QBF in Formal Verification

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

Alpine Verification Meeting 2005

Lausanne, Switzerland
6. October 2006
Quantified Boolean Formulae (QBF)

- propositional logic \text{(SAT} \subseteq \text{QBF)}
  - constants 0, 1
  - operators \&, \neg, \rightarrow, \leftrightarrow, \ldots
  - variables \(x, y, \ldots\) over boolean domain \(\mathbb{B} = \{0, 1\}\)

- quantifiers over boolean variables
  - valid \(\forall x[\exists y[x \leftrightarrow y]]\) (read \leftrightarrow as \(=\))
  - invalid \(\exists x[\forall y[x \leftrightarrow y]]\)
• semantics given as **expansion** of quantifiers

\[ \exists x[f] \equiv f[0/x] \lor f[1/x] \quad \forall x[f] \equiv f[0/x] \land f[1/x] \]

• expansion as translation from SAT to QBF is exponential
  – SAT problems have only existential quantifiers
  – expansion of universal quantifiers doubles formula size

• most likely no polynomial translation from SAT to QBF
  – otherwise \( \text{PSPACE} = \text{NP} \)
Tic-Tac-Toe

\begin{center}
\begin{tabular}{cccc}
\textbf{$s_0$} & \textbf{$s_1$} & \textbf{$s_2$} & \textbf{$s_3$} \\
\includegraphics[width=0.2\textwidth]{s0.png} & \includegraphics[width=0.2\textwidth]{s1.png} & \includegraphics[width=0.2\textwidth]{s2.png} & \includegraphics[width=0.2\textwidth]{s3.png} \\
\textbf{$s_4$} & \textbf{$s_5$} & \textbf{$s_6$} & \textbf{$s_7$} \\
\includegraphics[width=0.2\textwidth]{s4.png} & \includegraphics[width=0.2\textwidth]{s5.png} & \includegraphics[width=0.2\textwidth]{s6.png} & \includegraphics[width=0.2\textwidth]{s7.png} \\
\textbf{$s_8$} & \textbf{$s_9$} & \textbf{$s_{10}$} & \textbf{$s_{11}$} \\
\includegraphics[width=0.2\textwidth]{s8.png} & \includegraphics[width=0.2\textwidth]{s9.png} & \includegraphics[width=0.2\textwidth]{s10.png} & \includegraphics[width=0.2\textwidth]{s11.png} \\
\end{tabular}
\end{center}
\[ \neg \forall s_0[empty(s_0) \rightarrow \exists x_1[circle(s_0, x_1, s_1) \land \forall y_2[cross(s_1, y_2, s_2) \rightarrow \exists x_3[circle(s_2, x_3, s_3) \land \forall y_4[cross(s_3, y_4, s_4) \rightarrow \exists x_5[circle(s_4, x_5, s_5) \land \forall y_6[cross(s_5, y_6, s_6) \rightarrow \exists x_7[circle(s_6, x_7, s_7) \land \forall y_8[cross(s_7, y_8, s_8) \rightarrow \exists x_9[circle(s_8, x_9, s_9) \land \text{win}_{\text{circle}}(s_9)]]]]]]]]] \]
Model Checking

• explicit model checking  [ClarkeEmerson’82], [Holzmann’91]
  – program presented symbolically  (no transition matrix)
  – traversed state space represented explicitly
  – e.g. reached states are explicitly saved bit for bit in hash table
  ⇒ State Explosion Problem  (state space exponential in program size)

• symbolic model checking  [McMillan Thesis’93], [CoudertMadre’89]
  – use symbolic representations for sets of states
  – originally with Binary Decision Diagrams  [Bryant’86]
  – Bounded Model Checking using SAT  [BiereCimattiClarkeZhu’99]
Forward Fixpoint Algorithm: Initial and Bad States

A. Biere, FMV, JKU Linz
Forward Fixpoint Algorithm: Step 1
Forward Fixpoint Algorithm: Step 2

A. Biere, FMV, JKU Linz
Forward Fixpoint Algorithm: Step 3

A. Biere, FMV, JKU Linz
Forward Fixpoint Algorithm:  Bad State Reached

A. Biere, FMV, JKU Linz
Forward Fixpoint Algorithm: Termination, No Bad State Reachable

A. Biere, FMV, JKU Linz
Forward Least Fixpoint Algorithm for Model Checking Safety

A. Biere, FMV, JKU Linz

initial states $I$, transition relation $T$, bad states $B$

\[
\text{model-check}^{\textit{forward}} (I, T, B)
\]

$S_C = \emptyset; S_N = I$;

\textbf{while} $S_C \neq S_N$ \textbf{do}

\hspace{1em} \textbf{if} $B \cap S_N \neq \emptyset$ \textbf{then}

\hspace{2em} \textbf{return} "found error trace to bad states";

\hspace{1em} $S_C = S_N$;

\hspace{1em} $S_N = S_C \cup \text{Img}(S_C)$;

\textbf{done};

\textbf{return} "no bad state reachable";

symbolic model checking represents set of states in this BFS symbolically
Unrolling of Forward Least Fixpoint Algorithm

A. Biere, FMV, JKU Linz

0: continue? \[ S_C^0 \neq S_N^0 \quad \exists s_0 [I(s_0)] \]

0: terminate? \[ S_C^0 = S_N^0 \quad \forall s_0 [\neg I(s_0)] \]

0: bad state? \[ B \cap S_N^0 \neq \emptyset \quad \exists s_0 [I(s_0) \land B(s_0)] \]

1: continue? \[ S_C^1 \neq S_N^1 \quad \exists s_0, s_1 [I(s_0) \land T(s_0, s_1) \land \neg I(s_1)] \]

1: terminate? \[ S_C^1 = S_N^1 \quad \forall s_0, s_1 [I(s_0) \land T(s_0, s_1) \rightarrow I(s_1)] \]

1: bad state? \[ B \cap S_N^1 \neq \emptyset \quad \exists s_0, s_1 [I(s_0) \land T(s_0, s_1) \land B(s_1)] \]

2: continue? \[ S_C^2 \neq S_N^2 \quad \exists s_0, s_1, s_2 [I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land \neg (I(s_2) \lor \exists t_0 [I(t_0) \land T(t_0, s_2)])] \]

2: terminate? \[ S_C^2 = S_N^2 \quad \forall s_0, s_1, s_2 [I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \rightarrow I(s_2) \lor \exists t_0 [I(t_0) \land T(t_0, s_2)]] \]

2: bad state? \[ B \cap S_N^2 \neq \emptyset \quad \exists s_0, s_1, s_2 [I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land B(s_2)] \]
∀\(s_0, \ldots, s_{r+1}\) \[I(s_0) \land \bigwedge_{i=0}^{r} T(s_i, s_{i+1}) \rightarrow \]

∃\(t_0, \ldots, t_r\) \[I(t_0) \land s_{r+1} = t_r \land \bigwedge_{i=0}^{r-1} (t_i = t_{i+1} \lor T(t_i, t_{i+1}))\]

**Termination Check**

**radius** is smallest \(r\) for which formula is true
Radius Example

A. Biere, FMV, JKU Linz

initial states

states with distance 1 from initial states

single state with distance 2 from initial states

unreachable states
• checking $S_C = S_N$ in 2nd iteration results in QBF decision problem

$$\forall s_0, s_1, s_2 [I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \to I(s_2) \lor \exists t_0[I(t_0) \land T(t_0, s_2)]]$$

• not **eliminating quantifiers** results in QBF with one alternation
  
  – checking whether bad state is reached only needs SAT
  
  – number iterations bounded by radius $r = O(2^n)$

• so why not forget about termination and concentrate on bug finding?

  $\Rightarrow$ **Bounded Model Checking** (BMC)
0: continue? \( S_C^0 \neq S_N^0 \) \( \exists s_0[I(s_0)] \)

0: terminate? \( S_C^0 = S_N^0 \) \( \forall s_0[\neg I(s_0)] \)

0: bad state? \( B \cap S_N^0 \neq \emptyset \) \( \exists s_0[I(s_0) \land B(s_0)] \)

1: continue? \( S_C^1 \neq S_N^1 \) \( \exists s_0, s_1[I(s_0) \land T(s_0, s_1) \land \neg I(s_1)] \)

1: terminate? \( S_C^1 = S_N^1 \) \( \forall s_0, s_1[I(s_0) \land T(s_0, s_1) \rightarrow I(s_1)] \)

1: bad state? \( B \cap S_N^1 \neq \emptyset \) \( \exists s_0, s_1[I(s_0) \land T(s_0, s_1) \land B(s_1)] \)

2: continue? \( S_C^2 \neq S_N^2 \) \( \exists s_0, s_1, s_2[I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land \neg (I(s_2) \lor \exists t_0[I(t_0) \land T(t_0, s_2)])] \)

2: terminate? \( S_C^2 = S_N^2 \) \( \forall s_0, s_1, s_2[I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \rightarrow I(s_2) \lor \exists t_0[I(t_0) \land T(t_0, s_2)]] \)

2: bad state? \( B \cap S_N^1 \neq \emptyset \) \( \exists s_0, s_1, s_2[I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land B(s_2)] \)
Bounded Model Checking (BMC)

[BiereCimattiClarkeZhu TACAS’99]

- look only for counter example made of \( k \) states (the bound)

\[
I(s_0) \land \left( \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \right) \land \left( \bigvee_{i=0}^{k} \neg p(s_i) \right)
\]

- simple for safety properties \( Gp \) (e.g. \( p = \neg B \))

\[
I(s_0) \land \left( \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \right) \land \left( \bigvee_{i=0}^{k} \neg p(s_i) \right)
\]

- harder for liveness properties \( Fp \)

\[
I(s_0) \land \left( \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \right) \land \left( \bigvee_{l=0}^{k} T(s_l, s_{l+1}) \right) \land \left( \bigwedge_{i=0}^{k} \neg p(s_i) \right)
\]
• increase in efficiency of SAT solvers [ZChaff,MiniSAT,SATelite]

• SAT more robust than BDDs in bug finding
  (shallow bugs are easily reached by explicit model checking or testing)

• better unbounded but still SAT based model checking algorithms
  – $k$-induction [SinghSheeranStålmarck’00]
  – interpolation [McMillan’03]

• 3rd Intl. Workshop on Bounded Model Checking (BMC’05)

• other logics beside LTL and better encodings
  e.g. [LatvalaBiereHeljankoJuntilla’04]
Symbolic Transitive Closure

Transitive Closure

\[ T^* \equiv T^{2^n} \]

(assuming \( = \subseteq T \))

Standard Linear Unfolding

Iterative Squaring via Copying

\[ T^{i+1}(s,t) \equiv \exists m [ T^i(s,m) \land T(m,t) ] \]
\[ T^{2\cdot i}(s,t) \equiv \exists m [ T^i(s,m) \land T^i(m,t) ] \]

Non Copying Iterative Squaring

\[ T^{2\cdot i}(s,t) \equiv \exists m [ \forall c [ \exists l, r [(c \rightarrow (l,r) = (s,m)) \land (\overline{c} \rightarrow (l,r) = (m,t)) \land T^i(l,r)]] ] \]
dpll-sat(Assignment S)  [DavisLogemannLoveland62]
boolean-constraint-propagation()
if contains-empty-clause() then return false
if no-clause-left() then return true
v := next-unassigned-variable()
return dpll-sat(S ∪ {v ↦ false}) ∨ dpll-sat(S ∪ {v ↦ true})

dpll-qbf(Assignment S)  [CadoliGiovanardiSchaerf98]
boolean-constraint-propagation()
if contains-empty-clause() then return false
if no-clause-left() then return true
v := next-outermost-unassigned-variable()

@ := is-existential(v) ? ∨ : ∧
return dpll-sat(S ∪ {v ↦ false}) @ dpll-sat(S ∪ {v ↦ true})
Why is QBF harder than SAT?

\[ \models \forall x . \exists y . (x \leftrightarrow y) \]

\[ \not\models \exists y . \forall x . (x \leftrightarrow y) \]

Decision order matters!
State-of-the-Art in QBF Solvers

- most implementations DPLL alike:  
  - learning was added  
  - top-down:  split on variables from the outside to the inside

- multiple quantifier elimination procedures:
  - enumeration  
  - expansion  
  - bottom-up:  eliminate variables from the inside to the outside

- q-resolution  
  - with expansion

- symbolic representations  
  - BDDs
Summary

- applications fuel interest in SAT
  - incredible capacity increase (this year: MiniSAT, SATelite)
  - SAT solver competition affiliated to SAT conference
  - SAT is becoming a core verification technology

- QBF is catching up
  - solvers are getting better
  - new applications
  - richer structure