Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers

Florian Lonsing and Armin Biere

Johannes Kepler University, Linz, Austria

Abstract. Given a quantified boolean formula (QBF) in prenex conjunctive normal form, we consider the problem of identifying variable dependencies. In related work, a formal definition of dependencies has been suggested based on quantifier prefix reordering: two variables are independent if swapping them in the prefix does not change satisfiability of the formula. Instead of the general case, we focus on the sets of depending existential variables for all universal variables which are relevant particularly for expansion-based QBF solvers. We present an approach for efficiently computing existential dependency sets by means of a directed connection relation over variables and demonstrate how this relation can be compactly represented as a tree using a union-find data structure. Experimental results show the effectiveness of our approach.

1 Introduction

The logic of quantified boolean formulae (QBF) extends propositional logic (SAT) with universal quantification, thus making it exponentially more succinct than SAT. Because of this property, QBF is a natural modelling language with a variety of applications in model checking and verification [3, 6, 11, 17]. In the domain of SAT, encouraging progress has been made during the last years in the development of efficient decision procedures based on the DPLL-framework [10]. The success is due to advanced strategies for pruning the search space such as learning, backjumping or restarts. These techniques were successfully extended to DPLL-based algorithms for QBF [9, 15, 23] but, although still being important for performance, it turned out not to guarantee similar progress as for SAT.

There is strong indication [4, 12, 13, 16] that the quantifier prefix of QBFs in prenex conjunctive normal form (PCNF) could be one reason for this phenomenon. In QBF, the presence of different types of quantifiers introduces dependencies between variables which have to be respected by QBF solvers. In many cases, dependencies resulting from linear quantifier prefixes are too pessimistic and have negative influence on solver performance. In [20], a formal definition of dependencies has been suggested and it was shown that the problem of identifying the optimal (smallest) dependency set is, as the decision problem of QBF, PSPACE-complete [21]. Because of this fact, a compromise has to be found between efficiency and optimality. Various approaches have been suggested to identify dependencies and thus overcome the drawback of linear quantifier prefixes [1, 4, 5, 7, 13, 16, 18, 20]. To our knowledge, all of these approaches are based on analyzing the syntactic structure of a QBF.

Apart from search-based QBF solvers, which suffer from dependencies in exploring irrelevant parts of the search space, handling dependencies is crucial for solvers based on variable elimination [1, 2, 5, 8, 18]. These solvers have to cope with dependency-related size increase of the formula involved with eliminations.

In this paper we address the problem of computing dependency sets of universally quantified variables for QBFs in PCNF, which is relevant for expansion-based QBF solvers [1, 2, 5, 8, 18]. Our work is based on [5, 7]. We briefly introduce universal expansion and analyze an algorithm suggested in [7] for computing dependencies of universal variables. Starting from our analysis, we develop a formal definition of dependencies in the context of expansion which is based on a syntactic connection relation of variables. We obtain a directed dependency relation by defining an equivalence relation over existential variables represented as a tree excluding transitive edges. As experimental results demonstrate, this relation allows efficient computation of dependency sets for all universal variables in a QBF. Whereas our approach can not directly be applied in search-based solvers, we see the potential of an extension to dependency sets for existential variables.

2 Preliminaries

For a set of variables V, a *literal* is either a variable $x \in V$ or its negation $\neg x$ where $v(x) = x$ and $v(\neg x) = x$ denotes the variable of a literal. A *clause* is a disjunction over literals. A propositional formula is in *conjunctive normal form* (CNF) if it consists of a conjunction over clauses.

A quantified boolean formula (QBF) $F \equiv S_1 \dots S_n\, \phi$ in *prenex conjunctive normal form* (PCNF) consists of a propositional formula ϕ over a set of variables V and a *quantifier prefix* $S_1 \dots S_n$. The quantifier prefix is a linearly ordered set of *scopes* S_i, such that $S_1 < \dots < S_n$, which forms a partition on the set of variables: $V = S_1 \cup \dots \cup S_n$ and $S_i \cap S_j = \emptyset$ for $1 \leq i, j \leq n$ and $i \neq j$. A scope S_i is *existential* if it is associated with an existential quantifier, written as $q(S_i) = \exists$ and *universal* otherwise where $q(S_i) = \forall$. $V_\exists = \bigcup S_i$ for $q(S_i) = \exists$ denotes the set of existential variables, $V_\forall = \bigcup S_i$ for $q(S_i) = \forall$ the set of universal variables. For a variable $x \in S_i$, $s(x) = S_i$ denotes the scope of x and $q(x) = q(s(x))$. For two adjacent scopes S_i and S_{i+1} where $1 \leq i < n$, $q(S_i) \neq q(S_{i+1})$. The number of scopes is the number of *quantifier alternations*.

For a scope S_i and literal l, $\delta(S_i) = i$ and $\delta(l) = \delta(s(v(l)))$ denote the *level* of S_i and of l, respectively. For scopes S_i, S_j and literals l, k, S_j is *larger* than S_i and k is larger than l if $\delta(S_i) < \delta(S_j)$ and $\delta(l) < \delta(k)$, respectively. For some variable x, $R(x) = \{y \in V \mid \delta(x) \leq \delta(y)\}$. For a QBF, $V_{\exists,i} = \{y \in V_\exists \mid \delta(y) \geq i\}$.

In the following, QBFs in PCNF are considered such that for all clauses $C = (l_1 \vee \dots \vee l_k)$, $v(l_i) \neq v(l_j)$ and $\delta(l_i) \leq \delta(l_j)$ for $1 \leq i < j \leq k$ and $q(v(l_k)) = \exists$. A clause neither contains multiple nor complementary literals of one and the same variable, all literals are sorted ascendingly according to their level and the largest literal is existential. Universal reduction [5, 8] can be applied to remove literals l_k for which $q(v(l_k)) = \forall$. Furthermore, we assume that there occurs at least one literal for each $x \in V$ in the formula.

3 Universal Expansion

Apart from solving QBF using DPLL-based algorithms where a semantic search tree is implicitly constructed [9, 10], resolution and expansion can be applied in order to successively eliminate variables at the cost of formula size [1, 2, 5, 8, 18]. In [7], cost-based expansion of universal variables was applied for preprocessing QBF, which generalizes an approach first used in Quantor [5].

Basically, expanding a universal variable $x \in V_\forall$ involves copying a subformula, assigning x and *duplicating* depending existential variables $D(x) \subseteq (R(x) \setminus V_\forall)$. Details can be found in the aforementioned publications. We focus on the computation of $D(x)$ and $|D(x)|$, respectively. Duplicating variables is necessary in order to reflect the possibility of a depending existential variable to assume different values with respect to the value of the universal variable.

Example 1. In the satisfiable formula $\forall x \exists y\ (x \vee \neg y) \wedge (\neg x \vee y)$, y depends on x: y must be assigned *true* if $x = \mathit{true}$ and *false* otherwise. Incorrectly expanding x without duplicating y yields $\exists y\ (\neg y) \wedge (y)$, which is unsatisfiable. If y is duplicated, then the resulting formula $\exists y, y'\ (\neg y) \wedge (y')$ is equisatisfiable.

A popular approach for computing set $D(x)$ is based on rules for syntactically pushing quantifiers from the prefix inside the formula, thus minimizing the subformula within the range of a quantifier: $(Qx\ \phi \otimes \psi) \equiv (Qx\ \phi) \otimes \psi$ if $x \notin V(\psi)$, $\otimes \in \{\vee, \wedge\}$ and $Q \in \{\forall, \exists\}$. This method, also called *miniscoping* [1], has been applied in various contexts [1, 4, 5, 7, 13, 18, 20]. Informally, for some QBF and variable $x \in V_\forall$, $D(x) \subseteq (R(x) \setminus V_\forall)$ is the set of variables appearing to the right of x after pushing quantifiers inside F as far as possible.

In [5] a connection relation of existential variables was defined in order to compute $D(x)$ for $x \in S_{n-1}$, which was generalized in [7] to arbitrary universal scopes: two variables v and w are *locally connected* if they occur in a common clause. The original definition [7] for computing $D(x)$ where $x \in V_\forall$ is as follows:

$$
\begin{aligned}
D^0(x) &:= \{y \in (R(x) \setminus V_\forall) \mid x \text{ is locally connected to } y\} \\
D^{k+1}(x) &:= \{z \in (R(x) \setminus V_\forall) \mid z \text{ is locally connected to } y \in D_x^k\} \\
D(x) &:= \bigcup_k D_x^k
\end{aligned}
$$

Let $X = D(x) \cup \{x\}$. Set $D(x)$ where $x \in V_\forall$ has the following properties:

1. $D(x) \subseteq (R(x) \setminus V_\forall)$
2. $\forall y \in D(x) : q(y) = \exists$ and $\delta(x) < \delta(y)$
3. $\forall y \in D(x) : x$ is connected to y via clauses containing variables in X
4. $\forall y, z \in D(x) : y$ is connected to z via clauses containing variables in X

Essentially, $D(x)$ contains existential variables which have larger levels than x only and x is connected to all variables in $D(x)$ via clauses containing variables from $D(x) \cup \{x\}$. This is also the case for all pairs of variables in $D(x)$.

In an implementation directly applying the definition, set $D(x)$ can be computed by starting at clauses C such that $x \in C$, collecting existential variables

$y \in C$ where $\delta(y) \geq \delta(x)$ and recursively inspecting clauses containing y. The connection relation is implicitly constructed. This algorithm requires $O(|F|)$ time for one $x \in V_\forall$, where $|F|$ is the length of the formula.

Example 2. For the QBF in Fig. 1, $D(1) = \{3, 4, 8, 10, 12, 13\}$, $D(2) = \{5, 9, 14\}$, $D(6) = \{8, 12, 13\}$, $D(7) = \{10\}$ and $D(11) = \{12, 13\}$

4 Defining a Directed Dependency Relation

Based on the properties of set $D(x)$, an approach for efficient computation and representation of $D(x)$ for all $x \in V_\forall$ is presented. The idea is to avoid computing a connection relation for each universal variable from scratch. Instead, such a relation is constructed once for all existential variables, which forms the basis for retrieving sets $D(x)$ and computing $|D(x)|$, respectively. For example, in expansion-based QBF solvers this information could be used in variable selection heuristics. It is shown how the connection relation can be compactly represented by defining an equivalence relation on existential variables and by excluding transitive edges. In the following, a formal definition is developed.

Definition 1. *For $x, y \in V$, y is* locally depending *on x with respect to scope S_i, written as $x \rightarrow_i y$, if, and only if $q(y) = \exists$, $\delta(y) \geq i$ and there exists a clause C such that both $x \in C$ and $y \in C$. The reflexive and transitive closure of $\rightarrow_i$ is denoted by $\rightarrow_i^*$. If $x \rightarrow_i^* y$, then y is* transitively locally depending *on x.*

The term "locally" refers to the fact that the relation is defined with respect to some scope S_i. Intuitively, if $x \rightarrow_i^* y$ for $i = \delta(x)$, then there are *connecting sets* of variables and of clauses by applying $\rightarrow_i$ transitively.

Definition 2. *For $x, y \in V_\exists$, x is* transitively locally connected *to y with respect to scope S_i, written as $x \sim_i y$, if, and only if $q(x) = q(y) = \exists$ and $x \rightarrow_i^* y$.*

Lemma 1. *For $x, y \in V_\exists, i \leq min(\delta(x), \delta(y))$: if $x \sim_i y$ then $y \sim_i x$.*

Actually, $\sim_i$ is a special case of $\rightarrow_i^*$ by restricting the set of variables to $V_\exists$.

Example 3. For the QBF in Fig. 1, $3 \sim_2 10$ since $q(3) = q(10) = \exists$ and $3 \rightarrow_2^* 10$ via variables $12, 4$ and 10.

Definition 3. *For $x, y \in V$, x is* globally connected *to y, written as $x \approx y$, if, and only if either $x = y$ and $q(x) = \forall$ or $q(x) = q(y) = \exists$, $\delta(x) = \delta(y) = i$ and $x \sim_i y$.*

Relation $\approx$ is "global" because the definition is independent from a particular scope. It follows from Def. 3 that $\forall x, y \in V : x \approx y \Rightarrow s(x) = s(y) \Rightarrow \delta(x) = \delta(y)$.

Theorem 1. *$\approx$ is an equivalence relation. For $x \in V$, $[x]$ is the class of x.*

Example 4. For the QBF in Fig. 1, $3 \approx 4$ because $q(3) = q(4) = \exists$ and $\delta(3) = \delta(4) = 2$ and $3 \sim_2 4$ since $3 \rightarrow_2^* 4$ via variable 12. Furthermore, $12 \approx 13$.

Theorem 2. $\forall x, y \in V$*:* $x \rightarrow_i^* y$ *if, and only if* $\forall x' \in [x], y' \in [y] : x' \rightarrow_i^* y'$.

Thm. 2 states compatibility of $\rightarrow_i^*$ with $\approx$: if two variables are connected (dependent) then so are all members of their respective classes and vice versa.

Definition 4. $\rightsquigarrow^*$ *denotes the* global directed dependency relation. *For* $x, y \in V$, $[x] \rightsquigarrow^* [y]$ *if, and only if,* $\delta(x) \leq \delta(y)$ *and* $x \rightarrow_i^* y$ *for* $i = \delta(x)$. *The transitive reduction of* $\rightsquigarrow^*$ *is denoted by* $\rightsquigarrow$.

Corollary 1. $\forall x, y \in V$: *if* $[x] \rightsquigarrow^* [y]$ *and* $[x] \neq [y]$ *then* $\delta(x) < \delta(y)$.

Corollary 2. $\forall x, y \in V$: *if* $[x] \rightsquigarrow^* [y]$ *then either* $[x] = [y]$ *or* $\delta(x) < \delta(y)$.

By Cor. 1 and Cor. 2, if $[x] \rightsquigarrow^* [y]$ then either x and y are in the same class or in different classes but from different scopes.

Theorem 3. *For* $x \in V_\forall, i = \delta(x)$:
$D(x) = \{y \in V_\exists \mid x \rightarrow_i^* y\} = \{y \in V_\exists \mid [x] \rightarrow_i^* [y]\} = \{y \in V_\exists \mid [x] \rightsquigarrow^* [y]\}$.

By Thm. 3, relations $\rightarrow_i^*$, $\rightarrow_i^*$ combined with $\approx$ and $\rightsquigarrow^*$ are equivalent in theory for computing $D(x)$. Note that computing $D(x)$ by $\rightarrow_i^*$ corresponds to applying the original definition (see also Sec. 3) introduced in [7]. From a practical point of view, $\rightarrow_i^*$ is restricted to classes by $\approx$ which again can be improved with a compact representation of $\rightsquigarrow$.

4.1 Efficiently Representing Directed Dependency Relations

Lemma 2. *For* $\rightsquigarrow^*$ *on* $V_\exists$, $\rightsquigarrow$ *can be represented as a forest.*

By Lem. 2 and due to the properties of $\rightsquigarrow^*$, in $\rightsquigarrow$ at most one class is related to another, that is, situations like in directed acyclic graphs can not occur. This allows a representation as a forest.

The *connection forest* (c-forest) for a QBF with m existential scopes is a collection of trees over $V_\exists$ with respect to $\approx$ with the following properties:

1. $\forall x, y \in V_\exists$: there is an edge $([x], [y])$ if, and only if $[x] \rightsquigarrow [y]$.
2. $\forall x, y \in V_\exists$: there is a path from $[x]$ to $[y]$ if, and only if $[x] \rightsquigarrow^* [y]$.
3. the maximum length (number of edges) of a path is $m - 1$

All paths consist of classes only, the levels of which are strictly increasing by Cor. 1. Hence the maximum path length is $m - 1$.

4.2 Computing Dependencies by Directed Dependency Relations

Given a QBF F in PCNF, the corresponding c-forest for $V_\exists$ is the basis for computing $D(x)$ for all $x \in V_\forall$. For $x \in V_\forall, y \in V_\exists$ and the c-forest, let $h(x, [y]) = [y']$ such that $y' \in V_\exists, [y'] \rightsquigarrow^* [y]$, $\delta(x) < \delta(y')$ and there is no $y'' \in V_\exists$ with $\delta(x) < \delta(y'') < \delta(y')$ and $[y''] \rightsquigarrow^* [y']$. That is, in the c-forest $h(x, [y])$ denotes the smallest *ancestor* of $[y]$ which is larger than x. Computing set $D(x)$ for some $x \in V_\forall$ in a QBF F involves the following steps:

1. let $C(x) = \{C \in F \mid x \in C\}$
2. let $X(x) = \{[y] \mid y \in V_{\exists,i}, i = \delta(x) \text{ and } y \in C \text{ for } C \in C(x)\}$
3. let $H(x) = \{[z] \mid [z] = h(x, [y]) \text{ for } [y] \in X(x)\}$
4. let $H^*(x) = \{[y] \mid [z] \rightsquigarrow^* [y] \text{ for } [z] \in H(x)\}$
5. $D(x) = \{z \mid z \in [y] \text{ for } [y] \in H^*(x)\}$

Starting from clauses containing a literal of x, classes of existential variables larger than x are collected (steps 1 and 2). For all collected classes, the set of ancestors $H(x)$ is determined (step 3). Next, descendants of classes in $H(x)$ are collected in $H^*(x)$ (step 4). Finally, the members of classes in $H^*(x)$ exactly correspond to $D(x)$ (step 5). Computing $D(x)$ from a c-forest does not require searching since subtrees rooted at variables in $H(x)$ denote subsets of $D(x)$. Thus the c-forest, which is computed once and then shared between all $x \in V_\forall$, combined with sets $H(x)$ is sufficient to identify and compactly represent $D(x)$.

Example 5. Fig. 1 shows a c-forest (dotted edges) and sets $H(x)$ (solid edges). Variables 3,4 and 12,13 are in one class, respectively (horizontal edges). According to the steps in Sec. 4.2, $C(1) = C(6) = \{(1, 6, 8, 13)\}$, $X(1) = X(6) = \{[8], [13]\}$, $H(1) = \{[3]\}$, $H^*(1) = \{[3], [8], [10], [13]\}$, $H(6) = \{[8]\}$, $H^*(6) = \{[8], [13]\}$, $D(1) = \{3, 4, 8, 10, 12, 13\}$ and $D(6) = \{8, 12, 13\}$. Note that path $[8], [13]$ is shared between variables 1 and 6.

$\delta(S)$	$q(S)$	S	(2, 5, 9)
1	$\forall$	1, 2	(5, 9, 14)
2	$\exists$	3, 4, 5	(3, 8, 12)
3	$\forall$	6, 7	(4, 7, 10)
4	$\exists$	8, 9, 10	(4, 12, 13)
5	$\forall$	11	(1, 6, 8, 13)
6	$\exists$	12, 13, 14	(11, 12)

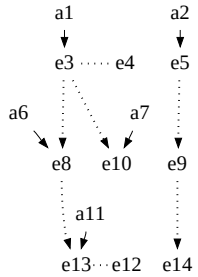

Fig. 1. QBF example. The table on the left shows the levels, quantifiers and variables for each scope in the first three columns and clauses as lists of literals in the last column. Variables and literals are uniquely identified by integers as in QDIMACS format [19]. The corresponding c-forest including sets $H(x)$ (see Sec. 4.1 and 4.2) is depicted on the right, where variable types are denoted by identifier prefixes "a" ($\forall$) or "e" ($\exists$).

5 Experimental Results

We have implemented a tool which, given a QBF in PCNF, builds the c-forest and determines sets $H(x)$ for all $x \in V_\forall$ incrementally. Clauses are inspected exactly once one after another: a pair of variables $x, y \in C$ for some clause C

where $x \in V_\forall$, $y \in V_\exists$ and $\delta(x) < \delta(y)$ results in an update of $H(x)$ by adding $h(x, [y])$. If $x, y \in V_\exists$ and $\delta(x) \leq \delta(y)$ then the c-forest is updated by inserting an edge for $[x] \rightsquigarrow^* [y]$. Relation $\approx$ is computed using an efficient union-find algorithm [22]. Table 1 shows experimental results. [1]

	QBFEVAL'05	*QBFEVAL'06*	*QBFEVAL'07*	*QBFEVAL'08*
size	211	216	1136	3328
total time	7.46	1.29	195.75	267.49
avg. time	0.04	0.01	0.17	0.08
max. $\lvert H^*(x)\rvert$	797	5	797	1872
avg. $\lvert H^*(x)\rvert$	19.51	1.21	39.07	8.24
max. $\lvert D(x)\rvert$	256535	9993	2177280	2177280
avg. $\lvert D(x)\rvert$	82055.87	4794.60	33447.6	19807
avg. $\frac{\lvert H^*(x)\rvert}{\lvert D(x)\rvert}$	3.44 %	0.04 %	6.42 %	1.21 %
$\approx_\exists$	3.08 %	3.95 %	2.20 %	7.37 %

Table 1. Experimental results on structured instances from QBF competitions 2005 to 2008 [14]. The first line shows the number of formulae per set. Run times are in seconds for the whole set and on average per formula. Maximum and average sizes of sets $H^*(x)$ and $D(x)$ for $x \in V_\forall$ are reported (see also Sec. 4.2). $\frac{|H^*(x)|}{|D(x)|}$ for $x \in V_\forall$ relates the sizes of the two representations of $D(x)$: the c-forest is compared to the directly computed set by $\rightarrow_i^*$. The worst-case value is 100%, which means no improvement can be achieved by the c-forest. On the contrary, the average of $\frac{|H^*(x)|}{|D(x)|}$ over all $x \in V_\forall$ indicates that the c-forest representing $\rightsquigarrow$ allows to represent $D(x)$ more compactly than $\rightarrow_i^*$. This observation is supported by the maximum and average values of $|H^*(x)|$ and $|D(x)|$. The last line reports the ratio between the total number of equivalence classes and the total number of *existential* variables per formula set, where the worst-case value is 100%: one class per variable. The values indicate that there are few, yet large classes which demonstrates the compacting effect of relation $\approx$.

6 Conclusion

We have presented an efficient way to compute dependency sets for all universal variables in QBFs, which is relevant for expansion-based QBF solvers. As previous work, our approach relies on a syntactic connection relation of variables. By defining an equivalence relation on existential variables and excluding transitive edges, we obtain a directed connection relation which can be implemented using a tree and a union-find data structure and which can be shared between all universal variables. Experiments show that dependencies can be compactly represented with our approach. We are planning to extend this method to dependency sets for existential variables for use in search-based QBF solvers. Furthermore, our

[1] Setup: 64-bit Ubuntu Linux 8.04, Intel Q6700 at 2.66 GHz, 8 GB of memory.

representation can be regarded as *static*, that is, the effect of removing clauses from the formula has not yet been taken into consideration. Certainly, a *dynamic* version will be more flexible when used in combination with variable expansion.

References

1. A. Ayari and D. A. Basin. QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In *Proc. FMCAD'02*.
2. M. Benedetti. sKizzo: a Suite to Evaluate and Certify QBFs. In *Proc. CADE'05*.
3. M. Benedetti and H. Mangassarian. QBF-Based Formal Verification: Experience and Perspectives. *JSAT*, 2008.
4. Marco Benedetti. Quantifier Trees for QBFs. In *Proc. SAT'05*.
5. A. Biere. Resolve and Expand. In *Proc. SAT'04*.
6. A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proc. TACAS'99*.
7. U. Bubeck and H. Kleine Büning. Bounded Universal Expansion for Preprocessing QBF. In *Proc. SAT'07*.
8. H. Kleine Büning, M. Karpinski, and A. Flögel. Resolution for Quantified Boolean Formulas. *Inf. Comput.*, 117(1), 1995.
9. M. Cadoli, A. Giovanardi, and M. Schaerf. An Algorithm to Evaluate Quantified Boolean Formulae. In *JAR*, 1998.
10. M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
11. N. Dershowitz, Z. Hanna, and J. Katz. Bounded Model Checking with QBF. In *Proc. SAT'05*.
12. U. Egly, M. Seidl, H. Tompits, S. Woltran, and M. Zolda. Comparing Different Prenexing Strategies for Quantified Boolean Formulas. In *Proc. SAT'03*.
13. U. Egly, H. Tompits, and S. Woltran. On Quantifier Shifting for Quantified Boolean Formulas. In *Proc. SAT'02*.
14. E. Giunchiglia, M. Narizzano, and A. Tacchella. QBF Solver Evaluation Portal, 2001-2008. `www.qbflib.org/index_eval.php`.
15. E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for Quantified Boolean Logic Satisfiability. *Artif. Intell.*, 145(1-2):99–120, 2003.
16. E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantifier Structure in Search-Based Procedures for QBFs. *TCAD*, 26(3):497–507, 2007.
17. T. Jussila and A. Biere. Compressing BMC Encodings with QBF. *ENTCS*, 174(3):45–56, 2007.
18. F. Lonsing and A. Biere. Nenofex: Expanding NNF for QBF Solving. In *Proc. SAT'08*.
19. QBFLIB. QDIMACS Standard v1.1. `http://www.qbflib.org/qdimacs.html`.
20. M. Samer and S. Szeider. Backdoor Sets of Quantified Boolean Formulas. In *Proc. SAT'07*.
21. L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time: Preliminary Report. In *STOC*, pages 1–9, 1973.
22. Robert Endre Tarjan. Efficiency of a Good But Not Linear Set Union Algorithm. *J. ACM*, 22(2):215–225, 1975.
23. L. Zhang and S. Malik. Conflict Driven Learning in a Quantified Boolean Satisfiability Solver. In *Proc. ICCAD'02*.