Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 251 (2009) 83–95

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers

Florian Lonsing and Armin Biere[1]

*Institute for Formal Models and Verification*
*Johannes Kepler University*
*Linz, Austria*

**Abstract**

Given a quantified boolean formula (QBF) in prenex conjunctive normal form (PCNF), 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. This is 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.

*Keywords:* Quantified Boolean Formulae, QBF, expansion, dependencies.

## 1 Introduction

The logic of quantified boolean formulae (QBF) extends propositional logic (SAT) with universal quantification. Whereas QBF is not more expressive than SAT, relevant problems from model checking and verification [6,8,13,19] often can be encoded more compactly in QBF than in SAT. 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 [12]. 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 [11,17,26] but, although still being important for performance, it turned out not to guarantee similar progress as for SAT.

[1] http://fmv.jku.at/

doi:10.1016/j.entcs.2009.08.029

There is strong indication [4,14,15,18] 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 [23] 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 [24]. 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 [2,4,7,9,15,18,20,23]. 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 [2,5,7,10,20]. 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 [2,5,7,10,20]. Our work is based on [7,9]. We briefly introduce universal expansion and analyze an algorithm suggested in [9] for computing dependencies of universal variables. Starting from our analysis we develop a formal definition of dependencies in the context of universal expansion. The definition is based on a syntactic connection relation of variables. We obtain a directed dependency relation by defining an equivalence relation over existential variables. This relation can be 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. As we do not consider dependencies of existential variables, our approach can not directly be applied in search-based solvers, yet we see the potential of extending our work to dependency sets for existential variables. [2]

## 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 $\phi$ in CNF 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 $i \neq j$.

[2] In [21] we have in fact extended our approach. We obtained a compact graph representation for dependencies of both existential and universal variables. This representation is also based on syntactic variable connections [23].

A scope $S_i$ is *existential* if it is associated with an existential quantifier, written as $q(S_i) = \exists$ and *universal* otherwise where $q(S_i) = \forall$. The set of existential and universal variables is denoted by $V_\exists = \bigcup S_i$ for $q(S_i) = \exists$ and $V_\forall = \bigcup S_i$ for $q(S_i) = \forall$, respectively. For a variable $x \in S_i$, $s(x) = S_i$ is the scope of $x$ and $q(x) = q(s(x))$ the *type* of $x$. For two adjacent scopes $S_i$ and $S_{i+1}$ where $1 \leq i < n$, $q(S_i) \neq q(S_{i+1})$. Given a QBF with $n$ scopes, there are $n-1$ *quantifier alternations*.

For some variable $x$, $R(x) = \{y \in V \mid \delta(x) \leq \delta(y)\}$. That is, $R(x)$ is the set of all variables larger than $x$.

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.

Let $R \subseteq V \times V$ be a binary relation on the set of variables $V$. The *reflexive and transitive closure* of $R$ is the smallest reflexive and transitive $R' \subseteq V \times V$ such that $R \subseteq R'$. The *reflexive and transitive reduction* of $R$ is the smallest $R' \subseteq V \times V$ such that $R$ and $R'$ have the same reflexive and transitive closure.

In the following, QBFs in PCNF are considered such that for all clauses $C = (l_1 \vee \ldots \vee l_k)$, $v(l_i) \neq v(l_j)$ 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 [7,10] 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 [11,12], resolution and expansion can be applied in order to successively eliminate variables at the cost of formula size [2,5,7,10,20]. In [9], cost-based expansion of universal variables was applied for preprocessing QBF, which generalizes an approach first used in Quantor [7].

Basically, expanding a universal variable $x \in V_\forall$ involves copying a subformula, assigning $x$ and *duplicating* depending existential variables $D(x) \subseteq (R(x) \setminus V_\forall)$. For detailed information about expansion we refer to the aforementioned publications. In this work we focus on efficient computation and representation 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 3.1** 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 $x =$ *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 sub-

| $i$ | $q(S_i)$ | $S_i$ | (a2, e5, e9) |
|---|---|---|---|
| 1 | $\forall$ | a1, a2 | (e5, e9, e14) |
| 2 | $\exists$ | e3, e4, e5 | (e3, e8, e12) |
| 3 | $\forall$ | a6, a7 | (e4, a7, e10) |
| 4 | $\exists$ | e8, e9, e10 | (e4, e12, e13) |
| 5 | $\forall$ | a11 | (a1, a6, e8, e13) |
| 6 | $\exists$ | e12, e13, e14 | (a11, e12) |

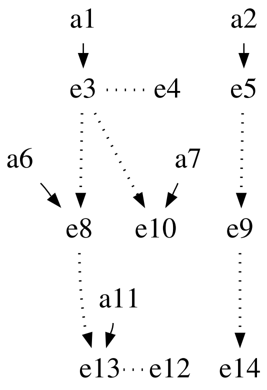

Fig. 1. QBF example. The table on the left shows the levels, quantifiers and variables for each scope in the first three columns and clauses as lists of literals in the last column. Variables and literals are uniquely identified by integers as in QDIMACS format [22]. The corresponding c-forest including sets $H(x)$ (see Sec. 4.1 and 4.2) is depicted on the right. Identifier prefixes "a" and "e" indicate variable types $\forall$ and $\exists$, respectively.

formula 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* [2], has been applied in various contexts [2,4,7,9,15,20,23]. 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 [7] a connection relation of existential variables was defined in order to compute $D(x)$ for $x \in S_{n-1}$, which was generalized in [9] to arbitrary universal scopes: two variables $v$ and $w$ are *locally connected* if they occur in a common clause. The original definition [9] for computing $D(x)$ where $x \in V_\forall$ is as follows:

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

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

(i) $D(x) \subseteq (R(x) \setminus V_\forall)$

(ii) For $y \in D(x) : q(y) = \exists$ and $\delta(x) < \delta(y)$

(iii) For $y \in D(x)$ : $x$ is connected to $y$ via clauses containing variables in $X$

(iv) For $y, z \in D(x)$ : $y$ 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 $\delta(x) \leq \delta(y)$ and recursively inspecting clauses containing $y$. During this process, 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 3.2** For the QBF in Fig. 1, $D(\mathsf{a1}) = \{\mathsf{e3}, \mathsf{e4}, \mathsf{e8}, \mathsf{e10}, \mathsf{e12}, \mathsf{e13}\}$, $D(\mathsf{a2}) = \{\mathsf{e5}, \mathsf{e9}, \mathsf{e14}\}$, $D(\mathsf{a6}) = \{\mathsf{e8}, \mathsf{e12}, \mathsf{e13}\}$, $D(\mathsf{a7}) = \{\mathsf{e10}\}$ and $D(\mathsf{a11}) = \{\mathsf{e12}, \mathsf{e13}\}$.

## 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 4.1** For a QBF, $V_{\exists,i} = \{y \in V_\exists \mid i \leq \delta(y)\}$.

In Def. 4.1 $V_{\exists,i}$ denotes the set of existential variables which are larger than or equal to scope $S_i$. We first introduce variable connections by clauses containing common variables (definition adapted from [23]). This will be needed for proving some of our results.

**Definition 4.2** For $x, y \in V$ and $X \subseteq V$, an $X$*-path* between $x$ and $y$ is a sequence $C_1, \ldots, C_k$ of clauses such that $x \in C_1$, $y \in C_k$ and $C_i \cap C_{i+1} \cap X \neq \emptyset$ for $1 \leq i < k$.

**Example 4.3** For the QBF in Fig. 1, there is an $X$-path between $\mathsf{a1}$ and $\mathsf{e3}$ for $X = \{\mathsf{e12}, \mathsf{e13}\}$ and clauses $(\mathsf{a1}, \mathsf{a6}, \mathsf{e8}, \mathsf{e13})$, $(\mathsf{e4}, \mathsf{e12}, \mathsf{e13})$ and $(\mathsf{e3}, \mathsf{e8}, \mathsf{e12})$.

**Definition 4.4** 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 $q(y) = \exists$, $i \leq \delta(y)$ 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 $S_i$. There is a correspondence between $\rightarrow_i^*$ and $X$-paths, which follows from Def. 4.4 and Def. 4.2.

**Corollary 4.5** *For $x, y \in V$, if $x \rightarrow_i^* y$, then there is an $X$-path between $x$ and $y$ for $X = V_{\exists,i}$.*

Due to Def. 4.4 the converse of Cor. 4.5 does not hold in general. For example, if there is an $X$-path between $x \in V_\exists$ and $y \in V_\forall$ then $x \not\rightarrow_i^* y$ for all $i$. A weaker variant can be stated as follows.

**Corollary 4.6** *For $x \in V, y \in V_\exists$, if there is an $X$-path between $x$ and $y$ for $X = V_{\exists,i}$ and $i \leq min(\delta(x), \delta(y))$, then $x \rightarrow_i^* y$.*

By Def. 4.4 connections with respect to a scope $S_j$ are preserved for any smaller scope $S_i$ as stated in the following corollary.

**Corollary 4.7** *For $x, y \in V$, $i \leq j$: if $x \rightarrow_j^* y$, then also $x \rightarrow_i^* y$.*

**Example 4.8** For the QBF in Fig. 1, $\mathsf{e3} \rightarrow_4 \mathsf{e8}$ but $\mathsf{e3} \not\rightarrow_5 \mathsf{e8}$, $\mathsf{e8} \rightarrow_6 \mathsf{e12}$ and by Cor. 4.7 also $\mathsf{e8} \rightarrow_1 \mathsf{e12}$, further $\mathsf{e3} \rightarrow_2^* \mathsf{e13}$.

**Definition 4.9** 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 $x \rightarrow_i^* y$.

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

**Lemma 4.10** *For $x, y \in V_\exists, i \leq min(\delta(x), \delta(y))$ : if $x \sim_i y$ then $y \sim_i x$.*

**Proof.** If $x \sim_i y$ for $x, y \in V_\exists$ and $i \leq min(\delta(x), \delta(y))$, then by Def. 4.9 also $x \rightarrow_i^* y$. By Cor. 4.5, there is an $X$-path between $x$ and $y$ for $X = V_{\exists,i}$ and a sequence of clauses $C_1, \ldots, C_k$. Then the reversed sequence of clauses $C_k, \ldots, C_1$ is an $X$-path between $y$ and $x$. Since $x, y \in V_\exists$ and $i \leq min(\delta(x), \delta(y))$, by Cor. 4.6 also $y \rightarrow_i^* x$ and further $y \sim_i x$ by Def. 4.9. □

**Example 4.11** For the QBF in Fig. 1, $\mathsf{e3} \sim_2 \mathsf{e10}$ since $q(\mathsf{e3}) = q(\mathsf{e10}) = \exists$ and $\mathsf{e3} \rightarrow_2^* \mathsf{e10}$ via variables $\mathsf{e12}, \mathsf{e4}$ and $\mathsf{e10}$.

**Definition 4.12** For $x, y \in V$, $x$ is *globally connected* to $y$, written as $x \approx y$, if, and only if either (1) $x = y$ or (2) $q(x) = q(y) = \exists$, $\delta(x) = \delta(y) = i$ and $x \sim_i y$.

Relation $\approx$ is "global" because the definition is independent from a particular scope. It follows from Def. 4.12 that for $x, y \in V$ if $x \approx y$ then also $s(x) = s(y)$ and $\delta(x) = \delta(y)$. We now prove that $\approx$ allows to merge existential variables from the same scope into equivalence classes. This is an important step towards a compact representation of dependencies.

**Theorem 4.13** *$\approx$ is an equivalence relation. For $x \in V$, $[x]$ is the class of $x$.*

**Proof.** Reflexivity is trivial since $x \approx x$ for $x \in V$ by Def. 4.12. If *not* $q(x) = q(y) = \exists$ then by Def. 4.12 $x \approx y$ if, and only if $x = y$. Since $=$ is an equivalence relation, symmetry and transitivity of $\approx$ follow immediately. Otherwise, assume $q(x) = q(y) = \exists$. If $x \approx y$ and $x = y$, then also $y \approx x$ by Def. 4.12. If $x \approx y$ and $x \neq y$ then by Def. 4.12 $x \sim_i y$ for $i = \delta(x) = \delta(y)$. Then by Lem. 4.10 and since $i = \delta(x) = \delta(y)$ also $y \sim_i x$ and hence $y \approx x$. Therefore $\approx$ is symmetric. To show transitivity, assume $x \approx y'$ and $y' \approx y$ for $y' \in V$. If $y' \in V_\forall$ then trivially $x = y' = y$ by Def. 4.12 and hence also $x \approx y$. Otherwise $y' \in V_\exists$ and by Def. 4.12 and Def. 4.9 also $x \rightarrow_i^* y'$, $y' \rightarrow_i^* y$ for $i = \delta(x) = \delta(y') = \delta(y)$ and $q(x) = q(y') = q(y) = \exists$. By $x \rightarrow_i^* y'$, $y' \rightarrow_i^* y$ and transitivity of $\rightarrow_i^*$, also $x \rightarrow_i^* y$, and further $x \approx y$ since $x \sim_i y$. □

**Example 4.14** For the QBF in Fig. 1, $\mathsf{e3} \approx \mathsf{e4}$ because $q(\mathsf{e3}) = q(\mathsf{e4}) = \exists$ and $\delta(\mathsf{e3}) = \delta(\mathsf{e4}) = 2$ and $\mathsf{e3} \sim_2 \mathsf{e4}$ since $\mathsf{e3} \rightarrow_2^* \mathsf{e4}$ via variable $\mathsf{e12}$. Furthermore, $\mathsf{e12} \approx \mathsf{e13}$.

**Lemma 4.15** *Let $x \in V, i = \delta(x)$ and $y \in [x]$. Then $x \rightarrow_i^* y$.*

**Proof.** If $x \in V_\forall$ and $y \in [x]$ then trivially $x = y$ because $x \approx y$ by Def. 4.12 and also $x \rightarrow^*_i x$ for $i = \delta(x)$ by reflexive closure of $\rightarrow_i$ as in Def. 4.4. If $x \in V_\exists$ then $x \approx y$ since $y \in [x]$ and $x \sim_i y$ for $i = \delta(x) = \delta(y)$ by Def. 4.12 and further $x \rightarrow^*_i y$ by Def. 4.9. □

**Theorem 4.16** *Let $x, y \in V, i \leq min(\delta(x), \delta(y))$. Then $x \rightarrow^*_i y$ if, and only if $x' \rightarrow^*_i y'$ for all $x' \in [x], y' \in [y]$.*

**Proof.** The proof works regardless of the types of $x$ and $y$ by Def. 4.4 (reflexivity of $\rightarrow^*_i$), Cor. 4.7 and Def. 4.12. Trivial cases arise for $V_\forall$. Assume $x \rightarrow^*_i y$ for $x, y \in V$ and $i \leq min(\delta(x), \delta(y))$. Then for $x' \in [x], y' \in [y]$, $x' \rightarrow^*_i x$ and $y \rightarrow^*_i y'$ by Cor. 4.7 and Def. 4.12. Since $x' \rightarrow^*_i x$, $x \rightarrow^*_i y$ (by assumption), $y \rightarrow^*_i y'$, also $x' \rightarrow^*_i y'$ by transitivity of $\rightarrow^*_i$. The other direction can be shown analogously. □

Lem. 4.15 and Thm. 4.16 state compatibility of $\rightarrow^*_i$ with $\approx$: if two variables are connected (dependent) then so are all members of their respective classes and vice versa. When regarding $[x]$ as an arbitrary class member, we may write, for example, $[x] \rightarrow^*_i [y]$ by Thm. 4.16. This notation denotes connections between classes. Note that Thm. 4.16 would not hold for arbitrary values of $i$. For example, if $\delta(x) < i$ then $x \not\rightarrow^*_i x'$ for $x' \in [x]$, which contradicts Def. 4.12.

**Definition 4.17** $\rightsquigarrow^*$ 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 $\rightsquigarrow$.

**Lemma 4.18** *For $x, y \in V$ : if $[x] \rightsquigarrow^* [y]$ and $[x] \neq [y]$ then $\delta(x) < \delta(y)$.*

**Proof.** Assume $[x] \rightsquigarrow^* [y]$, $[x] \neq [y]$ but $\delta(x) \geq \delta(y)$. Then by Def. 4.17, $\delta(x) = \delta(y)$ and hence also $q(x) = q(y)$. Since $[x] \rightsquigarrow^* [y]$, also $x \rightarrow^*_i y$ for $i = \delta(x) = \delta(y)$ and $[x] \rightarrow^*_i [y]$ by Thm. 4.16 and hence $[x] \approx [y]$ by Def. 4.9 and Def. 4.12. Then $[x] = [y]$ which contradicts the assumption. □

The proof of the following lemma works analogously to the one of Lem. 4.18.

**Lemma 4.19** *For $x, y \in V$ : if $[x] \rightsquigarrow^* [y]$ then either $[x] = [y]$ or $\delta(x) < \delta(y)$.*

By Lem. 4.18 and Lem. 4.19, if $[x] \rightsquigarrow^* [y]$ then either $x$ and $y$ are in the same class or in different classes but from different scopes.

**Theorem 4.20** *For $x \in V_\forall, i = \delta(x)$ :*
*$D(x) = \{y \in V_\exists \mid x \rightarrow^*_i y\} = \{y \in V_\exists \mid [x] \rightarrow^*_i [y]\} = \{y \in V_\exists \mid [x] \rightsquigarrow^* [y]\}$.*

**Proof.** Assume $x \in V_\forall$ and $i = \delta(x)$.

$D(x) = \{y \in V_\exists \mid x \rightarrow^*_i y\}$ holds since $D(x)$ as defined in Sec. 3 corresponds to the transitive closure of $\rightarrow_i$ in Def. 4.4.

To show $\{y \in V_\exists \mid x \rightarrow^*_i y\} \subseteq \{y \in V_\exists \mid [x] \rightarrow^*_i [y]\}$ first assume $x \rightarrow^*_i y, y \in V_\exists$. By Def. 4.12, $x \approx x$ and $x = [x]$, hence $[x] \rightarrow^*_i y$ and $i \leq \delta(y)$ by Def. 4.4. In fact, $i < \delta(y)$ because $i \neq \delta(y)$ since $q(x) \neq q(y)$. For $j = \delta(y) = \delta([y])$, $y \rightarrow^*_j [y]$ because $y \approx [y]$. By Cor. 4.7 and $y \rightarrow^*_j [y]$, also $y \rightarrow^*_i [y]$ since $i < j$. By transitivity of $\rightarrow^*_i$

as in Def. 4.4, $[x] \rightarrow_i^* y$ and $y \rightarrow_i^* [y]$, also $[x] \rightarrow_i^* [y]$. The other direction follows right from Thm. 4.16, hence $\{y \in V_\exists \mid x \rightarrow_i^* y\} = \{y \in V_\exists \mid [x] \rightarrow_i^* [y]\}$.

To show $\{y \in V_\exists \mid [x] \rightarrow_i^* [y]\} \subseteq \{y \in V_\exists \mid [x] \rightsquigarrow^* [y]\}$ first assume $[x] \rightarrow_i^* [y], y \in V_\exists$. Then $i < \delta([y])$ since $i \leq \delta([y])$ by Def. 4.4 and $i \neq \delta([y])$ because $q(x) \neq q(y)$. By Def. 4.12, $\delta(y) = \delta([y])$ and hence $\delta(x) < \delta(y)$ where $i = \delta(x)$. If $[x] \rightarrow_i^* [y]$ for $i = \delta(x)$ then also $x \rightarrow_i^* y$ and hence $[x] \rightsquigarrow^* [y]$ by Def. 4.17 and $\delta(x) < \delta(y)$. The other direction follows from Def. 4.17, $q(x) \neq q(y)$, $\delta(x) < \delta(y)$, $x \rightarrow_i^* y$ for $i = \delta(x)$ and Thm. 4.16. Hence $\{y \in V_\exists \mid [x] \rightarrow_i^* [y]\} = \{y \in V_\exists \mid [x] \rightsquigarrow^* [y]\}$. □

By Thm. 4.20, 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 [9]. 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*

Applying relation $\rightsquigarrow^*$ for dependency computation is most interesting for practical application. Since $\rightsquigarrow^*$ is directed, it restricts the set of classes to be considered when connections of a variable are determined. This contributes to compactness in addition to equivalence classes induced by $\approx$. In this section we first examine properties of $\rightsquigarrow^*$ over existential variables which allow to efficiently represent its reflexive and transitive reduction $\rightsquigarrow$ as a tree. This tree can be shared between all variables and is the basis for a graph data-structure representing dependencies of universal variables.

The following lemma states a property of $\rightsquigarrow^*$ which accounts for the tree structure of $\rightsquigarrow$.

**Lemma 4.21** *Let $x, y, z \in V_\exists$ where $\delta(x) \leq \delta(y)$. If $[x] \rightsquigarrow^* [z]$ and $[y] \rightsquigarrow^* [z]$ then $[x] \rightsquigarrow^* [y]$.*

**Proof.** Assume $[x] \rightsquigarrow^* [z]$ and $[y] \rightsquigarrow^* [z]$ for $x, y, z \in V_\exists$ where $\delta(x) \leq \delta(y)$. Then by Def. 4.17, $x \rightarrow_i^* z$ for $i = \delta(x)$ and $y \rightarrow_j^* z$ for $j = \delta(y)$ and $\delta(x) \leq \delta(y) \leq \delta(z)$. By Cor. 4.7 also $y \rightarrow_i^* z$ and by Lem. 4.10 $z \rightarrow_i^* y$. By Def. 4.4, $x \rightarrow_i^* z$ and $z \rightarrow_i^* y$, also $x \rightarrow_i^* y$ and $[x] \rightsquigarrow^* [y]$. □

**Corollary 4.22** *For $\rightsquigarrow^*$ on $V_\exists$, $\rightsquigarrow$ can be represented as a forest.*

By Lem. 4.21, whenever $[x] \rightsquigarrow^* [z]$ and $[y] \rightsquigarrow^* [z]$ for $\delta(x) \leq \delta(y)$ then $[x] \rightsquigarrow^* [y]$. That is, there is always a transitive edge of the form $[x] \rightsquigarrow^* [z]$ by $[x] \rightsquigarrow^* [y]$ and $[y] \rightsquigarrow^* [z]$. The transitive reduction $\rightsquigarrow$ of $\rightsquigarrow^*$ removes $[x] \rightsquigarrow^* [z]$ such that at most one class is related to another one. Hence by Cor. 4.22 and due to the properties of $\rightsquigarrow^*$, in $\rightsquigarrow$ situations like in directed acyclic graphs can not occur. This allows $\rightsquigarrow$ on existential variables to be represented as a forest. Note that since $\rightsquigarrow^*$ is directed by Def. 4.17 and hence also antisymmetric and acyclic, its transitive reduction $\rightsquigarrow$ is unique [1].

**Definition 4.23** 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:

(i) For $x, y \in V_\exists$ : there is an edge $([x],[y])$ if, and only if $[x] \rightsquigarrow [y]$.

(ii) For $x, y \in V_\exists$ : there is a path from $[x]$ to $[y]$ if, and only if $[x] \rightsquigarrow^* [y]$.

(iii) the maximum length (number of edges) of a path is $m - 1$.

All paths in the c-forest consist of classes only, the levels of which are strictly increasing by Lem. 4.18. Hence the maximum path length is $m - 1$.

## 4.2 Computing Dependencies by Directed Dependency Relations

Given a QBF in PCNF, the corresponding c-forest for $V_\exists$ is the basis for computing $D(x)$ for all $x \in V_\forall$.

**Definition 4.24** 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 $[y''] \rightsquigarrow^* [y]$.

In the c-forest class $h(x,[y])$ denotes the *smallest ancestor* of $[y]$ which is larger than $x$, hence $h(x,[y])$ is minimal with respect to the scope ordering.

**Definition 4.25** For a QBF $F$, $x \in V_\forall$ and the c-forest, the set of *descendants* $H^*(x)$ is defined as follows:

(i) $C(x) := \{C \in F \mid x \in C\}$

(ii) $V_C(x) := \{[y] \mid y \in V_{\exists,i}, i = \delta(x) \text{ and } y \in C \text{ for } C \in C(x)\}$

(iii) $H(x) := \{[z] \mid [z] = h(x,[y]) \text{ for } [y] \in V_C(x)\}$

(iv) $H^*(x) := \{[y] \mid [z] \rightsquigarrow^* [y] \text{ for } [z] \in H(x)\}$

From clauses containing $x$, classes of existential variables larger than $x$ are collected in $V_C(x)$. Note that if $[y] \in V_C(x)$ then $x \rightarrow_i^* y$ for $i = \delta(x)$. $H(x)$ contains smallest ancestors for classes in $V_C(x)$. $H^*(x)$ comprises descendants of classes in $H(x)$ and represents all connections of $x$ to existential variables larger than $x$.

**Lemma 4.26** *For $x \in V_\forall, i = \delta(x)$ : if $[y] \in H^*(x)$ then $x \rightarrow_i^* y$.*

**Proof.** Let $[y] \in H^*(x)$ for $x \in V_\forall$. Then by Def. 4.25 $[z] \rightsquigarrow^* [y]$ for $[z] \in H(x)$ where $[z] = h(x,[y'])$ for some $[y'] \in V_C(x)$. Then $x \rightarrow_i y'$ for $i = \delta(x)$ by definition of set $V_C(x)$ and also $x \rightarrow_i^* [y']$ by Thm. 4.16. By Def. 4.17 and since $\delta(x) < \delta(y')$ also $[x] \rightsquigarrow^* [y']$. Because $[z] = h(x,[y'])$ also $[z] \rightsquigarrow^* [y']$ by definition of function $h$. By Def. 4.24 $\delta(x) < \delta(z)$ and by $[x] \rightsquigarrow^* [y']$, $[z] \rightsquigarrow^* [y']$ and Lem. 4.21 also $[x] \rightsquigarrow^* [z]$. By Def. 4.17, $[x] \rightsquigarrow^* [z]$ and $[z] \rightsquigarrow^* [y]$ also $[x] \rightsquigarrow^* [y]$ and finally $x \rightarrow_i^* y$. $\square$

For $x \in V_\forall$, the set of descendants $H^*(x)$ in the c-forest exactly characterizes connections of $x$ to relevant (larger) existential variables. This is sufficient for computing $D(x)$ as stated in the following theorem.

**Theorem 4.27** *For* $x \in V_\forall : D(x) = \{y \mid y \in [z] \text{ for } [z] \in H^*(x)\}$.

**Proof.** Assume $x \in V_\forall$. Direction $\supseteq$ follows right from Lem. 4.26 and Thm. 4.20 since if $y \in [z]$ for $[z] \in H^*(x)$ then $x \to_i^* y$ for $i = \delta(x)$ by Lem. 4.26 and further $y \in D(x)$ by Thm. 4.20.

To show $\subseteq$, assume $y \in D(x)$ and $i = \delta(x)$. Then by Thm. 4.20 $x \to_i^* y$ and by Cor. 4.5 there is an $X$-path $P$ between $x$ and $y$ for $X = V_{\exists,i}$. Hence there are clauses $C_1, \ldots, C_k$ where $x \in C_1$ and $y \in C_k$. Since $P$ connects $x$ and $y$, also $x, y_1 \in C_1$ for some $y_1 \in V_{\exists,i}$ with $\delta(x) < \delta(y_1)$. Such $y_1$ always exists since by assumption the largest literal in a clause is existential. Thus $[y_1] \in V_C(x)$ and $[z_1] \in H(x)$ for $[z_1] = h(x, [y_1])$. Then by Def. 4.24, $[z_1] \rightsquigarrow^* [y_1]$. $P$ is also an $X$-path between $y_1$ and $y$ by $C_1, \ldots, C_k$, hence $y_1 \to_i^* y$ and $\delta(x) < \delta(y_1), \delta(x) < \delta(y)$. Let $w$ denote the smallest connecting variable in $P$ between $y_1, y$: $m = \delta(w) = min(\{\delta(v) \mid v \in C_i \cap C_{i+1} \cap X, 1 \leq i < k\})$. Since $m$ is minimal, also $y_1 \to_m^* w$, $w \to_m^* y$ and by Lem. 4.10 $w \to_m^* y_1$. By Def. 4.17 and since $m = \delta(w)$, also $[w] \rightsquigarrow^* [y_1]$, $[w] \rightsquigarrow^* [y]$. By Lem. 4.21, $[z_1] \rightsquigarrow^* [y_1]$ and $[w] \rightsquigarrow^* [y_1]$, also $[z_1] \rightsquigarrow^* [w]$. Then by $[z_1] \rightsquigarrow^* [w]$, $[w] \rightsquigarrow^* [y]$ and transitivity also $[z_1] \rightsquigarrow^* [y]$, hence $[y] \in H^*(x)$ because $[z_1] \in H(x)$. □

By Thm 4.27 and Def. 4.25 members of classes in $H^*(x)$ exactly correspond to $D(x)$. Computing $D(x)$ from a c-forest therefore does not require searching since subtrees rooted at classes 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 4.28** Fig. 1 shows a c-forest (dotted edges) and sets $H(x)$ (solid edges). Variables $\mathsf{e3}, \mathsf{e4}$ and $\mathsf{e12}, \mathsf{e13}$ are in one class, respectively (horizontal edges). According to Def. 4.25, $C(\mathsf{a1}) = C(\mathsf{a6}) = \{(\mathsf{a1}, \mathsf{a6}, \mathsf{e8}, \mathsf{e13})\}$, $V_C(\mathsf{a1}) = V_C(\mathsf{a6}) = \{[\mathsf{e8}], [\mathsf{e13}]\}$, $H(\mathsf{a1}) = \{[\mathsf{e3}]\}$, $H^*(\mathsf{a1}) = \{[\mathsf{e3}], [\mathsf{e8}], [\mathsf{e10}], [\mathsf{e13}]\}$, $H(\mathsf{a6}) = \{[\mathsf{e8}]\}$, $H^*(\mathsf{a6}) = \{[\mathsf{e8}], [\mathsf{e13}]\}$, $D(\mathsf{a1}) = \{\mathsf{e3}, \mathsf{e4}, \mathsf{e8}, \mathsf{e10}, \mathsf{e12}, \mathsf{e13}\}$ and $D(\mathsf{a6}) = \{\mathsf{e8}, \mathsf{e12}, \mathsf{e13}\}$. Note that the path from $[\mathsf{e8}]$ to $[\mathsf{e13}]$ is shared between variables $\mathsf{a1}$ and $\mathsf{a6}$.

## 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 [25]. Tab. 1 shows experimental results on structured formulae from QBF competitions [16].[3]

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 $x \in V_\forall$

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

| | *2005* | *2006* | *2007* | *2008* |
|---|---|---|---|---|
| *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 formulae from QBF competitions 2005 to 2008 [16]. Columns labelled 2005 to 2008 denote formulae sets from QBFEVAL competitions of the respective year. See section 5 for comments.

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 and represent 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. This relation 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 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] Aho, A. V., M. R. Garey and J. D. Ullman, *The Transitive Reduction of a Directed Graph*, SIAM J. Comput. **1** (1972), pp. 131–137.

[2] Ayari, A. and D. A. Basin, *QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers*, in: M. Aagaard and J. W. O'Leary, editors, *FMCAD*, Lecture Notes in Computer Science **2517** (2002), pp. 187–201.

[3] Bacchus, F. and T. Walsh, editors, "Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings," Lecture Notes in Computer Science **3569**, Springer, 2005.

[4] Benedetti, M., *Quantifier Trees for QBFs*, in: Bacchus and Walsh [3], pp. 378–385.

[5] Benedetti, M., *sKizzo: A Suite to Evaluate and Certify QBFs*, in: R. Nieuwenhuis, editor, *CADE*, Lecture Notes in Computer Science **3632** (2005), pp. 369–376.

[6] Benedetti, M. and H. Mangassarian, *QBF-Based Formal Verification: Experience and Perspectives*, JSAT **5** (2008), pp. 133–191.

[7] Biere, A., *Resolve and Expand*, in: H. H. Hoos and D. G. Mitchell, editors, *SAT (Selected Papers)*, Lecture Notes in Computer Science **3542** (2004), pp. 59–70.

[8] Biere, A., A. Cimatti, E. M. Clarke and Y. Zhu, *Symbolic Model Checking without BDDs*, in: R. Cleaveland, editor, *TACAS*, Lecture Notes in Computer Science **1579** (1999), pp. 193–207.

[9] Bubeck, U. and H. K. Büning, *Bounded Universal Expansion for Preprocessing QBF*, in: J. Marques-Silva and K. A. Sakallah, editors, *SAT*, Lecture Notes in Computer Science **4501** (2007), pp. 244–257.

[10] Büning, H. K., M. Karpinski and A. Flögel, *Resolution for Quantified Boolean Formulas*, Inf. Comput. **117** (1995), pp. 12–18.

[11] Cadoli, M., A. Giovanardi and M. Schaerf, *An Algorithm to Evaluate Quantified Boolean Formulae*, in: *AAAI/IAAI*, 1998, pp. 262–267.

[12] Davis, M., G. Logemann and D. Loveland, *A Machine Program for Theorem-proving*, Commun. ACM **5** (1962), pp. 394–397.

[13] Dershowitz, N., Z. Hanna and J. Katz, *Bounded Model Checking with QBF*, in: Bacchus and Walsh [3], pp. 408–414.

[14] Egly, U., M. Seidl, H. Tompits, S. Woltran and M. Zolda, *Comparing Different Prenexing Strategies for Quantified Boolean Formulas*, in: E. Giunchiglia and A. Tacchella, editors, *SAT*, Lecture Notes in Computer Science **2919** (2003), pp. 214–228.

[15] Egly, U., H. Tompits and S. Woltran, *On Quantifier Shifting for Quantified Boolean Formulas*, in: *Proc. SAT'02 Workshop on Theory and Applications on QBFs*, 2002, pp. 48–61.

[16] Giunchiglia, E., M. Narizzano and A. Tacchella, *QBF Solver Evaluation Portal* (2001), `http://www.qbflib.org/index_eval.php`.

[17] Giunchiglia, E., M. Narizzano and A. Tacchella, *Backjumping for Quantified Boolean Logic satisfiability*, Artif. Intell. **145** (2003), pp. 99–120.

[18] Giunchiglia, E., M. Narizzano and A. Tacchella, *Quantifier Structure in Search-Based Procedures for QBFs*, IEEE Trans. on CAD of Integrated Circuits and Systems **26** (2007), pp. 497–507.

[19] Jussila, T. and A. Biere, *Compressing BMC Encodings with QBF*, ENTCS **174** (2007), pp. 45–56.

[20] Lonsing, F. and A. Biere, *Nenofex: Expanding NNF for QBF Solving*, in: H. K. Büning and X. Zhao, editors, *SAT*, Lecture Notes in Computer Science **4996** (2008), pp. 196–210.

[21] Lonsing, F. and A. Biere, *A Compact Representation for Syntactic Dependencies in QBFs*, in: O. Kullmann, editor, *SAT*, Lecture Notes in Computer Science **5584** (2009), pp. 398–411.

[22] QBFLIB, *QDIMACS Standard v1.1*, `http://www.qbflib.org/qdimacs.html`.

[23] Samer, M. and S. Szeider, *Backdoor Sets of Quantified Boolean Formulas*, Journal of Automated Reasoning (JAR) **42** (2009), pp. 77–97.

[24] Stockmeyer, L. J. and A. R. Meyer, *Word Problems Requiring Exponential Time: Preliminary Report*, in: *STOC* (1973), pp. 1–9.

[25] Tarjan, R. E., *Efficiency of a Good But Not Linear Set Union Algorithm*, J. ACM **22** (1975), pp. 215–225.

[26] Zhang, L. and S. Malik, *Conflict driven learning in a quantified Boolean Satisfiability solver*, in: L. T. Pileggi and A. Kuehlmann, editors, *ICCAD* (2002), pp. 442–449.