(\exists c; [M, \top, \overline{a}, F_0, b, F_4], [F_1, \overline{c}, \overline{F_2}, \overline{F_3}] = [F_5])}\,{}_8}{(\exists c; [M, \top, \overline{a}, F_0, b, F_4], [F_1, \overline{c}, \overline{F_2}] = [\overline{F_3}])}\,{}_4}{(\exists c; [M, \top, \overline{a}, F_0, b] = [F_4], [F_1, \overline{c}, \overline{F_2}])}\,{}_9}{(\exists c; [M, \top, \overline{a}, F_0] = [b], [F_1, \overline{c}, \overline{F_2}])}\,{}_7}{(\forall b \exists c; [M, \top, \overline{a}, F_0], [F_1, \overline{c}] = [\overline{F_2}])}\,\forall$$

et $\nabla_\top$:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\exists c; [M, \top, a, F_1, \overline{b}, F_3], [\overline{F_0}, \overline{c}, F_2, F_4, \overline{F_5}])}{(\exists c; [M, \top, a, F_1, \overline{b}, F_3], [F_0, c, \overline{F_2}, \overline{F_4}] = [F_5])}\,{}_8}{(\exists c; [M, \top, a, F_1, \overline{b}] = [F_3], [F_0, c, \overline{F_2}, \overline{F_4}])}\,{}_9}{(\exists c; [M, \top, a, F_1, \overline{b}], [F_0, c, \overline{F_2}] = [\overline{F_4}])}\,{}_6}{(\exists c; [M, \top, a, F_1, \overline{b}], [F_0, c, \overline{F_2}])}\,{}_4 \qquad \dfrac{\dfrac{\dfrac{\dfrac{(\varepsilon; [M, \top, a, F_1, b, F_4, F_0, c, \overline{F_2}, F_3, \overline{F_5}])}{(\exists c; [M, \top, a, F_1, b, F_4] = [F_0, c, \overline{F_2}, F_3, \overline{F_5}])}\,\exists\top}{(\exists c; [M, \top, a, F_1, b, F_4], [F_0, c, \overline{F_2}, F_3] = [\overline{F_5}])}\,{}_8}{(\exists c; [M, \top, a, F_1, b, F_4], [F_0, c, \overline{F_2}] = [F_3])}\,{}_4}{(\exists c; [M, \top, a, F_1, b] = [F_4], [F_0, c, \overline{F_2}])}\,{}_9}{(\exists c; [M, \top, a, F_1, b], [F_0, c, \overline{F_2}])}\,{}_7}{(\forall b \exists c; [M, \top, a, F_1], [F_0, c] = [\overline{F_2}])}\,\forall$$

Table 1 – Relations et opérateurs de filtrage pour CSP(QBF)

$v = \top$, v une variable	$\mathrm{DomBool}(v) = \{\top\}$
$v = \bot$, v une variable	$\mathrm{DomBool}(v) = \{\bot\}$
$v \neq \top$, v une variable	$\mathrm{DomBool}(v) = \{\bot\}$
$v \neq \bot$, v une variable	$\mathrm{DomBool}(v) = \{\top\}$
$v_i \sim v_j$, v_i, v_j deux variables	**si** $\overline{\mathrm{DomEq}(v_i)} = \mathrm{DomEq}(v_j)$ ou $\mathrm{DomEq}(v_i) \cap \mathrm{DomEq}(v_j) = \emptyset$ **alors** $\mathrm{DomEq}(v_i) = \mathrm{DomEq}(v_j) = \emptyset$ **sinon si** $\max(\mathrm{DomEq}(v_i)) \prec \max(\mathrm{DomEq}(v_j))$ **alors** $\mathrm{DomEq}(v_j) = \mathrm{DomEq}(v_i)$ **sinon** $\mathrm{DomEq}(v_i) = \mathrm{DomEq}(v_j)$ **fin si**
$v_i \not\sim v_j$, v_i, v_j deux variables	**si** $\mathrm{DomEq}(v_i) = \mathrm{DomEq}(v_j)$ ou $\overline{\mathrm{DomEq}(v_i)} \cap \mathrm{DomEq}(v_j) = \emptyset$ **alors** $\mathrm{DomEq}(v_i) = \mathrm{DomEq}(v_j) = \emptyset$ **sinon si** $\max(\mathrm{DomEq}(v_i)) \prec \max(\mathrm{DomEq}(v_j))$ **alors** $\mathrm{DomEq}(v_j) = \overline{\mathrm{DomEq}(v_i)}$ **sinon** $\mathrm{DomEq}(v_i) = \overline{\mathrm{DomEq}(v_j)}$ **fin si**
$\mathrm{imp}(v_i, v_j, v_k)$, v_i, v_j, v_k des variables ou $\top$ ou $\bot$	**cas** $v_k \not\sim v_j : v_i \neq \top, v_j \neq \top$ **cas** $v_i \sim v_k : v_i = \top, v_j = \top$ **cas** $v_k = \bot : v_i = \top, v_j \neq \top$ **cas** $v_j = \bot : v_k \not\sim v_i$ **cas** $v_i \not\sim v_j : v_k \not\sim v_i$ **cas** $v_i = \bot : v_k = \top$ **cas** $v_j = \top : v_k = \top$ **cas** $v_i \sim v_j : v_k = \top$ **cas** $v_i = \top : v_k \sim v_j$

$\{1^-\}$. Ainsi à la fin de la propagation, nous obtenons trois domaines d'équivalences : DomEq(a) = $\{1^+\}$, DomEq(c) = $\{1^+\}$ et DomEq(d) = $\{1^-\}$.

La propagation seule n'est pas suffisante pour résoudre un QCSP. Pour ce faire, nous avons besoin d'un algorithme de recherche quantifié pour atteindre la complétude vis-à-vis du QCSP [11].

Définition 4 (algorithme de recherche quantifié $QCSP_QBF$**)**

Entrée: *Une QBF* ϕ
Sortie: `valide` *si la QBF* ϕ *est valide et* `non_valide` *sinon*
$C :=$ *décompose*(ϕ)
tant que *vrai* ***faire***
  selon *propage*(C) ***faire***
    cas *échec*
    retour-arrière jusqu'au dernier choix sur un symbole propositionnel existentiellement quantifié
    si *aucun* ***alors***
      retourner `non_valide`
    fin si
    cas *succès*
    retour-arrière jusqu'au dernier choix sur un symbole propositionnel universellement quantifié
    si *aucun* ***alors***
      retourner `valide`
    fin si
    cas *branche*
    sélectionne le symbole propositionnel suivant x
    sélectionne une valeur booléenne K *pour le symbole propositionnel*
    ajoute à C *la contrainte* $(x = K)$
  fin selon
fin tant que

Dans la définition 4, la fonction *décompose* fait référence à la décomposition d'une formule en contraintes selon l'introduction de littéraux existentiellement quantifiés [24] et la fonction *propage* propage toute modification des domaines jusqu'à ce qu'un point-fixe soit atteint.

Les valeurs des nouveaux domaines accolés aux domaines booléens ne sont pas utilisées lors des branchements, ils n'augmentent donc pas la taille de l'arbre de recherche et sont seulement utilisés lors des phases de propagations.

Les règles d'élimination du système S_{QBF} sont assurées par l'algorithme de recherche quantifié. Le calcul de point-fixe de la propagation permet de définir l'axiome du système de séquents uniquement par sa condition 1 (i.e. la mise à jour d'une relation de formules explicitement contradictoire).

Le théorème suivant fait le lien entre l'algorithme $QCSP_QBF$ et le système S_{QBF}.

Théoreme 1 (Correction et complétude de l'algorithme $QCSP_QBF$ **vis-à-vis du système** S_{QBF}**)** *Soit une QBF* QM. $Id_{QM}([M] = [\top])$ *admet une preuve si et seulement si* $QCSP_QBF(QM)$ *retourne* `valide`. $Id_{QM}([M] = [\top])$ *n'admet aucune preuve si et seulement si* $QCSP_QBF(QM)$ *retourne* `non_valide`.

Preuve esquissée du théorème 1. Le domaine d'équivalence et les opérateurs de filtrage associés assurent la notion d'explicitement contradictoire : un domaine d'équivalence pour une variable ne peut contenir qu'une seule polarité (ou être vide) ; quand un domaine d'équivalence pour une variable universellement quantifiée est mise à jour avec le domaine d'équivalence d'une autre variable universellement quantifiée alors les deux domaines deviennent vide car l'intersection l'est ; quand un domaine d'équivalence pour une variable existentiellement quantifiée est mise à jour avec le domaine d'équivalence d'une variable universellement quantifiée telle que la première précède la seconde dans le lieur alors les deux domaines deviennent vide car l'intersection l'est. L'algorithme de recherche quantifiée implémente le système S_{QBF}. Chaque exécution de l'algorithme peut être simulée par une preuve dans le système S_{QBF}. Réciproquement, de chaque preuve dans le système S_{QBF} il est possible de construire un arbre de recherche pour l'algorithme de recherche quantifié ; celui-ci étant construit récursivement par permutation des règles de fusion et montée dans l'arbre des règles d'élimination des quantificateurs.

4 Résultats expérimentaux

Nous avons implanté la contrainte d'équivalence dans Gecode [22], un environnement générique de développement de contraintes (i.e. une boîte à outils pour développer des systèmes à base de contraintes) implémenté en C++. Nous n'avons en rien eu besoin de modifier Gecode. Nous avons implémenté le domaine d'équivalence avec des intervalles et déconnecté toutes les optimisations. La table 2 rapporte quelques résultats expérimentaux. La première colonne donne le nom du benchmark issu de la bibliothèque $QBFLIB_1.0$ [13] ; la seconde colonne montre le lieur du benchmark ; les troisième, quatrième et cinquième colonnes rapportent respectivement le nombre de propagations, le nombre de nœuds de l'arbre de recherche et le nombre d'échecs pour le solveur *sans* la contrainte d'équivalence ; la sixième colonne montre le gain en temps de calcul ($= \frac{\text{le temps cpu sans*100}}{\text{le temps cpu avec}} - 100$, ∞ signifie que le solveur n'a pas été en mesure de décider en

Table 2 – Résultats expérimentaux (#P, #N, #E pour respectivement le nombre de propagations, de noeuds et d'échecs)

		solveur sans			solveur avec			
instances	lieur	#P	#N	#E	gain	#P	#N	#E
counter4_2.qbf	$\exists^{12}\forall^{8}$	5630	556	3	10%	3911	205	3
counter4_3.qbf	$\exists^{16}\forall^{12}$	80326	8596	7	100%	54176	3369	7
counter4_4.qbf	$\exists^{20}\forall^{16}$	1237107	136431	15	190%	741592	39066	15
counter4_5.qbf	$\exists^{24}\forall^{20}$	19857976	2179745	31	160%	13065634	827942	31
counter4_6.qbf	$\exists^{28}\forall^{24}$	318342127	34935929	65	190%	203660692	11905482	65
counter5_2.qbf	$\exists^{15}\forall^{10}$	17102	2114	3	30%	9844	557	3
counter5_4.qbf	$\exists^{24}\forall^{20}$	16623813	2134043	15	320%	8610737	483418	15
counter6_2.qbf	$\exists^{18}\forall^{12}$	56410	8296	3	110%	24724	1497	3
counter6_4.qbf	$\exists^{30}\forall^{24}$	207892331	31010694	15	∞	91037164	5557142	15
counter7_2.qbf	$\exists^{21}\forall^{14}$	198683	32942	3	270%	63115	4017	3
counter8_2.qbf	$\exists^{16}\forall^{24}$	719668	131380	3	750%	161471	10829	3
ring4_2.qbf	$\exists^{21}\forall^{14}$	6962131	1053640	516	230%	2875645	295639	433
ring5_2.qbf	$\exists^{24}\forall^{16}$	26652296	4291339	614	410%	8600128	823310	523
ring6_2.qbf	$\exists^{27}\forall^{18}$	101449882	17590062	712	620%	25932045	2434105	621
semaphore3_2.qbf	$\exists^{27}\forall^{18}$	54782065	1304353	23	100%	31154191	652233	23
semaphore_2.qbf	$\exists^{20}\forall^{14}$	2691738	80409	23	90%	1571193	40257	23

moins de 30 minutes) ; les septième, huitième et neuvième colonnes rapportent respectivement le nombre de propagation, le nombre de nœuds de l'arbre de recherche et le nombre d'échecs pour le solveur *avec* la contrainte d'équivalence. La principale conclusion est la suivante : le solveur avec contrainte d'équivalence est *toujours* plus rapide que le solveur sans celle-ci.

5 Conclusion

Nous avons défini dans cet article une nouvelle contrainte, la contrainte d'équivalence, et présenté comment le problème de validité des QBF pouvait être vu comme un CSP associé à un algorithme de recherche quantifié. Tout élément retiré par la contrainte classique d'égalité l'est egalement par notre contrainte d'équivalence. Mais elle permet aussi de supprimer des éléments supplémentaires non filtrés par la contrainte usuelle. Le filtrage réalisé par la contrainte d'équivalence est donc plus puissant que celui de la contrainte d'égalité. Dans nos résultats expérimentaux, la résolution est plus rapide si la contrainte d'équivalence est présente que si elle est omise.

La puissance du filtrage réside dans sa capacité à éliminer le maximum de valeurs ne pouvant faire partie d'une solution du problème. Dans le cas de domaines booléens, le filtrage peut au plus retirer une valeur par domaine sans détecter d'inconsistance. Dans ce sens, le filtrage est sans doute plus performant pour des domaines de plus grandes tailles. Notre contrainte d'équivalence dépasse le cadre des domaines booléens et peut être étendue à des domaines de plus grande taille. Nous pensons donc à l'adapter pour te tels domaines et espérons que là aussi les performances du solveur s'en trouveraient améliorées.

Références

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

[2] A. Ayari and D. Basin. QUBOS : Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers. In *Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD'02)*, pages 187–201, 2002.

[3] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05)*, pages 285–300, 2005.

[4] M. Benedetti and H. Mangassarian. Experience and Perspectives in QBF-Based Formal Verification. *Journal on Satisfiability, Boolean Modeling and Computation*, 5 :133–191, 2008.

[5] P. Besnard, T. Schaub, H. Tompits, and S. Woltran. Paraconsistent reasoning via quantified

Boolean formulas, I : Axiomatising signed systems. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 320–331, 2002.

[6] A. Biere. Resolve and Expand. In *Proceedings of the 7th International Confrerence on Theory and Applications of Satisfiability Testing (SAT'04)*, pages 59–70, 2004.

[7] L. Bordeaux and E. Monfroy. Beyond NP : Arc-consistency for quantified constraints. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 371–386, 2002.

[8] M. Cadoli, A. Giovanardi, and M. Schaerf. Experimental Analysis of the Computational Cost of Evaluating Quantified Boolean Formulae. In *Proceedings of the 5th Conference of the Italian Association for Artificial Intelligence (AIIA'97)*, pages 207–218, 1997.

[9] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communication of the ACM*, 5 :394–397, 1962.

[10] U. Egly, M. Seidl, and S. Woltran. A solver for QBFs in negation normal form. *Constraints*, 14(1) :38–79, 2009.

[11] I.P Gent, P. Nightingale, A. Rowley, and K. Stergiou. Solving quantified constraint satisfaction problems. *Journal of Artificial Intelligence*, 172(6-7) :738–771, 2008.

[12] E. Giunchiglia, M. Maratea, A. Tacchella, and D. Zambonin. Evaluating search heuristics and optimization techniques in propositional satisfiability. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01)*, pages 347–363, 2001.

[13] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `www.qbflib.org`.

[14] 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, 2006.

[15] A. Goultiaeva, V. Iverson, and F. Bacchus. A Compact Representation for Syntactic Dependencies in QBFs. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 412–426, 2009.

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

[17] F. Lonsing and A. Biere. Nenofex : Expanding NNF for QBF Solving. In *Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT'08)*, pages 196–210, 2008.

[18] G. Pan and M.Y. Vardi. Symbolic Decision Procedures for QBF. In *International Conference on Principles and Practice of Constraint Programming*, 2004.

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

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

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

[22] C. Schulte and G. Tack. Views and Iterators for Generic Constraint Implementations. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, pages 817–821, 2005.

[23] M. Sheeran and G. 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.

[24] I. Stéphan. Boolean Propagation Based on Literals for Quantified Boolean Formulae. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, pages 452–456, 2006.

[25] I. Stéphan. Sémantique et calcul syntaxique pour les formules booléennes quantifiées. In *Actes des Quatrièmes Journées Francophones de Programmation par Contraintes (JFPC'09)*, pages 275–284, 2009.

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

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

[28] L. Zhang. Solving QBF with combined conjunctive and disjunctive normal form. In *National Conference on Artificial Intelligence (AAAI'06)*, 2006.