Dealing with symmetries in Quantified Boolean Formulas

Gilles Audemard, Bertrand Mazure, and Lakhdar Saïs

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

Abstract. Many reasoning task and combinatorial problems exhibit symmetries. Exploiting symmetries has been proved very important in reducing search efforts. This important task is widely investigated in constraint satisfaction problems and satisfiability of boolean formulas. In this paper, we show how symmetries can be naturally extended to Quantified Boolean Formulas (QBFs). A symmetries detection algorithm is given, extending the CNF approach proposed by Aloul *et al.* A new hybrid solver that handle QBFs and Symmetry Breaking predicates is then proposed. Experiments, conducted on instances from the last competition on QBFs, show that many of them contains symmetries. Breaking such symmetries lead to interesting improvements of QBFs solver on certain class of instances.

1 Introduction

Solving Quantified Boolean Formulas (QBF) has become an attractive and important research area over the last years. Such increasing interest, might be related to different factors, including the fact that many important artificial intelligence (AI) problems (planning, non monotonic reasoning, formal verification, etc.) can be reduced to QBFs which is considered as the canonical problem of the PSPACE complexity class. Another important reason comes from the recent impressive progress reached in the practical resolution of the satisfiability problem. Many solvers for QBFs has been proposed recently (e.g. [7, 12, 9]), most of them are obtained by extending satisfiability results. This is not surprising, since QBFs is a natural extension of SAT (satisfiability problem of boolean formula).

Some class of QBFs encoding real-world application and/or AI problems contains many symmetries, exploiting such structures might lead to a great reduction of the search space. Exploiting symmetries is widely investigated and considered as an important task to deal with the intractability of many combinatorial problems such as constraint satisfaction problems (CSP) and satisfiability of boolean formula.

In this paper, we show how symmetries can be naturally extended to Quantified Boolean Formulas. A symmetries detection algorithm is given, extending the SAT-based approach proposed by Aloul *et al.* to QBFs [1, 2]. As expected, experiments on many instances proposed in the last competition on QBFs [8], show that certain classes of instances encoding real-word problems contains many symmetries. Consequently, a first approach exploiting symmetries during search is proposed.

The paper is organised as follows. After some preliminary definitions on quantified boolean formulas, symmetries framework in QBFs is presented. It is then shown how symmetries detection method by Aloul *et al.* can be naturally extended to handle QBFs. Preliminary approach on exploiting such symmetries is then described. Experiments of our detection method on the instances of the last QBFs competition show that our approach detect many symmetries. Exploiting such symmetries in QBF solver leads to interesting improvements with respect to some classes of QBF03 evaluation instances. Finally, promising paths for future research are discussed in the conclusion.

2 Definitions and Preliminaries

Let $\mathcal{P}$ be a finite set of propositional variables. Then, $\mathcal{L}_{\mathcal{P}}$ is the language of quantified Boolean formulas built over $\mathcal{P}$ using ordinary boolean formulas (including propositional constants $\top$ and $\bot$) plus the additional quantification ($\exists$ and $\forall$) over propositional variables.
We consider quantified boolean formula in the prenex form: $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$ (in short $QX\Psi$, QX is called a prefix and Ψ a matrix) where $Q_i \in \{\exists, \forall\}$, $X_k, \ldots, X_1$ are disjoint sets of

variables and Ψ a boolean formula. Consecutive variables with the same quantifier are grouped. A QBF Φ is said to be in clausal form if Φ is in prenex form and Ψ is in Conjunctive Normal Form (CNF). We define $Var(\Phi) = \bigcup_{i\in\{1,\ldots,k\}} X_i$ the set of variables of Φ. A literal is the occurrence of propositional variable in either positive (l) or negative form ($\neg l$). $Lit(\Phi) = \bigcup_{i\in\{1,\ldots,k\}} Lit(X_i)$ the set of complete literals of Φ, where $Lit(X_i) = \{x_i, \neg x_i | x_i \in X_i\}$.

In the sequel, we introduce some necessary notations. Let S be the set of assignments over the set of variables V. The Up-projection (resp. Down-projection) of a set of assignments S on a set of variables $X \subset V$, denoted $S \uparrow X$ (resp. $S \downarrow X$), is obtained by restricting each assignment to literals in X (resp. in $V \backslash X$). The set of all possible assignments over X is denoted by 2^X. An assignment over X is denoted by a set of literals $\overrightarrow{x}$. In the same way, Up-projection and Down-projection also apply on a set of literals $\overrightarrow{x}$. If $\overrightarrow{y}$ is an assignment over Y s.t. $Y \cap X = \emptyset$, then $\overrightarrow{y}.S$ denote the set of interpretations obtained by concatenating $\overrightarrow{y}$ with each interpretation of S. Finally, $\Psi(\overrightarrow{x})$ denote the boolean formula Ψ simplified with the partial assignment $\overrightarrow{x}$.
A QBF is valid (is true) if there exists a solution (called a total policy) defined as follows. It is a simplified version of the definition appeared in [10]

Definition 1. *Let $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$ be a quantified boolean formula and $\pi = \{m_1, \ldots, m_n\}$ a set models of the boolean formula Ψ. π is a total policy of the quantified boolean formula Φ iff π recursively verifies the following conditions:*

1. *$k = 0$, and $\Psi = \top$*
2. *if $Q_k = \forall$, then $\pi \uparrow X_k = 2^{X_k}$, and $\forall \overrightarrow{x}_k \in 2^{X_k}$, $\pi \downarrow \overrightarrow{x}_k$ is a total policy of $Q_{k-1}X_{k-1}, \ldots, Q_1 X_1 \Psi(\overrightarrow{x}_k)$*
3. *if $Q_k = \exists$, then $\pi \uparrow \overrightarrow{X}_k = \{\overrightarrow{x}_k\}$ and $\pi \downarrow \overrightarrow{x}_k$ is a total policy of $Q_{k-1}X_{k-1}, \ldots, Q_1 X_1 \Psi(\overrightarrow{x}_k)$*

Remark 1. Let π be a total policy of $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$. If $Q_k = \forall$ then we can rewrite π as $\bigcup_{\overrightarrow{x}_k \in 2^{X_k}} \{\overrightarrow{x}_k.(\pi \downarrow \overrightarrow{x}_k)\}$ and if $Q_k = \exists$, then $\pi \uparrow X_k = \{\overrightarrow{x}_k\}$ and π can be rewritten as $\{\overrightarrow{x}_k.(\pi \downarrow \overrightarrow{x}_k)\}$

3 Symmetries of Quantified Boolean Formulas

Let $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$ be a quantified boolean formula. Let σ a permutation over the literals of Φ defined as follows $\sigma : Lit(\Phi) \mapsto Lit(\Phi)$. We can extends the definition of the permutation σ to Φ in the following way : $\sigma(\Phi) = Q_k \sigma(X_k), \ldots, Q_1 \sigma(X_1) \sigma(\Psi)$. For example, if Ψ is in clausal form then $\sigma(\Psi) = \{\sigma(c)/c \in \Psi\}$ and $\sigma(c) = \{\sigma(l)/l \in c\}$.

Definition 2. *Let $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$ be a quantified boolean formula. Let σ be a permutation over the literals of Φ. σ is a symmetry of Φ iff*

- *$\sigma(\neg x) = \neg\sigma(x)$ for all literals $x \in Lit(\Phi)$*
- *$\sigma(\Phi) = \Phi$ i.e $\sigma(\Psi) = \Psi$ and $\forall i \in \{1, \ldots, k\} \sigma(X_i) = X_i$.*

Let us note that each symmetry σ of a QBF Φ is also a symmetry of the boolean formula Ψ. The converse is not true. So the set of symmetries of Φ is a subset of the set of symmetries of Ψ.

Definition 3. *Let $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$ be a quantified boolean formula, σ a symmetry of Φ and $\pi = \{m_1, \ldots, m_n\}$ a total policy of Φ, we define $\sigma(\pi) = \{\sigma(m_i)/m_i \in \pi\}$ and $\sigma(m_i) = \{\sigma(l)/l \in m_i\}$.*

Proposition 1. *Let σ be a symmetry of a boolean formula $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$, π is set of assignments over $Var(\Phi)$. π is a policy of Φ, iff $\sigma(\pi)$ is a policy of Φ.*

Proof. By induction on k, we show that if π is a total policy then $\sigma(\pi)$ is also a total policy of Φ. For $k = 1$, we consider only the case where $Q_1 = \exists$ (the first quantifier can not be universal one; otherwise Φ is not valid and π is empty). In such a case all variables of Φ are existentially quantified, then it might be considered as a boolean formula, so $\sigma(\pi)$ is also a set of models (propositional case of symmetry). Now suppose that the proposition holds for $k-1$, let us show that it holds for k. We have π a total policy of $\Phi = Q_k X_k, \ldots, Q_1 X_1 \Psi$, we consider the two following case,

1. $Q_k = \forall$: by definition of total policy $\forall \overrightarrow{x}_k \in 2^{X_k}$ $(\pi \downarrow \overrightarrow{x}_k)$ is a total policy of $Q_{k-1}X_{k-1}, \ldots, Q_1X_1\Psi(\overrightarrow{x}_k)$. By hypothesis, $\sigma(\pi \downarrow \overrightarrow{x}_k)$ is a total policy of $Q_{k-1}X_{k-1}, \ldots, Q_1X_1\Psi(\overrightarrow{x}_k)$. $\forall \overrightarrow{x}_k \in 2^{X_k}$, we also have $\sigma(\overrightarrow{x}_k) \in 2^{X_k}$. Consequently $\bigcup_{\overrightarrow{x}_k}\{\sigma(\overrightarrow{x}_k).\sigma(\pi \downarrow \overrightarrow{x}_k)\} = \sigma(\pi)$ is a total policy of Φ.
2. $Q_k = \exists : (\pi \uparrow X_k) = \{\overrightarrow{x}_k\}$ and $\pi \downarrow \overrightarrow{x}_k$ is a total policy of $Q_{k-1}X_{k-1}, \ldots, Q_1X_1\Psi(\overrightarrow{x}_k)$. Using induction hypothesis $\sigma(\pi \downarrow \overrightarrow{x}_k)$ is also a total policy. Then $\sigma(\pi) = \sigma(\overrightarrow{x}_k).\sigma(\pi \downarrow \overrightarrow{x}_k)$ is a total policy of Φ.

The converse can be proved in the same way using σ^{-1}.

Example 1. Let $\Phi = \forall x_1x_2\exists x_3x_4\ \Psi$ be a QBF, where $\Psi = \underbrace{(\neg x_1 \vee \neg x_2 \vee x_3)}_{c_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee x_4)}_{c_2} \wedge \underbrace{(x_1 \vee \neg x_3 \vee \neg x_4)}_{c_3} \wedge \underbrace{(x_2 \vee \neg x_3 \vee \neg x_4)}_{c_4}$. The permutation $\sigma_1 = (x_1, x_2)(x_3)(x_4)$ and $\sigma_2 = (x_1)(x_2)(x_3, x_4)$ are symmetries of Ψ. In the sequel, self-permuted literals are omitted (like x_3 and x_4 in σ_1). The formula Φ is valid i.e. $\pi = \{(\neg x_1, \neg x_2, \neg x_3, \neg x_4), (\neg x_1, x_2, \neg x_3, x_4), (x_1, \neg x_2, x_3, \neg x_4), (x_1, x_2, x_3, x_4)\}$ is a total policy of Φ. One can verify that $\sigma_1(\pi) = \{(\neg x_2, \neg x_1, \neg x_3, \neg x_4), (\neg x_2, x_1, \neg x_3, x_4), (x_2, \neg x_1, x_3, \neg x_4), (x_2, x_1, x_3, x_4)\}$ and $\sigma_2(\pi) = \{(\neg x_1, \neg x_2, \neg x_4, \neg x_3), (\neg x_1, x_2, \neg x_4, x_3), (x_1, \neg x_2, x_4, \neg x_3), (x_1, x_2, x_4, x_3)\}$ are also total policies of Φ.

In the context of boolean formulas, symmetries can be considered as an equivalence relation on the set of interpretations of the boolean formula that induce equivalence classes, which either contains only models or contains no models. Symmetries on QBF extend such an equivalence relation to superset of interpretations, each equivalence class either contains total policies, or no such policies.

4 Detection

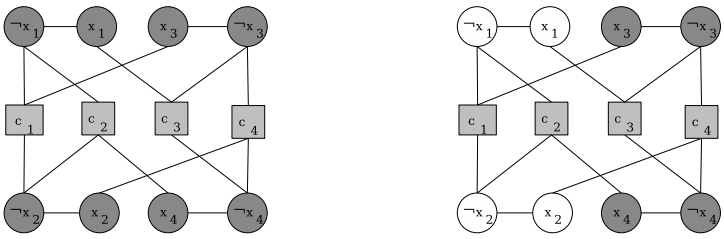
Different techniques for detecting symmetries in boolean formulas have been proposed. Some of them are designed to deal directly with the boolean formula [3]. Other techniques commonly used consists in reducing the problem of symmetry detection into graph isomorphism problem [4, 5] (i.e. problem of finding a one to one mapping between two graphs G and H). The complexity of the graph isomorphism problem is a hard open problem (it is believed to lie between P and NP [4]). In our context, we deal with graph automorphism problem (i.e. finding a one to one mapping between G and G) which is a particular case of graph isomorphism. Many algorithms have been proposed to compute graph automorphism. Let us mention NAUTY [11], one of the most efficient in practice.

Recently, Aloul *et al.* [1] proposed an interesting technique that transform CNF formula Ψ into a graph G_Ψ where vertices are labeled with colours. Such coloured vertices are considered when searching for automorphism on the graph (i.e. vertices with different colours can not be mapped each others). The CNF formula is translated into a graph in the following way (showed in example 2):

- Each variable is represented by two vertices, one for the positive literal, the other for the negative one. A dark gray colour is associated to such vertices.
- An edge is created between two opposite literals.
- Each non binary clause is represented by a light gray colour vertex linked by edges to its associated literals.
- Each binary clause $l_1 \vee l_2$ is represented by a double edge between the vertices l_1 and l_2. This avoid additional vertex to be created.

Example 2. Let Ψ be the CNF formula given in example 1. Figure 1.a gives the associated graph $G_\Psi = (V, E)$. Where V, contains $2 \times |Var(\Psi)| = 8$ vertices (dark gray colour) associated to $Lit(\Psi)$ and 4 vertices (light gray colour) corresponding to the clauses of Ψ. The set of edges E, links literals to their opposites (e.g. $(x_1, \neg x_1)$ and clauses to its involved literals (i.e. $(c_1, \neg x_1), (c_1, \neg x_2)$ and (c_1, x_3)).
Using NAUTY [11], three automorphism that leave G_Ψ invariant are computed : $a_1 = (x_1, x_2)(x_3)(x_4)$ $[(c_1)(c_2)(c_3, c_4)]$, $a_2 = (x_1)(x_2)(x_3, x_4)[(c_1, c_2)(c_3)(c_4)]$, and $a_3 = (x_1, x_3)(x_2, x_4)[(c_1, c_3)(c_2, c_4)]$. The corresponding symmetries σ_1, σ_2 and σ_3 are obtained respectively from a_1, a_2 and a_3 by projection on literals of Ψ.

It is not difficult to extend the above transformation to QBF. Indeed, we need to consider the prefix of the QBF (see the second condition of the definition 2), because distinct literals can not be symmetric if they does not belong to the same quantifier group. To this end, we only need to introduce additional colours to make a distinction between literals vertices whose variables belong to different quantifier groups. For a QBF with k quantifier groups, we introduce k different colours. Then, QBF symmetries are detected using NAUTY on the resulting graph. In the following example, we illustrate such extended transformation from QBF to graph and the detected symmetries.



a. from CNF to Graph b. from QBF to Graph

Fig. 1. Graph reduction

Example 3. Let $\Phi = \forall x_1, x_2 \exists x_3 x_4 \Psi$ be the QBF of example 1. The figure 1.b gives the associated graph G_Φ. Here, there are 3 colours, one for the non binary clauses (light gray), one for the first universal group (white), and the last for the existential group (dark gray). This QBF Φ has two non trivial symmetry (x_1, x_2) and (x_3, x_4). The symmetry $(x_1, x_3)(x_2, x_4)$ of Ψ (see example 2) is not a symmetry of Φ, since x_1 and x_3 are not in the same quantifier group.

5 Exploiting symmetry in QBFs solvers

Symmetry breaking has been extensively investigated in the context of constraint satisfaction and satisfiability problems. The different approach proposed to break symmetries can be conveniently classified as dynamic and static schemes. Dynamic breaking, generaly search and break symmetries during search with or without using breaking predicates [3, 6]. Such approach have additional potential, in the way that it detect and exploit local symmetries (i.e. that might appears during search). Static breaking schemes refer to the techniques that detect symmetries in a preprocessing step, symmetries are generally broken by generating additional constraints, called symmetry breaking predicates(SBP) [4, 1]. Such SBP predicates eliminate from each equivalence class of symmetric models all models except one. However, in the general case, the set of symmetry predicates might be of exponential size. Recently, Aloul *et al* [1, 2] extend the approach by Crawford[4] using group theory and the concept of non redundant generator, reducing considerably the SBP size.

Contrary to the boolean case, in the context of QBFs, symmetry breaking predicates (clauses) can not conjunctively added to the QBF. More precisely, as variables are quantified, one can obtain clauses with all of its variables universally quantified, and consequently the QBF become not valid. To avoid such drawback, we propose (as shown in the Figure 2) to consider the SBP boolean formula generated according to Aloul *et al* proposed technique [1] as independent from the QBF one. An hybrid solver is then designed to deal simultaneously with the QBF and SBP formulas. At each step of QBFs backtracking-based solver (OpenQBF solver), we invoke a Satisfiability Solver (limited to Boolean constraint propagation process) on the SBP boolean formula to achieve unit propagation of the current assigned variable. If x is the current variable assigned by the QBF solver, such assignment is then propagated by the SAT solver on the SBP formula. If y is a literal deducted from the SBP formula, then two situations might be encountered and communicated to QBF Solver:

- If y is universally quantified in the QBF formula then the branch corresponding to $\neg y$ is assumed to be *t*rue and the QBF solver propagate y to confirm this assertion (i.e. if a model is

found on y then we have also a model with $\neg y$, otherwise a conflict occurs since y is universally quantified).
– If y is existentially quantified in the QBF then the branch corresponding to $\neg y$ is assumed to be $false$, in the same way, y is propagated by the QBF solver.

The Figure 2 summarises our proposed approach. The QBF solver with symmetry takes a quantified boolean formula in clausal form as input, then a preprocessing (Shatter for QBF) step on the original QBF, deliver a boolean formula made of Symmetry Breaking Predicates (SBP). An hybrid solver, combining a QBF solver with a SAT propagation solver, operates on the original QBF and the SBP formula. Grayed parts of the figure 2 correspond to our main contributions.

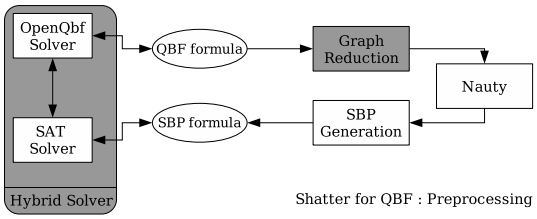


Fig. 2. QBF Solver with symmetry

6 Experiments overview of detected symmetries

We present first experiments done on a Pentium IV 3 GHz with 1GB of RAM, using a a modified version of Shatter [2] to detect symmetries on QBF formulas.

In tables 1.a and 1.b, results of symmetry detection method (as shown in section 4) on QBF03 evaluation benchmarks [8] are presented. These The QBF03 Evaluation was made of 1350 benchmarks classified according to their hardness with respect to QBF solvers presented at the competition (Easy, Medium, Hard (non solved)). At least, 729 of them are generated randomly.

Table 1.a present the number of symmetric benchmarks (the column #P gives the number of instances, #S the number of symmetric instances and % the percentage of symmetrical instances) with respect to the complete benchmarks collection. A large number of them contains symmetries (i.e. 38.5% of instances exhibit symmetries). As we can see, the easy problems are highly symmetric (80% of instances), whereas the medium category contains less symmetries (29.9%). Let us note that medium category is composed of a large fraction of random generated instances that rarely contains symmetries (see table 1.b). More interestingly, about half of the hard benchmarks representing the most difficult QBFs instances (not solved in the last competition) contains symmetries (47.9%). Breaking such symmetries might help QBF solvers to handle such hard instances. Table

Category	#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. QBF03 instances: from easy to hard

Category	#P	#S	%
Random	729	86	11.8
Non Random	621	434	69.9

b. QBF03 instances: Random vs Non Random

Table 1. Results on QBF03 Evaluation problems

2 summarise the first experimental comparison between our hybrid solver Figure2 that detect and exploit symmetries and the OpenQBF solver i.e. extended version of the Davis and Putnam procedure. For each problem instance (from the QBF'03 evaluation), a comparison is given in term of cpu times (seconds) and the number of visited nodes with and without exploiting symmetries. In addition, the number of computed symmetries and the time (seconds) needed to find them are

given. As we can see, symmetries are computed efficiently (less than one seconds). *BLOCKS3ii.4.3* planning instances contains a lot of symmetries, however such symmetries are not very relevant in the sense that they always permute variables appearing in the last quantifier group. Consequently, the branch's pruned by symmetry are located on the bottom of the search tree. Very interesting results are obtained on the two last instances (*TOILET6.1.iv.12* and *CHAIN17v18*), where symmetries are detected between variables appearing in all quantifier groups. Breaking such symmetries achieve significant reductions in the search space.

		symmetry results		OpenQBF + Symmetry		OpenQBF	
problem	Valid	Time	nb symmetries	Time	Nodes	Times	Nodes
BLOCKS3ii.4.3	No	0.02	5.7e9	78	130 521	77.6	130 521
TOILET6.1.iv.12	Yes	0.02	20 160	0.2	14	119	366 467
CHAIN17v18	Yes	0.58	3.43e11	69	65 794	227	131 105

Table 2. OpenQBF with and without symmetries

7 Conclusion and future works

Clearly, we have shown that symmetries can be naturally extended to handle Quantified Boolean Formula (QBFs). A symmetry detection algorithm is given, extending the CNF approach proposed in [4, 2] to QBFs. We have shown how such symmetries can be exploited thanks to an hybrid solver that handle both the QBF and the Symmetry Breaking Predicates formula. Preliminary experiments show the interest of our approach with respect to the discovered symmetries and the obtained gain on some class of instances. Since symmetries of the boolean formula is a superset of the QBF symmetries, an interesting path of research concerns the exploitation of such lost symmetries.

References

1. F. Aloul, A. Ramani, I. Markov, and K. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. Technical Report CSE-TR-463-02, University of Michigan, 2002.
2. F. Aloul, A. Ramani, I. Markov, and K. Sakallah. Shatter: Efficient breaking for boolean satisfiability. In *proceedings of DAC*, pages 836–839. 2003.
3. B. Benhamou and L. Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12(1):89–102, 1994.
4. J. Crawford. A theorical analysis of reasoning by symmetry in first order logic. In *Proceedings of Workshop on Tractable Reasonning, AAI92*, pages 17–22, 1992.
5. J. Crawford, M. Ginsberg, E. Luck, and A. Roy. Symmetry-breaking predicates for search problems. In *procedinngs of KR'96*, pages 148–159. 1996.
6. I. Gent and B. Smith. Symmetry breaking in constraint programming. In *Proceedings of ECAI*, pages 599–603. 2000.
7. E. Giunchiglia, M. Narizzano, and A. Tacchella. QuBE : A system for deciding Quantified Boolean Formulas Satisfiability. In *proceedings of IJCAR*. 2001.
8. D. Le Berre, L. Simon, and A. Tachella. Challenges in the QBF Arena: the SAT'03 evaluation of QBF solvers. In *proceedings of SAT*. 2003.
9. R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proceedings of Tableaux 2002*. pages 160–175. 2002.
10. S. Coste, H. Fargier, J. Lang, D. Le Berre, and P. Marquis. Function problems for quantified boolean formulas. Technical report, CRIL - University of Artois, 2003.
11. B. McKay. nauty user's guide (version 1.5). Technical Report TR-CS90 -02, Australian National University, 1990.
12. L. Zhang and S. Malik. Conflict Driven Learning in a Quantified Boolean Satisfiability Solver In *Proceedings of ICCAD*, pages 442–449. 2002.