

# Symétries et formules booléennes quantifiées

---

Gilles Audemard      Bertrand Mazure  
Lakdhar Saïs

CRIL CNRS – Université d'Artois  
rue Jean Souvraz SP-18 F-62307 Lens Cedex France  
email : {audemard,mazure,sais}@cril.univ-artois.fr

## Résumé

De nombreuses tâches et problèmes combinatoires contiennent des symétries. La résolution de tels problèmes conduit à répéter inlassablement l'étude de situations ou de sous-problèmes équivalents. Depuis plusieurs années, l'exploitation des symétries a permis une réduction significative de l'espace de recherche. Ce paradigme important a été étudié de manière extensive dans de nombreux domaines, comme les problèmes de satisfaction de contraintes (CSP) ou la satisfiabilité de formules booléennes (SAT). Dans cet article, nous montrons comment les symétries peuvent naturellement être étendues aux formules booléennes quantifiées (QBF). Étendant l'approche introduite par Aloul *et al.* pour les formules booléennes, un algorithme de détection des symétries est proposé. Un nouveau solveur hybride opérant simultanément sur la QBF initiale et les contraintes de symétrie (« *Symmetry Breaking Predicates* ») est ensuite proposé. Les expérimentations réalisées montrent qu'un grand nombre d'instances utilisées lors de l'évaluation QBF'03 contiennent des symétries. Leur exploitation a permis une amélioration des solveurs QBF sur certaines classes d'instances QBFs.

## 1 Introduction

Depuis quelques années, la résolution des formules booléennes quantifiées (QBF) est un domaine de recherche en pleine ébullition. Cet intérêt peut être expliqué par plusieurs facteurs. D'une part de nombreux problèmes en intelligence artificielle (planification, raisonnement non monotone, vérification formelle, ...) peuvent se réduire à des formules booléennes quantifiées. De l'autre, les progrès incessants réalisés sur la résolution du problème SAT permettent maintenant de s'attaquer à des formules de plus en plus grandes et difficiles. C'est ainsi que dernièrement, de nombreux solveurs pour résoudre des instances QBF ont été proposés (e.g. [GNT01, ZM02, Let02]). La plupart d'entre eux, étendent les derniers résultats obtenus sur la résolution du problème SAT. Ceci n'a rien de surprenant puisque les formules QBF sont une extension naturelle du problème SAT.

Certaines classes d’instances QBF encodent des problèmes « du monde réel » et contiennent de nombreuses symétries. On peut espérer une réduction de l’espace de recherche en supprimant ces symétries. En effet, l’exploitation des symétries a déjà montré son importance pour résoudre des problèmes combinatoires intraitables jusqu’alors. Ceci a par exemple été le cas pour les problèmes de satisfaction de contraintes (e.g. [Pug02, GS00]) et pour le problème SAT (e.g. [ARMS02b, CGLR96, BS94]).

Dans cet article, nous montrons comment les symétries peuvent naturellement être étendues aux formules booléennes quantifiées. Un algorithme de détection des symétries est proposé. Il étend l’algorithme de détection proposé par Aloul *et al.* pour le problème SAT [ARMS02a, ARMS03]. Les expérimentations conduites sur les instances de l’évaluation QBF’03 [BST03], montrent qu’une grande partie de celles-ci, les instances non générées aléatoirement, contiennent des symétries. Par conséquent, nous proposons également une première approche exploitant ces symétries. Elle est basée sur un algorithme hybride résolvant simultanément une instance QBF et une instance SAT issue des prédicats brisant les symétries (« *Symmetry breaking predicates* », SBP en anglais [CGLR96]). Le reste de cet article est organisé comme suit. Après des définitions et notations préliminaires, les symétries des formules booléennes quantifiées sont introduites. Il est ensuite montré comment on peut détecter ces symétries en étendant naturellement l’algorithme proposé par Aloul *et al.* pour le problème SAT [ARMS02a]. Une première approche exploitant les symétries est décrite. Une expérimentation basée sur les instances de la dernière évaluation QBF est alors réalisée montrant que de nombreuses symétries sont détectées. Pour terminer, des futures extensions et applications de ces travaux sont discutées avant la conclusion.

## 2 Préliminaires et Définitions

Soit  $\mathcal{P}$  un ensemble de variables propositionnelles.  $\mathcal{L}_{\mathcal{P}}$  désigne alors le langage des formules booléennes quantifiées construites à partir de  $\mathcal{P}$  en utilisant les formules booléennes classiques (incluant les constantes  $\top$  et  $\perp$ ) ainsi que les quantifications  $\exists$  et  $\forall$ .

On considère les formules booléennes quantifiées mise sous forme préfixe :  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$  (ou encore  $QX\Psi$ ,  $QX$  est le préfixe de  $\Phi$  et  $\Psi$  sa matrice) tel que  $Q_i \in \{\exists, \forall\}$ ,  $X_k, \dots, X_1$  sont des ensembles disjoints de variables et  $\Psi$  une formule booléenne. Les variables consécutives ayant le même quantificateur sont groupées. Une QBF  $\Phi$  est dite sous forme normale si  $\Phi$  est sous forme préfixe et si  $\Psi$  est sous forme normale conjonctive (CNF). Nous définissons  $Var(\Phi) = \bigcup_{i \in \{1, \dots, k\}} X_i$  l’ensemble des variables de  $\Phi$ . Un littéral est une variable propositionnelle sous forme positive ( $l$ ) ou négative ( $\neg l$ ).  $Lit(\Phi) = \bigcup_{i \in \{1, \dots, k\}} Lit(X_i)$  est l’ensemble complet des littéraux de  $\Phi$ , où  $Lit(X_i) = \{x_i, \neg x_i \mid x_i \in X_i\}$ .

Nous introduisons quelques notations nécessaires à la compréhension de l’article. Soit  $S$  l’ensemble des affectations possibles basées sur l’ensemble des variables  $V$ . La Uprojection (resp. la Dprojection) d’un ensemble d’affectation  $S$  sur l’ensemble des variables de  $X \subset V$ , notée  $S \uparrow X$  (resp.  $S \downarrow X$ ), est obtenue en restreignant chaque affectation de  $S$  aux littéraux de  $X$  (resp. dans  $V \setminus X$ ). L’ensemble des affectations possibles sur  $X$  est notée  $2^X$ . Une affectation sur  $X$  est notée par le vecteur  $\vec{x}$ . De même, Uprojection et Dprojection peuvent également s’appliquer sur un vecteur de littéraux  $\vec{x}$ . Si  $\vec{y}$  est une affectation sur  $Y$  tel que  $Y \cap X = \emptyset$ , alors  $\vec{y}.S$  désigne l’ensemble des interprétations obtenues en concaténant  $\vec{y}$  avec chaque interprétation de  $S$ . Finalement,  $\Psi(\vec{x})$  désigne

la formule  $\Psi$  simplifiée par l'affectation partielle  $\vec{x}$ .

Une QBF est valide (vraie) s'il existe une solution, appelée *police totale* définie comme suit. Cette définition est une version simplifiée de celle apparaissant dans [MFL<sup>+</sup>02].

**Définition 1** Soient  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$  une formule booléenne quantifiée et  $\pi = \{m_1, \dots, m_n\}$  un ensemble de modèles de la formule booléenne  $\Psi$ .  $\pi$  est une police totale de la QBF  $\Phi$  si et seulement si  $\pi$  vérifie récursivement les conditions suivantes conditions :

1.  $k = 0$ , et  $\Psi = \top$
2. Si  $Q_k = \forall$ , alors  $\pi \uparrow X_k = 2^{X_k}$ , et  $\forall \vec{x}_k \in 2^{X_k}$ ,  $\pi \downarrow \vec{x}_k$  est une police totale de  $Q_{k-1} X_{k-1}, \dots, Q_1 X_1 \Psi(\vec{x}_k)$
3. Si  $Q_k = \exists$ , alors  $\pi \uparrow \vec{X}_k = \{\vec{x}_k\}$  et  $\pi \downarrow \vec{x}_k$  est une police totale de  $Q_{k-1} X_{k-1}, \dots, Q_1 X_1 \Psi(\vec{x}_k)$

**Remarque 1** Soit  $\pi$  une police totale de  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$ . Si  $Q_k = \forall$  alors on peut réécrire  $\pi$  de la manière suivante  $\bigcup_{\vec{x}_k \in 2^{X_k}} \{\vec{x}_k.(\pi \downarrow \vec{x}_k)\}$  et si  $Q_k = \exists$ , alors  $\pi \uparrow X_k = \{\vec{x}_k\}$  et  $\pi$  peut être réécrit comme  $\{\vec{x}_k.(\pi \downarrow \vec{x}_k)\}$

### Exemple 1

Soit  $\Phi = \exists x_5 x_6 \forall x_2 x_4 \exists x_1 x_3 \Psi$  une formule QBF, où  $\Psi = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_4 \vee x_3) \wedge (x_5 \vee x_6)$ .  $\Phi$  est valide et  $\pi = \{(\neg x_5, x_6, x_2, x_4, \neg x_1, x_3), (\neg x_5, x_6, x_2, \neg x_4, \neg x_1, x_3), (\neg x_5, x_6, \neg x_2, \neg x_4, x_1, \neg x_3), (\neg x_5, x_6, \neg x_2, x_4, x_1, x_3)\}$  est une police totale de  $\Phi$  (illustré dans figure 1). Les différents opérateurs de projection sont illustrés ci dessous :

- $\pi \uparrow \{x_5, x_6\} = \{(\neg x_5, x_6)\}$
- $\pi' = \pi \downarrow \{x_5, x_6\} = \{(x_2, x_4, \neg x_1, x_3), (x_2, \neg x_4, \neg x_1, x_3), (\neg x_2, \neg x_4, x_1, \neg x_3), (\neg x_2, x_4, x_1, x_3)\}$
- $\pi' \uparrow \{x_2, x_4\} = \{(\neg x_2, \neg x_4), (\neg x_2, x_4), (x_2, \neg x_4), (x_2, x_4)\} = 2^{\{x_2, x_4\}}$

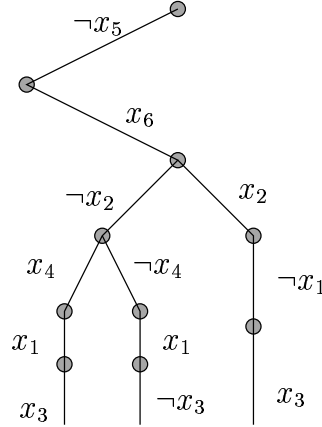


FIG. 1 – Représentation arborescente d'une police (exemple 1)

### 3 Symétries et Formules Booléennes Quantifiées

Soient  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$  une formule booléenne quantifiée et  $\sigma : Lit(\Phi) \mapsto Lit(\Phi)$  une permutation sur les littéraux de  $\Phi$ . Il est possible d'étendre la définition de la permutation  $\sigma$  à  $\Phi$  de la manière suivante :  $\sigma(\Phi) = Q_k \sigma(X_k), \dots, Q_1 \sigma(X_1) \sigma(\Psi)$ . Par exemple, si  $\Psi$  est sous forme clausale alors  $\sigma(\Psi) = \{\sigma(c)/c \in \Psi\}$  et  $\sigma(c) = \{\sigma(l)/l \in c\}$ .

**Définition 2** Soit  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$  une formule booléenne quantifiée. Soit  $\sigma$  une permutation sur les littéraux de  $\Phi$ .  $\sigma$  est une symétrie de  $\Phi$  si et seulement si :

- $\forall x \in Lit(\Phi), \sigma(\neg x) = \neg \sigma(x)$
- $\sigma(\Phi) = \Phi$  i.e  $\sigma(\Psi) = \Psi$  et  $\forall i \in \{1, \dots, k\} \sigma(X_i) = X_i$ .

Notons que chaque symétrie  $\sigma$  d'une QBF  $\Phi$  est également une symétrie de la formule booléenne  $\Psi$ . Mais, La réciproque n'est pas vraie. Donc l'ensemble des symétries de  $\Phi$  est un sous-ensemble des symétries de  $\Psi$ .

**Définition 3** Soient  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$  une formule booléenne quantifiée,  $\sigma$  une symétrie de  $\Phi$  et  $\pi = \{m_1, \dots, m_n\}$  une police totale de  $\Phi$ , on définit  $\sigma(\pi) = \{\sigma(m_i)/m_i \in \pi\}$  et  $\sigma(m_i) = \{\sigma(l)/l \in m_i\}$ .

**Proposition 1** Soient  $\sigma$  une symétrie de la formule booléenne quantifiée  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$ ,  $\pi$  un ensemble d'affectations sur  $Var(\Phi)$ .  $\pi$  est une police totale de  $\Phi$  si et seulement si  $\sigma(\pi)$  est une police totale de  $\Phi$ .

*Preuve* Par induction sur  $k$ , nous montrons que si  $\pi$  est une police totale alors  $\sigma(\pi)$  est également une police totale.

For  $k = 1$ , on considère uniquement le cas où  $Q_1 = \exists$  (le premier groupe de quantificateur ne peut être universel, autrement  $\Phi$  n'est pas valide et  $\pi$  est vide). Dans ce cas, toutes les variables de  $\Phi$  sont quantifiées existentiellement et nous nous retrouvons dans le cas propositionnel de base, donc  $\sigma(\pi)$  est également un ensemble de modèles.

Supposons maintenant que la proposition est vraie pour le rang  $k-1$ , montrons qu'elle est également vraie pour  $k$ .  $\pi$  est une police totale de  $\Phi = Q_k X_k, \dots, Q_1 X_1 \Psi$ , deux cas sont à considérer :

1.  $Q_k = \forall$  : par définition des polices totales  $\forall \vec{x}_k \in 2^{X_k}$  ( $\pi \downarrow \vec{x}_k$ ) est également une police totale de  $Q_{k-1} X_{k-1}, \dots, Q_1 X_1 \Psi(\vec{x}_k)$ . Par hypothèse,  $\sigma(\pi \downarrow \vec{x}_k)$  est une police totale de  $Q_{k-1} X_{k-1}, \dots, Q_1 X_1 \Psi(\vec{x}_k)$ .  $\forall \vec{x}_k \in 2^{X_k}$ , nous avons donc également  $\sigma(\vec{x}_k) \in 2^{X_k}$ . Par conséquent,  $\bigcup_{\vec{x}_k} \{\sigma(\vec{x}_k). \sigma(\pi \downarrow \vec{x}_k)\} = \sigma(\pi)$  est une police totale de  $\Phi$ .
2.  $Q_k = \exists$  : ( $\pi \uparrow X_k$ ) =  $\{\vec{x}_k\}$  et  $\pi \downarrow \vec{x}_k$  est une police totale de  $Q_{k-1} X_{k-1}, \dots, Q_1 X_1 \Psi(\vec{x}_k)$ . En utilisant l'hypothèse d'induction,  $\sigma(\pi \downarrow \vec{x}_k)$  est également une police totale. Donc,  $\sigma(\pi) = \sigma(\vec{x}_k). \sigma(\pi \downarrow \vec{x}_k)$  est une police totale de  $\Phi$ .

La réciproque peut être montrée en utilisant  $\sigma^{-1}$ .

#### Exemple 2

Soit  $\Phi = \forall x_1 x_2 \exists x_3 x_4 \Psi$  une QBF, où

$$\Psi = \underbrace{(\neg x_1 \vee \neg x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee x_4)}_{c_2} \wedge \underbrace{(x_1 \vee \neg x_3 \vee \neg x_4)}_{c_3} \wedge \underbrace{(x_2 \vee \neg x_3 \vee \neg x_4)}_{c_4}$$

Les permutation  $\sigma_1 = (x_1, x_2)(x_3)(x_4)$  et  $\sigma_2 = (x_1)(x_2)(x_3, x_4)$  sont des symétries de  $\Psi$ . Dans la suite les littéraux qui se permutent avec eux-mêmes seront omis (comme  $x_3$  et  $x_4$  dans  $\sigma_1$ ).  $\Phi$  est une formule valide i.e.  $\pi = \{(\neg x_1, \neg x_2, \neg x_3, \neg x_4), (\neg x_1, x_2, \neg x_3, x_4), (x_1, \neg x_2, x_3, \neg x_4), (x_1, x_2, x_3, x_4)\}$  est une police totale de  $\Phi$ . On peut vérifier que  $\sigma_1(\pi) = \{(\neg x_2, \neg x_1, \neg x_3, \neg x_4), (\neg x_2, x_1, \neg x_3, x_4), (x_2, \neg x_1, x_3, \neg x_4), (x_2, x_1, x_3, x_4)\}$  et  $\sigma_2(\pi) = \{(\neg x_1, \neg x_2, \neg x_4, \neg x_3), (\neg x_1, x_2, \neg x_4, x_3), (x_1, \neg x_2, x_4, \neg x_3), (x_1, x_2, x_4, x_3)\}$  sont également des polices totales de  $\Phi$ .

Dans le contexte des formules booléennes, les symétries peuvent être vues comme une relation d'équivalence entre les ensembles des interprétations de la formule booléenne, induisant des classes d'équivalence où chaque représentant est un modèle ou non. Les symétries sur les formules booléennes quantifiées étendent cette relation d'équivalence à des « super-ensembles » d'interprétations, chaque classe d'équivalence contenant ou non des polices totales.

## 4 Détection

Depuis quelques années, différentes techniques pour détecter les symétries dans les formules booléennes ont été proposées. Certaines d'entre elles, opèrent la détection directement sur la formule booléenne [BS94]. D'autres techniques habituellement utilisées consistent à réduire le problème de la détection des symétries à celui de recherche d'isomorphisme de graphes [Cra92, CGLR96] (problème consistant à trouver un « mapping » entre deux graphes  $G$  et  $H$ ). La complexité du problème d'isomorphisme de graphes est un problème ouvert (il est communément admis qu'il est à la frontière entre les classes P et NP [Cra92]). Dans notre contexte, nous nous intéressons au cas particulier d'*automorphisme* de graphes (trouver un « mapping » entre  $G$  et  $G$ ). Beaucoup de programmes ont été proposés pour résoudre ce problème. Mentionnons tout particulièrement NAUTY [McK90] qui est l'un des plus efficace en pratique.

Récemment, Aloul *et al.* [ARMS02a] ont proposé une technique intéressante transformant une formule QBF  $\Psi$  en un graphe  $G_\Psi$  contenant des sommets colorés. Bien entendu, ces couleurs sont prises en compte lors de la recherche d'automorphisme de graphes (i.e. des sommets ne peuvent pas être permutés avec des sommets d'une couleur différente). La formule CNF est transformée en graphe de la manière suivante (voir l'exemple 3) :

- Chaque variable est représentée par deux sommets, un pour le littéral positif, le second pour le littéral négatif. Une couleur gris foncé est donnée à ces sommets.
- Une arête relie deux littéraux opposés.
- Toute clause non binaire est représentée par un sommet de couleur gris clair elle est reliée à tous les littéraux qui la compose.
- Toute clause binaire  $l_1 \vee l_2$  est représentée par une double arête entre les sommets  $l_1$  et  $l_2$ . Ceci afin de réduire le nombre de sommets nécessaires.

### Exemple 3

Soit  $\Psi$  la formule CNF de l'exemple 2. La figure 2.a donne le graphe associé  $G_\Psi = (V, E)$ . L'ensemble des sommets  $V$  contient  $2 \times |\text{Var}(\Psi)| = 8$  sommets (couleur gris foncé) associés aux littéraux de  $\text{Lit}(\Psi)$  et 4 sommets (couleur gris clair) correspondant aux clauses de  $\Psi$ . L'ensemble des arêtes  $E$  relie les littéraux avec leurs opposés (e.g.  $(x_1, \neg x_1)$  et les clauses aux littéraux qui les composent (i.e.  $(c_1, \neg x_1)$ ,  $(c_1, \neg x_2)$  and  $(c_1, x_3)$ ).

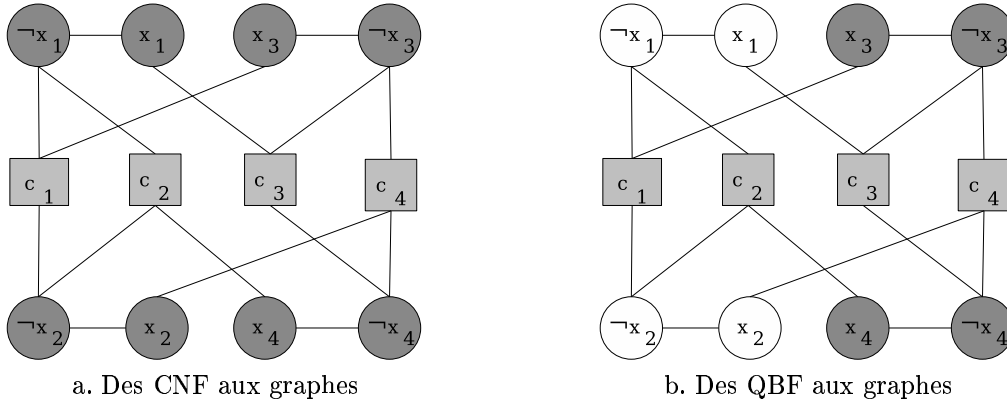


FIG. 2 – Réduction en graphe

En utilisant NAUTY [McK90], trois automorphismes laissant invariant  $G_\Psi$  sont calculés :  $a_1 = (x_1, x_2)(x_3, x_4)[(c_1)(c_2)(c_3, c_4)]$ ,  $a_2 = (x_1)(x_2)(x_3, x_4)[(c_1, c_2)(c_3)(c_4)]$ , et  $a_3 = (x_1, x_3)(x_2, x_4)[(c_1, c_3)(c_2, c_4)]$ . Les symétries correspondantes  $\sigma_1$ ,  $\sigma_2$  et  $\sigma_3$  sont obtenues respectivement de  $a_1$ ,  $a_2$  et  $a_3$  par projection sur les littéraux de  $\Psi$ .

Il est relativement aisé d'étendre cette transformation aux formules QBF. En effet, il nous faut considérer le préfixe de la QBF (voir la deuxième condition de la définition 2), puisque deux littéraux ne peuvent pas être symétriques s'il n'appartiennent pas au même groupe de quantificateurs. Dans cette optique, nous avons juste besoin de rajouter des couleurs additionnelles pour faire la distinction entre les sommets (des littéraux) des différents groupes de quantificateurs. Pour une formule QBF contenant  $k$  groupes de quantificateurs, nous introduisons donc  $k$  couleurs différentes. Ainsi, les symétries des QBFs sont détectées sur le graphe résultant en utilisant NAUTY. Nous illustrons cette transformation étendue dans l'exemple suivant :

#### Exemple 4

Soit  $\Phi = \forall x_1, x_2 \exists x_3 x_4 \Psi$  la formule QBF de l'exemple 2. La figure 2.b donne le graphe associé  $G_\Phi$ . Ici, nous avons besoin de trois couleurs, une pour les clauses non binaires (gris clair), une pour le premier groupe de variables universelles (blanche), et enfin, la dernière pour le groupe de variables existentielles (gris foncé). Cette formule QBF  $\Phi$  possède deux symétries non triviales  $(x_1, x_2)$  et  $(x_3, x_4)$ . La symétrie  $(x_1, x_3)(x_2, x_4)$  de  $\Psi$  (voir exemple 3) n'est pas une symétrie de  $\Phi$ , puisque  $x_1$  et  $x_3$  n'appartiennent pas au même groupe de quantificateur.

## 5 Exploiter les symétries dans un solveur QBF

Les suppressions des symétries ont été particulièrement étudiées sur les problèmes de satisfaction de contraintes et sur le problème de satisfaisabilité. Les différentes approches proposées pour casser les symétries peuvent être classifiées en deux catégories : les approches dynamiques d'une part et les approches statiques d'autre part. Généralement, on parle d'approche dynamique lorsque les symétries sont détectées et supprimées

durant la recherche de la solution du problème initial. Par exemple, [BS94] ou encore [GS00] ajoutent (dynamiquement) des prédicats cassant les symétries. L'atout majeur de ces approches vient du fait que des symétries locales sont détectées et supprimées (symétries apparaissant durant la recherche sur des sous-problèmes du problème initial). À l'inverse, l'approche statique se réfère aux techniques détectant les symétries dans une phase de prétraitement, les symétries sont alors généralement supprimées en ajoutant des contraintes appelées en anglais « *Symmetry Breaking Predicates (SBP)* » [Cra92, ARMS02a]. Ces SBP éliminent tous les modèles isomorphes d'une classe d'équivalence, excepté un, le représentant. L'inconvénient majeur, vient du fait que dans le cas général, l'ensemble des prédicats ajoutés peut avoir une taille exponentielle. Récemment, Aloul *et al* [ARMS02a, ARMS03] ont étendu cette approche, initialement proposé par [Cra92] en utilisant l'algèbre des groupes et le concept de générateur non redondant, réduisant ainsi considérablement la taille du SBP.

Dans le contexte des formules booléennes quantifiées, et contrairement à l'approche booléenne, les prédicats cassant les symétries ne peuvent pas être ajoutés à la matrice. En effet, on peut obtenir des clauses ne contenant que des variables universellement quantifiées et rendre ainsi la formule non valide, comme le montre l'exemple 5.

### Exemple 5

Reprenons la QBF  $\Phi$  de l'exemple 2. Une manière de « casser » la symétrie  $\sigma_1$  est d'ajouter à  $\Psi$  la clause  $(\neg x_1 \vee x_2)$ . Cette clause étant constituée uniquement de variables universellement quantifiées, la nouvelle QBF obtenue en ajoutant cette clause à la matrice de  $\Phi$  est non valide.

Afin d'éviter ce problème, nous proposons une première approche où les prédicats SBP, générés suivant la méthode de Aloul *et al* [ARMS02a], sont considérés comme une formule booléenne indépendante de la formule QBF initiale. Un solveur hybride permet alors de traiter simultanément la formule SBP de la formule QBF. La figure 3 schématise cette méthode hybride. Nous avons implémenté cette approche à partir de la plate-forme OPENQBF. Cette plate-forme, développée au CRIL en Java, a participé aux évaluations QBF de 2003 et 2004 et est un solveur étendant la procédure de Davis et Putnam aux formules QBF mises sous forme normale.

Le solveur hybride est donc constitué d'un solveur QBF et d'un mécanisme de propagation unitaire sur une formule booléenne (SBP). À chaque étape du solveur QBF, nous invoquons le solveur SAT sur la formule SBP afin de propager les littéraux unitaires. Si  $x$  est la variable courante à assigner dans le solveur QBF, cette variable est également affectée dans le SBP à l'aide du solveur SAT. De manière réciproque, si  $y$  est un littéral déduit de la formule SBP, alors deux cas peuvent se présenter :

- Soit  $y$  est universellement quantifié dans la formule QBF, alors la branche correspondant à  $\neg y$  est considérée comme vrai et le solveur QBF propage  $y$  pour affirmer cette hypothèse (si un modèle est trouvé avec  $y$  alors il existe un modèle avec  $\neg y$ , autrement un conflit apparaît puisque  $y$  est universellement quantifié).
- Soit  $y$  est existentiellement quantifié, alors la branche correspondant à  $\neg y$  est considérée comme inconsistante. Dans le même esprit,  $y$  est alors propagé dans le solveur QBF.

La figure 3 résume cette approche hybride. Le solveur QBF contentant des symétries prend une formule QBF normale en entrée. Une étape de prétraitement est alors réalisé sur la formule QBF afin de déterminer les SBP, une version modifiée de SHATTER

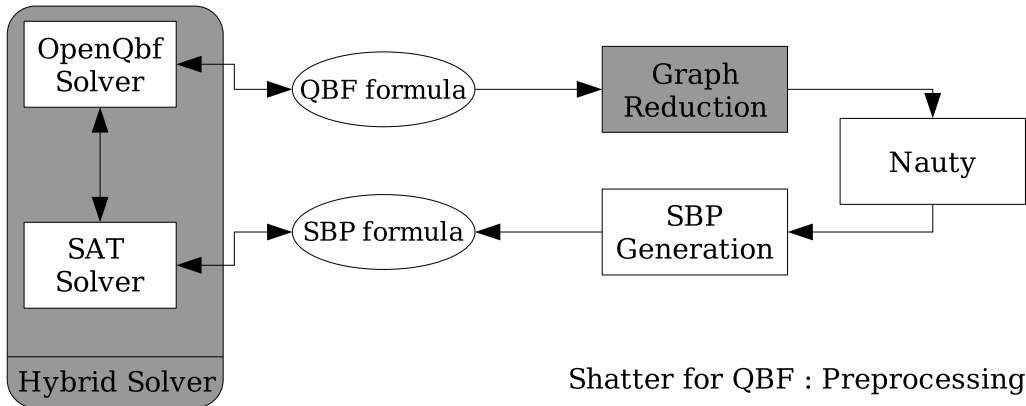


FIG. 3 – Solveur QBF cassant des symétries<sup>1</sup>

[ARMS03, ARMS02a] effectue cette étape. Le solveur hybride, combinaison d’un solveur QBF et d’un mécanisme de propagation SAT, opère alors sur la formule QBF originale et sur les SBP.

## 6 Expérimentations préliminaires

Nous présentons les premières expérimentations qui ont été réalisées sur un Pentium IV 3 GHz avec 1Go de RAM, dans le but de détecter et d’exploiter les symétries dans les formules booléennes quantifiées. Comme dit dans la section précédente, une version modifiée de SHATTER [ARMS03, ARMS02a] est utilisée lors de la phase de détection.

Dans les tables 1.a et 1.b, nous présentons les résultats de la phase de la détection des symétries (voir la section 4) sur les instances de l’évaluation QBF’03 [BST03]. L’évaluation QBF’03 a été réalisée à partir de 1350 instances QBF classifiées selon leur difficulté par rapport aux solveurs en présence, les catégories Easy, Medium et Hard (toujours non résolus). Au moins 729 sur ces 1350 instances ont été générées selon une méthode aléatoire. La table 1.a présente le nombre d’instances symétriques (la colonne #P donne le nombre d’instances, #S le nombre d’instances symétriques et % le pourcentage d’instances symétriques) sur la collection complète de l’évaluation QBF’03. Une grande partie des instances contiennent des symétries (38.5 %). Comme on peut le constater, les problèmes faciles contiennent dans une grande majorité des symétries (de l’ordre de 80%). Alors que cela n’est pas vrai pour la catégorie des instances medium ( $\approx 30\%$ ). Il est à noter que cette catégorie est largement composée d’instances aléatoires qui ne contiennent que très rarement des symétries (voir la table 1.b). Plus intéressant, presque la moitié des instances difficiles, toujours ouvertes, contiennent des symétries. Casser les symétries sera probablement une voie pour résoudre ces instances.

La table 2 résume la première comparaison expérimentale entre notre solveur hybride (voir la figure 3) qui détecte et exploite les symétries et le solveur OPENQBF qui est une simple extension de la procédure de DP pour les instances QBF. La comparaison est faite sur le nombre de noeuds (#N) parcourus et sur le temps cpu (Tps) nécessaire

<sup>1</sup>La partie grise de la figure correspond à notre contribution.

Catégorie	#P	#S	% (#S/#P)
Easy	178	144	80.9
Medium	1030	308	29.9
Hard	142	68	47.9
Total	1350	520	38.5

a. Instances QBF03 : de Facile à Dure

Catégorie	#P	#S	%
Aléatoire	729	86	11.8
Non Aléatoire	621	434	69.9

b. Instances QBF03 : Aléatoire vs Non Aléatoire

TAB. 1 – Détection des Symétries sur les instances de QBF’03

		Détection		OpenQBF + Symétries		OpenQBF	
instance	Valide	Tps	#Sym	Tps	#N	Tps	#N
BLOCKS3ii.4.3	Non	0.02	5.7e9	78	130 521	77.6	130 521
TOILET6.1.iv.12	Oui	0.02	20 160	0.2	14	119	366 467
CHAIN17v18	Oui	0.58	3.43e11	69	65 794	227	131 105

TAB. 2 – OpenQBF avec et sans symétries

(en secondes) pour résoudre l’instance. De plus, le nombre de symétries (#Sym) et le temps nécessaire à leur calcul est également fourni. Nous pouvons constater que le calcul des symétries est très rapide (moins d’une seconde). L’instance *BLOCKS3ii.4.3* issue de la planification contient beaucoup de symétries. Néanmoins, ces symétries ne sont pas extrêmement intéressantes puisque se trouvant dans le dernier groupe de quantificateurs, par conséquent les branches coupées par les symétries se situent dans la partie basse de l’arbre de recherche. Des résultats très intéressants sont obtenus sur les instances *TOILET6.1.iv.12* et *CHAIN17v18* où les variables symétriques appartiennent à tous les groupes de quantificateurs. La suppression de ces symétries apporte une réduction conséquente de l’arbre de recherche.

## 7 Conclusion et Recherches futures

Nous avons montré comment étendre la notion de symétries sur les formules booléennes quantifiées. Un algorithme de détection des symétries des formules booléennes quantifiées est donné, étendant l’approche proposée par [Cra92, ARMS03] pour les formules booléennes (CNF). Nous avons montré comment les symétries détectées peuvent ensuite être supprimées par l’intermédiaire d’un solveur hybride qui travaille simultanément sur la formule QBF de départ et sur les « *Symmetry Breaking Predicates* » ajoutés.

Des résultats préliminaires montrent l’intérêt de notre approche en terme de symétries

détectées et du gain obtenu sur certaines classes d'instances. Les symétries d'une formule QBF forment un sous-ensemble des symétries d'une formule booléenne, une perspective intéressante à étudier concerne l'exploitation des symétries perdues par l'introduction des quantificateurs. Nous envisageons également d'étudier la possibilité d'intégrer directement les SBP dans la formule booléenne quantifiée.

## Références

- [ARMS02a] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. Technical Report CSE-TR-463-02, University of Michigan, 2002.
- [ARMS02b] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Solving difficult SAT instances in the presence of symmetry. In *Proceedings of the 39th Design Automation Conference (DAC 2002)*, pages 731–736. ACM Press, 2002.
- [ARMS03] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Shatter : Efficient breaking for boolean satisfiability. In *proceedings of the international conference on Design Automation Conference (DAC)*, pages 836–839. ACM Press, 2003.
- [BS94] Belaid Benhamou and Lakhdar Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1) :89–102, February 1994.
- [BST03] Daniel Le Berre, Laurent Simon, and Armando Tacchella. Challenges in the qbf arena : the sat'03 evaluation of qbf solvers. In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT2003)*, volume 2919 of *LNAI*, pages 452–467, 2003.
- [CGLR96] James Crawford, Matthew L. Ginsberg, Eugene Luck, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *KR'96 : Principles of Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann, San Francisco, California, 1996.
- [Cra92] James Crawford. A theoretical analysis of reasoning by symmetry in first order logic. In *Proceedings of Workshop on Tractable Reasoning, AAAI92*, pages 17–22, July 1992.
- [GNT01] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. QuBE : A system for deciding Quantified Boolean Formulas Satisfiability. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'01)*, Siena, Italy, June 2001.
- [GS00] Ian P. Gent and Barbara Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proceedings of European Conference on Artificial Intelligence ECAI-2000*, pages 599–603. IOS Press, 2000.
- [Let02] Reinhold Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proceedings of Tableaux 2002*, pages 160–175, Copenhagen, Denmark, 2002.

- [McK90] Brendan D. McKay. nauty user's guide (version 1.5). Technical Report TR-CS90 -02, Computer Science Department, Australian National University, 1990.
- [MFL<sup>+</sup>02] Sylvie Coste Marquis, Helene Fargier, Jerome Lang, Daniel Le Berre, and Pierre Marquis. Résolution de formules booléennes quantifiées : problèmes et algorithmes. In *Actes du 13eme congrès AFRIF-AFIA Reconnaissance des forme et intelligence artificielle (RFIA)*, pages 289–298, 2002.
- [Pug02] Jean François Puget. Symmetry breaking revisited. In *SymCon'02 - Symmetry in Constraints - CP02 Workshop*, 2002.
- [ZM02] Lintao Zhang and Sharad Malik. Towards a symmetric treatment of satisfaction and conflicts in quantified boolean formula evaluation. In *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 200–215, Ithaca, NY, USA, Sept. 2002.