A Compact Representation for Syntactic Dependencies in QBFs

Florian Lonsing and Armin Biere

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
florian.lonsing@jku.at
http://fmv.jku.at

SAT'09
June 30 - July 3, 2009
Swansea, Wales, United Kingdom

JOHANNES KEPLER UNIVERSITY LINZ | JKU

TNF

Overview

QBF: Quantified Boolean Formulae.

QDPLL: DPLL-like QBF solvers, formulae in prenex CNF.

- decision order must respect "quantification order".

Example (Dependencies in QBF)

$\forall x \exists y. (x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable. Value of y depends on value of x.
$\rightarrow$ erroneously conclude unsatisfiability if y is assigned before x.

Our Results:

- given: syntactic dependency relation D.
- static and compact dependency graph (DAG) representing D.
- graph is applicable to QBF solvers of any kind.
- in QDPLL: find assignable variables before decision-making.
- experiments: structured formulae from QBFEVAL 2005 - 2008.

Preliminaries

QBFs in Prenex CNF: $S_1 \ldots S_n. \phi$

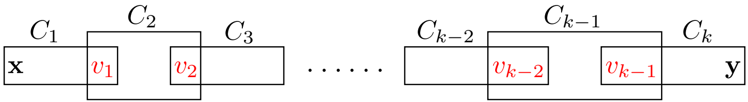
- ϕ in CNF and quantifier-free, quantified variables $V = V_\exists \cup V_\forall$.
- scopes $S_1 < \ldots < S_n$, ordered by nesting $\delta(S_i) = i$, type $q(S_i) \in \{\forall, \exists\}$.

Dependency Schemes [SamerSzeider-JAR'09]:

- relation $D \subseteq (V_\exists \times V_\forall) \cup (V_\forall \times V_\exists)$.
- $y \in D(x)$: "y depends on x", i.e. assign x before y in QDPLL.
- $|D_1| < |D_2|$: D_1 less restrictive, i.e. more freedom for decisions in QDPLL.

Example (Trivial Dependency Scheme)

D^{triv}: $y \in D^{\text{triv}}(x) \Leftrightarrow \delta(x) < \delta(y)$ and $q(x) \neq q(y)$.



Definition (X-path)

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

- For $x \in V$: if $q(x) = \exists$ then $\overline{q(x)} := \forall$ and $\overline{q(x)} := \exists$ otherwise.
- For a QBF and $q \in \{\exists, \forall\}$: $V_{q,i} := \{y \in V_q \mid i \leq \delta(y)\}$.

Definition (Standard Dependency Scheme)

For $x \in V, i = \delta(x) + 1$:
$D^{\text{std}}(x) = \{y \in V_{\overline{q(x)},i} \mid \text{there is an } X\text{-path between } x \text{ and } y \text{ for } X = V_{\exists,i}\}$.

- $D^{\text{std}}(x)$ contains all differently quantified, *larger* y which are connected to x over *existential* variables *larger* than x.
- Observe: $|D^{std}| < |D^{triv}|$.

Goal: A Compact Graph for D^{std}

In Practice: computing *full* D^{std} in $O(|V|.|\phi|)$ time.

- traversing clauses in ϕ for *each* $x \in V$.
- too expensive to be done dynamically at decision points within QDPLL.

Our Work:

- *static* and *compact* graph representation (DAG) for D^{std}.
- graph is built once, serves as over-approximation for exact D^{std}.
- classes of variables represent connection information.
- connection information is shared between variables.

Towards a Graph Representation: Connections

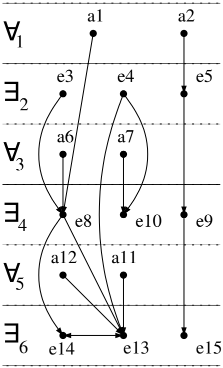
Definition (Variable Connection)

For $x, y \in V$, x is *connected* to y wrt. scope S_i ($x \rightarrow_i y$) iff. $y \in V_\exists$, $i \leq \delta(y)$ and $x, y \in C$ for $C \in \phi$. Relation $\rightarrow_i^*$ is the refl. trans. closure of $\rightarrow_i$.

Example (ongoing)

i	$q(S_i)$	S_i	
1	$\forall$	a1, a2	(a2, e5, e9)
2	$\exists$	e3, e4, e5	(e5, e9, e15)
3	$\forall$	a6, a7	(e3, e8, e13)
4	$\exists$	e8, e9, e10	(e4, a7, e10)
5	$\forall$	a11, a12	(e4, e13, e14)
6	$\exists$	e13, e14, e15	(a1, a6, e8, e14)
			(a11, a12, e13)

- trans. edges not shown.
- e3 $\rightarrow_4$ e8 but e3 $\not\rightarrow_5$ e8.
- e3 $\rightarrow_2^*$ e14 and also e14 $\rightarrow_2^*$ e3.



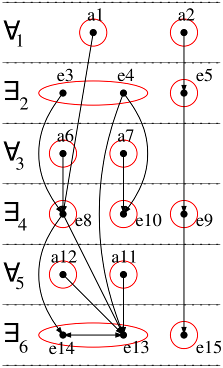
From Connections to Classes

Definition (Equivalence)

For $x, y \in V$, x is *equivalent* to y ($x \approx y$) iff. either (1) $x = y$ or (2) $q(x) = q(y) = \exists$, $\delta(x) = \delta(y) = i$ and $x \rightarrow_i^* y$.

Example (continued)

- e3 $\approx$ e4 since $q(\text{e3}) = q(\text{e4}) = \exists$, $\delta(\text{e3}) = \delta(\text{e4}) = 2$ and e3 $\rightarrow_2^*$ e4.
- e5 $\not\approx$ e4 because e5 $\not\rightarrow_2^*$ e4.
- trivially a11 $\approx$ a11 and e3 $\not\approx$ e14.



- $\rightarrow_i^*$ on $\approx$ potentially more compact.
- $[x] \rightarrow_i^* [y]$: connection between classes.

Representing Class Connections: Theory

Goal: compact representation of *all* existential class connections for D^{std}.

Problem: with $\rightarrow_i^*$ on $\approx$ still need to search for connected classes.

Definition (Directed Connection)

For $x \in V, y \in V_\exists$, $[x] \rightsquigarrow^* [y]$ iff. $\delta(x) \leq \delta(y)$ and $x \rightarrow_i^* y$ for $i = \delta(x)$.
Relation $\rightsquigarrow$ is the refl. trans. reduction of $\rightsquigarrow^*$.

- $[x] \rightsquigarrow^* [y]$ respects scope ordering, excludes variables smaller than x.

Lemma (Connection Forest, C-Forest)

For $V_\exists$, $\rightsquigarrow$ can be represented as a forest.

C-Forest: compact representation of all *existential* connections.

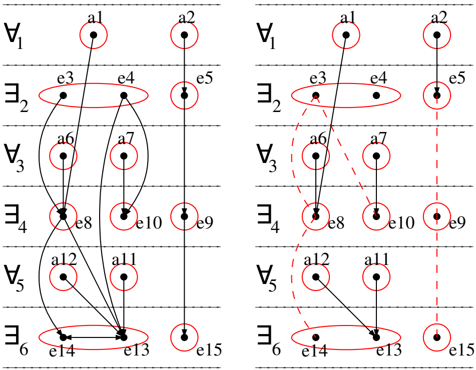
- no more searching: classes are connected to all of their descendants.

Representing Class Connections: Example

Example (continued)

Right figure:

- $\leadsto$ on $V_\exists$: c-forest.
- [e3] $\leadsto$ [e8]
- [e8] $\leadsto$ [e14]
- [e3] $\leadsto^*$ [e14]
- also [e4] $\leadsto^*$ [e14] since [e4] $=$ [e3]



Practical Problem: how to find connected descendants of x in c-forest?

Essentially: need set of “root classes” for each $x \in V$.

- descendants of root classes *exactly* represent all connections of x.
- computing D^{std}: c-forest + root classes.

Root Classes and Descendants

Finding Root Classes: finding smallest ancestors in c-forest.

Definition (Smallest Ancestor)

For $y \in V_\exists, i \leq \delta(y)$ and the c-forest, $h(i, [y]) = [y']$ denotes the smallest ancestor of $[y]$ in the c-forest such that $i \leq \delta(y')$.

Definition (Root Classes)

For $x \in V$, $H_i(x) := \{[z] \mid [z] = h(i, [y])$ for $[y]$ where $x \rightarrow_i y\}$ is the set of root classes of x with respect to scope S_i.

- Finding root classes $H_i(x)$ starting from clauses containing x.

Definition (Root Class Descendants)

For $x \in V$, $H_i^*(x) := \{[y] \mid [z] \rightsquigarrow^* [y]$ for $[z] \in H_i(x)\}$ is the set of root class descendants of x with respect to scope S_i.

- Sets H_i^* are sufficient for computing D^{std} from c-forest.

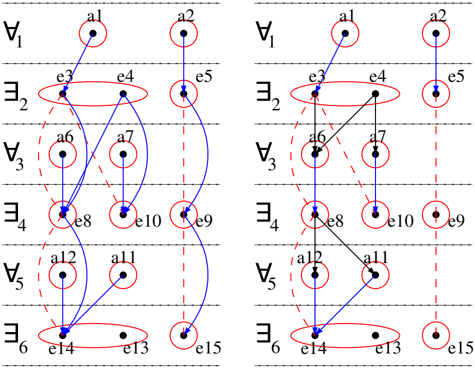
Completing the Dependency Graph

Theorem (D^{std} by Checking Root Class Decendants)

For $x \in V, i = \delta(x) + 1$:
$D^{std}(x) = \{y \in V_{\overline{q(x)},i} \mid H_i^*(x) \cap H_j^*(y) \neq \emptyset$ *for* $j = \delta(y)\}$.

Example (continued)

- left figure: blue edges $\rightarrow$ as sets $H_i(x)$ for $i = \delta(x) + 1$, $x \in V$.
- right figure: D^{std} graph
- implicitly:
  e15 $\in D^{std}$(a2)
  e13 $\in D^{std}$(a1)
  e13 $\in D^{std}$(a11)



Graph for D^{std}: c-forest as core

- already present: for $x \in V_\forall$, $i = \delta(x) + 1$: $D^{std}(x) = H_i^*(x)$
- for $x \in V_\exists$: insert edges to represent $D^{std}(x)$

Experimental Results

	QBFEVAL'05	QBFEVAL'06	QBFEVAL'07	QBFEVAL'08
size	211	216	1136	3328
total time	7.94	1.35	227.05	300.31
max. time	0.58	0.03	7.96	8.11
avg. time	**0.04**	**0.01**	**0.2**	**0.09**
$x \in V_\forall$				
max. $\lvert D^{\mathrm{std}}(x)\rvert$	256535	9993	2177280	2177280
avg. $\lvert D^{\mathrm{std}}(x)\rvert$	82055.87	4794.60	33447.6	19807
max. $\lvert H_i(x)\rvert$	256	1	518	518
avg. $\lvert H_i(x)\rvert$	3.26	0.98	2.02	1.14
max. $\lvert H_i^*(x)\rvert$	797	5	797	1872
avg. $\lvert H_i^*(x)\rvert$	19.51	1.12	39.06	8.24
avg. $\frac{\lvert\{[y]\in D^{\mathrm{std}}(x)\}\rvert}{\lvert\{y\in D^{\mathrm{std}}(x)\}\rvert}$	**3.44%**	**0.04%**	**6.42%**	**1.21%**
$x \in V_\exists$				
max. $\lvert D^{\mathrm{std}}(x)\rvert$	5040	440	5040	22696
avg. $\lvert D^{\mathrm{std}}(x)\rvert$	12.76	2.98	3.24	4
max. $\lvert H_i(x)\rvert$	24	7	490	490
avg. $\lvert H_i(x)\rvert$	0.14	0.13	0.17	0.13
max. $\lvert H_i^*(x)\rvert$	797	7	797	1872
avg. $\lvert H_i^*(x)\rvert$	5.16	0.16	1.32	1.31
avg. $\frac{\lvert\{[y]\in D^{\mathrm{std}}(x)\}\rvert}{\lvert\{y\in D^{\mathrm{std}}(x)\}\rvert}$	**2.37%**	**0.4%**	**2.76%**	**2.09%**
classes per variables	**10.96%**	**4.99%**	**11.45%**	**7.11%**

QDPLL for QBF: quantification order matters.

- limited freedom for decisions.

Dependency Schemes:

- dependency relations $D \subseteq (V_\exists \times V_\forall) \cup (V_\forall \times V_\exists)$.
- $y \in D(x)$: assign x before y in QDPLL.
- Standard Dependency Scheme D^{std}: based on variable connections.

Achievements: static, compact graph representation D^{std} for QBF in PCNF.

- compactness: connection relation on equivalence classes.
- c-forest: sharing connection information.
- two orders of magnitude more compact than simple graph.

Ongoing and Future Work:

- integration into QDPLL: maintaing top-down "decision frontier".
- comparing dependency schemes in QDPLL.

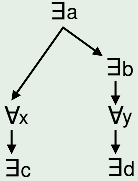
[Appendix] Mini-scoping is non-deterministic

Example

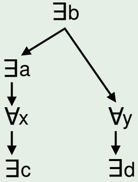
$$\exists a, b \forall x, y \exists c, d.\ (a \vee x \vee c) \wedge (a \vee b) \wedge (b \vee d) \wedge (y \vee d)$$

After minimizing $\exists c, \exists d, \forall x$ and $\forall y$, *non-deterministic* choice:

1. minimize $\exists b$ before $\exists a$



2. minimize $\exists a$ before $\exists b$



Extract D^{tree} from parse tree (descendants):

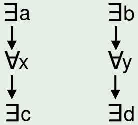
1. $D^{\text{tree}} = \{(a, x), (x, c), (a, y), (b, y), (y, d)\}$
2. $D^{\text{tree}} = \{(b, x), (a, x), (x, c), (b, y), (y, d)\}$

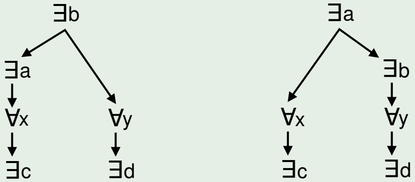
[Appendix] D^{std} less restrictive than D^{tree}

Example (continued)

$$\exists a, b \forall x, y \exists c, d.\ (a \vee x \vee c) \wedge (a \vee b) \wedge (b \vee d) \wedge (y \vee d)$$

By D^{std} we get:





Either $(a, y) \in D^{\text{tree}}$ or $(b, x) \in D^{\text{tree}}$ but *both* $(a, y) \notin D^{\text{std}}$ *and* $(b, x) \notin D^{\text{std}}$.

Theorem

For $x \in V, i = \delta(x) + 1$:

$$D^{\text{std}}(x) = \{y \in V_{\overline{q(x)},i} \mid \exists w \in V_{\exists,i} : x \rightarrow_i^* w \text{ and } y \rightarrow_i^* w\} \quad (1)$$

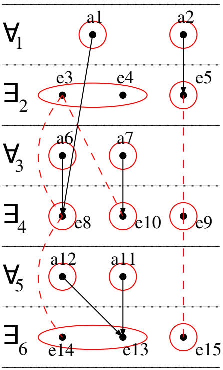
$$= \{y \in V_{\overline{q(x)},i} \mid \exists w \in V_{\exists,i} : x \rightarrow_i^* [w] \text{ and } [y] \rightarrow_i^* [w]\} \quad (2)$$

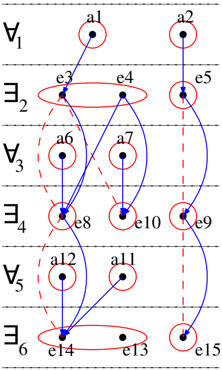
$$= \{y \in V_{\overline{q(x)},i} \mid \exists w \in V_{\exists,i} : x \rightarrow_i^* [w] \text{ and } [y] \rightsquigarrow^* [w]\} \quad (3)$$

Example (continued)

Blue edges $\rightarrow$: sets $H_i(x)$ for $i = \delta(x) + 1$, $x \in V$.

i	$q(S_i)$	S_i	
1	$\forall$	a1, a2	(a2, e5, e9)
2	$\exists$	e3, e4, e5	(e5, e9, e15)
3	$\forall$	a6, a7	(e3, e8, e13)
4	$\exists$	e8, e9, e10	(e4, a7, e10)
5	$\forall$	a11, a12	(e4, e13, e14)
6	$\exists$	e13, e14, e15	(a1, a6, e8, e14)
			(a11, a12, e13)



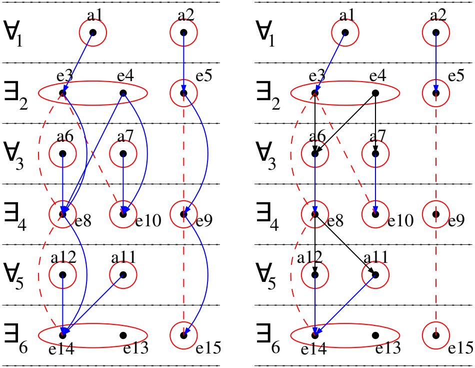


Theorem (D^{std} by Checking Root Class Decendants)

For $x \in V, i = \delta(x) + 1$:
$D^{std}(x) = \{y \in V_{\overline{q(x)},i} \mid H_i^*(x) \cap H_j^*(y) \neq \emptyset \text{ for } j = \delta(y)\}$.

Example (continued)

- right figure: D^{std} graph
- implicitly:
  e15 $\in D^{std}$(a2)
  e13 $\in D^{std}$(a1)
  e13 $\in D^{std}$(a11)



Graph for D^{std}:

- already present: for $x \in V_\forall$, $i = \delta(x) + 1$: $D^{std}(x) = H_i^*(x)$
- for $x \in V_\exists$: insert (non-transitive) edges to represent $D^{std}(x)$

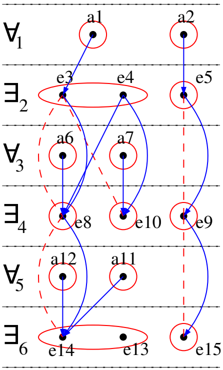
Theorem (D^{std} by Checking Root Class Decendants)

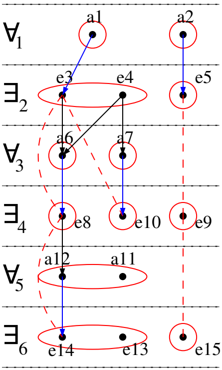
For $x \in V$, $i = \delta(x) + 1$:
$D^{\text{std}}(x) = \{y \in V_{\overline{q(x)},i} \mid H_i^*(x) \cap H_j^*(y) \neq \emptyset$ *for* $j = \delta(y)\}$.

Example (continued)

- right figure: D^{std} graph
- implicitly:
  e15 $\in D^{\text{std}}$(a2)
  e13 $\in D^{\text{std}}$(a1)
  e13 $\in D^{\text{std}}$(a11)
  a11 $\in D^{\text{std}}$(e8)

Optimization: merging universal classes [a11], [a12] with same H_i.





Graph for D^{std}:

- already present: for $x \in V_\forall$, $i = \delta(x) + 1$: $D^{\text{std}}(x) = H_i^*(x)$
- for $x \in V_\exists$: insert (non-transitive) edges to represent $D^{\text{std}}(x)$

[Appendix] Experimental Results

	QBFEVAL'05	QBFEVAL'06	QBFEVAL'07	QBFEVAL'08
size	211	216	1136	3328
total time	7.94	1.35	227.05	300.31
max. time	0.58	0.03	7.96	8.11
avg. time	0.04	0.01	0.2	0.09
$x \in V_\forall$				
max. $\lvert D^{std}(x) \rvert$	256535	9993	2177280	2177280
avg. $\lvert D^{std}(x) \rvert$	82055.87	4794.60	33447.6	19807
max. $\lvert H_i(x) \rvert$	256	1	518	518
avg. $\lvert H_i(x) \rvert$	3.26	0.98	2.02	1.14
max. $\lvert H_i^*(x) \rvert$	797	5	797	1872
avg. $\lvert H_i^*(x) \rvert$	19.51	1.12	39.06	8.24
avg. $\frac{\lvert \{[y] \in D^{std}(x)\} \rvert}{\lvert \{y \in D^{std}(x)\} \rvert}$	3.44%	0.04%	6.42%	1.21%
classes per variables	28.2%	10.23%	40.31%	21.29%
$x \in V_\exists$				
max. $\lvert D^{std}(x) \rvert$	5040	440	5040	22696
avg. $\lvert D^{std}(x) \rvert$	12.76	2.98	3.24	4
max. $\lvert H_i(x) \rvert$	24	7	490	490
avg. $\lvert H_i(x) \rvert$	0.14	0.13	0.17	0.13
max. $\lvert H_i^*(x) \rvert$	797	7	797	1872
avg. $\lvert H_i^*(x) \rvert$	5.16	0.16	1.32	1.31
avg. $\frac{\lvert \{[y] \in D^{std}(x)\} \rvert}{\lvert \{y \in D^{std}(x)\} \rvert}$	2.37%	0.4%	2.76%	2.09%
classes per variables	10.96%	4.99%	11.45%	7.11%