Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers

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

MEMICS'08
November 14 - 16, 2008
Znojmo, Czech Republic

JOHANNES KEPLER UNIVERSITY LINZ | JKU

TNF

Quantified Boolean Formulae (QBF)

- propositional formula, $\forall/\exists$ quantification
- PSPACE-completeness: natural modelling language

Variable Dependencies in QBF

- two types: $\forall\exists$ and $\exists\forall$
- influence on decision procedures for QBF
- our focus: expansion-based solvers, case $\forall\exists$

Results

- given: syntactic dependency relation D for case $\forall\exists$
- average-case compact representation for directed variant of D
  - equivalence relation on $\exists$-variables
  - efficient retrieval of $\exists$-dependencies for $\forall$-variables
- experimental results: benchmarks from QBF competitions 2005 - 2008

Quantified Boolean Formulae (QBF): $S_1 \ldots S_n\ \phi$

- prenex conjunctive normal form (PCNF), e.g. $\forall x_1 \exists x_2 x_3\ \phi$
- propositional formula ϕ in CNF over variables $V = V_\forall \cup V_\exists$
- quantifier prefix $S_1 \ldots S_n$
  - scopes S_i, $q(S_i) \in \{\forall, \exists\}$: quantified variables
  - linear orderings: $\delta(S_1) = 1 < \ldots < \delta(S_n) = n$, $\delta(x) = \delta(S_i)$ if $x \in S_i$

Variable Dependencies in QBF

- $\delta(S_1) < \ldots < \delta(S_n)$: often pessimistic
- dependency computation in practice: optimality vs. efficiency
  - polynomial time: syntactic analysis of formula

Example

$$\forall x \exists y\ (\neg x \vee y) \wedge (x \vee \neg y) \text{ is satisfiable}$$

Value of y depends on x: $x = \top \rightarrow y = \top$, $x = \bot \rightarrow y = \bot$

Motivation

Universal Expansion: $\forall x\ \phi \equiv \phi[x/0] \wedge \phi[x/1]$

- existential dependencies for $x \in V_\forall : D(x) \subseteq \{y \in V_\exists \mid \delta(x) < \delta(y)\}$

Computing $D(x)$ via Syntactic Connection Relation

- $y, z \in V$: y locally connected to z if $y, z \in C$ for clause $C \in \phi$
- inf.: $y \in D(x)$ if x transitively connected to y via common clauses
- recursive computation: $O(|\phi|)$ for one $x \in V_\forall$

Goal: Avoiding Recomputation of Connection Relation

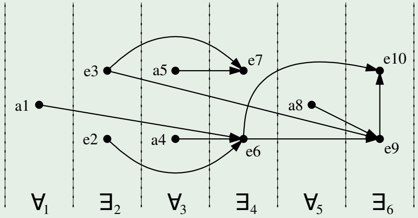
- building a global connection relation wrt. common clauses
- idea: extract once from ϕ, exploiting shared parts for all $x \in V_\forall$
- compact representation and retrieval of $D(x)$, $|D(x)|$

Definition (local dependence/connection)

For $x, y \in V$: $x \to_i y \iff q(y) = \exists$ and $x, y \in C$, $C \in \phi$ and $\delta(y) \geq i$.
Connecting sets of variables and clauses by refl. and trans. closure $\to_i^*$.

"connection": write $x \sim_i y$ if $q(x) = q(y) = \exists$ and $x \to_i^* y$.

Example



- trans. edges not shown
- `a1` $\to_1$ `e6`, `e6` $\to_1$ `e9`
- `e9` $\to_1$ `e6`
- `a1` $\to_1^*$ `e7` by `e6`, `e9`, `e3`

(Application) For $x \in V_\forall, i = \delta(x) : D(x) = \{y \in V_\exists \mid x \to_i^* y\}$.

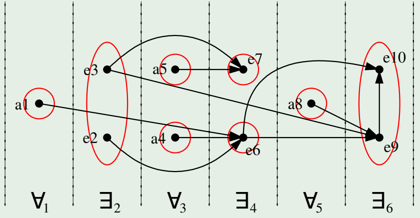
Definition (equivalence relation)

For $x, y \in V$: $x \approx y \iff x = y$ and $q(x) = \forall$ or $\delta(x) = \delta(y) = i, q(x) = q(y) = \exists$ and $x \sim_i y$. $[x]$ denotes the class of x.

Theorem (compatibility of $\rightarrow_i^*$ with $\approx$)

For $x, y \in V$: $x \rightarrow_i^* y \iff \forall x' \in [x], y' \in [y] : x' \rightarrow_i^* y'$.

Example (continued)



- partition of scopes
- e2 $\approx$ e3
- e3 $\not\approx$ e9
- e6 $\not\approx$ e7 since e6 $\not\sim_4$ e7

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

Definition (directed dependence/connection)

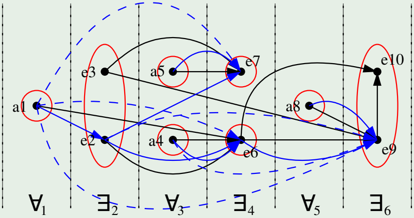
For $x, y \in V$: $[x] \rightsquigarrow^* [y] \iff \delta(x) \leq \delta(y)$ and $x \rightarrow_i^* y$ for $i = \delta(x)$.

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

Theorem (computing dependency sets)

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

Example (continued)



- $\rightsquigarrow^*$ defined on classes
- e2 $\rightsquigarrow^*$ e9, but e9 $\not\rightsquigarrow^*$ e2
- dashed: transitive edges
- solid: transitive reduction

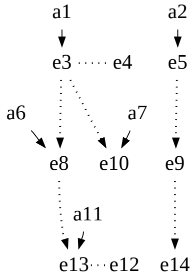
An Efficient Tree Representation for $\rightsquigarrow^*$

Lemma

For $\rightsquigarrow^*$ on $V_\exists$, the transitive reduction $\rightsquigarrow$ can be represented as forest.

Connection Forest of a QBF

- representation of global, shared connection relation for $V_\exists$
- for $y, z \in V_\exists$: edge $([y],[z]) \iff [y] \rightsquigarrow [z]$
- for $y, z \in V_\exists$: path from $[y]$ to $[z] \iff [y] \rightsquigarrow^* [z]$



Augmented Connection Forest

- additionally: set of “entry points” $H(x)$ for all $x \in V_\forall$
- $H(x)$ derived from clauses containing literals of x

Computing $D(x)$ by Connection Forest

1. collect descendant classes: $H^*(x) := \{[y] \mid [z] \rightsquigarrow^* [y], [z] \in H(x)\}$
2. collect members of descendants: $D(x) = \{z \mid z \in [y], [y] \in H^*(x)\}$

Experimental Results

	QBFEVAL'05	*QBFEVAL'06*	*QBFEVAL'07*	*QBFEVAL'08*
size	211	216	1136	3328
max. $\lvert H^*(x)\rvert$ *avg.* $\lvert H^*(x)\rvert$	797 19.51	5 1.21	797 39.07	1872 8.24
max. $\lvert D(x)\rvert$ *avg.* $\lvert D(x)\rvert$	256535 82055.87	9993 4794.60	2177280 33447.6	2177280 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 %

- structured QBF formulae from QBF competitions 2005 - 2008
- comparing forest representation with $|D(x)|$
- number of successors $|H^*(x)|$ much smaller than $|D(x)|$
- line $\approx_\exists$: number of $\exists$-classes per $\exists$-variable in whole formula set
- compression by $\approx$: few, but large classes for $S_i, q(S_i) = \exists$

Variable Dependencies in QBF

- influence solver performance
- common approach: syntactic connection relation (connecting clauses)
- focus: expansion-based solvers, $\forall\exists$ dependencies

Augmented Connection Forests

- directed version of connection relation, equivalence relation on $V_\exists$
- average-case compact representation
- sharing connection information between all $x \in V_\forall$
- computation of $D(x)$, $|D(x)|$ for all $x \in V_\forall$

Future Work

- dynamic vs. static version
- extension to $\exists\forall$ dependencies
- combination with search-based solvers