Reachability Analysis with QBF

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

Workshop Designing Correct Circuits
DCC’06

Vienna, Austria, March 25, 2006
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
  \[\Rightarrow\]  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: Bad State Reached

DCC’06, Vienna – A. Biere, FMV, JKU Linz
Forward Fixpoint Algorithm: Termination, No Bad State Reachable

DCC’06, Vienna – A. Biere, FMV, JKU Linz
initial states \( I \), transition relation \( T \), bad states \( B \)

\[
\text{model-check}^{\text{forward}}(I, T, B) = \sqrt{0}; S_N = I;
\]
\[
\text{while } S_C \neq S_N \text{ do}
\]
\[
\text{if } B \cap S_N \neq 0 \text{ then}
\]
\[
\text{return } \text{“found error trace to bad states”};
\]
\[
S_C = S_N;
\]
\[
S_N = S_C \cup \text{Img}(S_C);
\]
\[
\text{done};
\]
\[
\text{return } \text{“no bad state reachable”};
\]

symbolic model checking represents set of states in this BFS symbolically
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? \[ B \cap S_N^1 \neq \emptyset \quad \exists s_0, s_1, s_2[I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land B(s_2)] \]
Termination Check = Determine Radius

∀s_0, ..., s_{r+1} [ I(s_0) \land T(s_0, s_1) \land \cdots \land T(s_r, s_{r+1}) \rightarrow

\exists t_0, ..., t_r, t_{r-1} [ I(t_0) \land T(t_0, t_1) \land \cdots \land T(t_{r-1}, t_r) \land

(t_0 = s_{r+1} \lor t_1 = s_{r+1} \lor \cdots \lor t_r = s_{r+1})]]

radius is smallest \( r \) for which formula is true
initial states

unreachable states

states with distance 1 from initial states

single state with distance 2 from initial states
Quantified Boolean Formulae (QBF)

- propositional logic \((SAT \subseteq QBF)\)
  - constants \(0, 1\)
  - operators \(\land, \neg, \rightarrow, \leftrightarrow, \ldots\)
  - 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 \(\equiv\))
  - invalid \(\exists x[\forall y[x \leftrightarrow y]]\)
QBF Semantics

- 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 \( \text{PSPACE} = \text{NP} \)
• checking $S_C = S_N$ in 2nd iteration results in QBF decision problem

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

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

• successfully used in Software Model Checking

  [CookKröningSharygina SPIN’05]

• termination check *often* costly $\Rightarrow$ **Bounded Model Checking (BMC)**
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? \[ B \cap S_N^1 \neq \emptyset \quad \exists s_0, s_1, s_2[I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land B(s_2)] \]
Bounded Model Checking (BMC)

[BiereCimattiClarkeZhu TACAS’99]

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

- simple for safety properties $\mathcal{G}p$ (e.g. $p = \neg B$)

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

- harder for liveness properties $\mathcal{F}p$

$$I(s_0) \land (\bigwedge_{i=0}^{k-1} T(s_i,s_{i+1})) \land (\bigvee_{l=0}^{k} T(s_l,s_{l+1})) \land \bigwedge_{i=0}^{k} \neg p(s_i)$$
• 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 CAV’03]

• 4th Intl. Workshop on Bounded Model Checking (BMC’06)

• other logics beside LTL and better encodings
  e.g. [LatvalaBiereHeljankoJuntilla FMCAD’04]
more specifically \( k \)-induction

- does there exist \( k \) such that the following formula is \textit{unsatisfiable}

\[
\overline{B(s_0)} \land \cdots \land \overline{B(s_{k-1})} \land T(s_0, s_1) \land \cdots \land T(s_{k-1}, s_k) \land B(s_k) \land \bigwedge_{0 \leq i < j \leq k} s_i \neq s_j
\]

- if \textit{unsatisfiable} and \( \neg \text{BMC}(k) \) then \textit{bad state unreachable}

backwards version of \textit{reoccurrence radius}

\( k = 0 \) check whether \( \neg B \) tautological (propositionally)

\( k = 1 \) check whether \( \neg B \) inductive for \( T \)
• **radius**  longest shortest from an initial state to a reachable state

• **reoccurrence radius**  longest *simple* path
  
  – *simple* = without reoccurring state

• reoccurrence radius can be exponentially larger than diameter
  
  – $n$ bit register with load signal, initialized with zero
  
  – reoccurrence radius $2^n - 1$

  – diameter 1

• applies to backward reoccurrence radius and thus $k$-induction as well
Reoccurrence radius: $O(n)$

Radius: $O(1)$
Transitive Closure

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

(assuming \( = \subseteq T \))

Standard Linear Unfolding

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

Iterative Squaring via Copying

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

Non-Copying Iterative Squaring

\[ T^{2 \cdot i}(s, t) \equiv \exists m [ \forall c [ \exists l, r [(c \rightarrow (l, r) = (s, m)) \land (c \rightarrow (l, r) = (m, t)) \land T^i(l, r)] ] \]
• flat circuit model exponential in size of hierarchical model

  – $M_0$ has one register
  – $M_{i+1}$ instantiates $M_i$ twice
  – $M_n$ has $2^n$ registers

• model hierarchy/repetitions in QBF as in non-copying iterative squaring

  – $T(s, t)$ interpreted as combinatorial circuit with inputs $s$, outputs $t$

• conjecture: [Savitch70] even applies to hierarchical descriptions
• for counter example to check satisfiability of
\[ \exists s_0, s_1[I(s_0) \land T(s_0, s_1) \land p(s_1) \land \forall s_2[T(s_1, s_2) \rightarrow \neg q(s_2)]] \]

• for counter example to check satisfiability of
\[ \exists s_0, s_1[I(s_0) \land T(s_0, s_1) \land p(s_1) \land \forall s_2[T(s_1, s_2) \rightarrow \neg q(s_1) \land \neg q(s_2)]] \]
(assume \(\neg q\)-predicated diameter \(\leq 2\))

• similarly sequential equivalence checking
\[ \text{EFAG}(o_1 = o_2) \]
• transition logic of industrial circuits can be very large

• use QBF to *share* transition relation $T$ among time frames

\[
\exists s_0, s_1, s_2, s_3 [ \\
\forall i = 0, 1, 2 [ \\
\exists l, r [ (i = 0 \rightarrow (l = s_0 \land r = s_1)) \land \\
(i = 1 \rightarrow (l = s_1 \land r = s_2)) \land \\
(i = 2 \rightarrow (l = s_2 \land r = s_3)) \land \\
T(l, r) \land \\
(B(s_0) \lor B(s_1) \lor B(s_2) \lor B(s_3))]]]
\]

• constant formula size reduction (only)

• experiments show space vs. time trade off
rectification problem

- parameters $p$
- inputs $i$
- generic circuit $g$
- specification $s$

QBF solver can find parameters $p$

black box equivalence checking [SchollBecker DAC’01]

FPGA synthesis [LingSinghBrown SAT’05]
original SAT formulation of simple path constraints quadratic in bound $k$

$$| \bigwedge_{0 \leq i < j \leq k} s_i \neq s_j | = O(k^2)$$

can be reduced to $O(k \cdot \log k)$  

[KröningShtrichman VMCAI’03]

with QBF becomes linear $O(k)$:

$$\bigwedge_{0 \leq i < j \leq k} s_i \neq s_j \equiv \forall j = 0, \ldots, k \left[ \exists s \left[ \bigwedge_{0 \leq i \leq k} (j = i \leftrightarrow s = s_i) \right] \right]$$
Reachability with QBF: Experiments

still work in progress

- bounded model checker for flat circuits with $k$ induction
  \texttt{smv2qbf}

- can also produce forward/backward diameter checking problems in QBF

- so far instances have been quite challenging for current QBF solvers

- found some toy examples which can be checked much faster with QBF
  - for instance the $n$ bit register with load signal discussed before

- non-copying iterative squaring does not give any benefits (yet)
**DPLL for SAT and QBF**

\[\text{dpll-sat}(\text{Assignment } S)\]  
[DavisLogemannLoveland62]

\[
\text{boolean-constraint-propagation()}
\]

\[
\text{if } \text{contains-empty-clause()} \text{ then return } false
\]

\[
\text{if } \text{no-clause-left()} \text{ then return } true
\]

\[
\nu := \text{next-unassigned-variable()}
\]

\[
\text{return } \text{dpll-sat}(S \cup \{\nu \mapsto false\}) \lor \text{dpll-sat}(S \cup \{\nu \mapsto true\})
\]

\[\text{dpll-qbf}(\text{Assignment } S)\]  
[CadoliGiovanardiSchaerf98]

\[
\text{boolean-constraint-propagation()}
\]

\[
\text{if } \text{contains-empty-clause()} \text{ then return } false
\]

\[
\text{if } \text{no-clause-left()} \text{ then return } true
\]

\[
\nu := \text{next-outermost-unassigned-variable()}
\]

\[
\odot := \text{is-existential}(\nu) ? \lor : \land
\]

\[
\text{return } \text{dpll-sat}(S \cup \{\nu \mapsto false\}) \land \text{dpll-sat}(S \cup \{\nu \mapsto true\})
\]
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: [Cadoli...98][Rintanen01]
  - learning was added [Giunchiglia...01] [Letz01] [ZhangMalik02]
  - top-down: split on variables from the outside to the inside

- multiple quantifier elimination procedures:
  - enumeration [PlaistedBiereZhu03] [McMillan02]
  - expansion [Aziz-Abdulla...00] [WilliamsBiere...00] [AyariBasin02]
  - bottom-up: eliminate variables from the inside to the outside

- q-resolution [KleineBüning...95], with expansion [Biere04]

- symbolic representations [PanVardi04] [Benedetti05] BDDs
Summary

• applications fuel interest in SAT
  – incredible capacity increase (last year: MiniSAT, SATelite)
  – SAT Solver Competition resp. SAT Race affiliated to SAT conference
  – SAT is becoming a core verification technology

• QBF is catching up and is exponentially more succinct
  – solvers are getting better (first competitive QBF Evaluation 2006)
  – new applications:
    CTL, Termination, Trans. Closure, Hierarchy/Sharing, Simple Paths
  – richer structure than SAT ⇒ many opportunities for optimizations