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

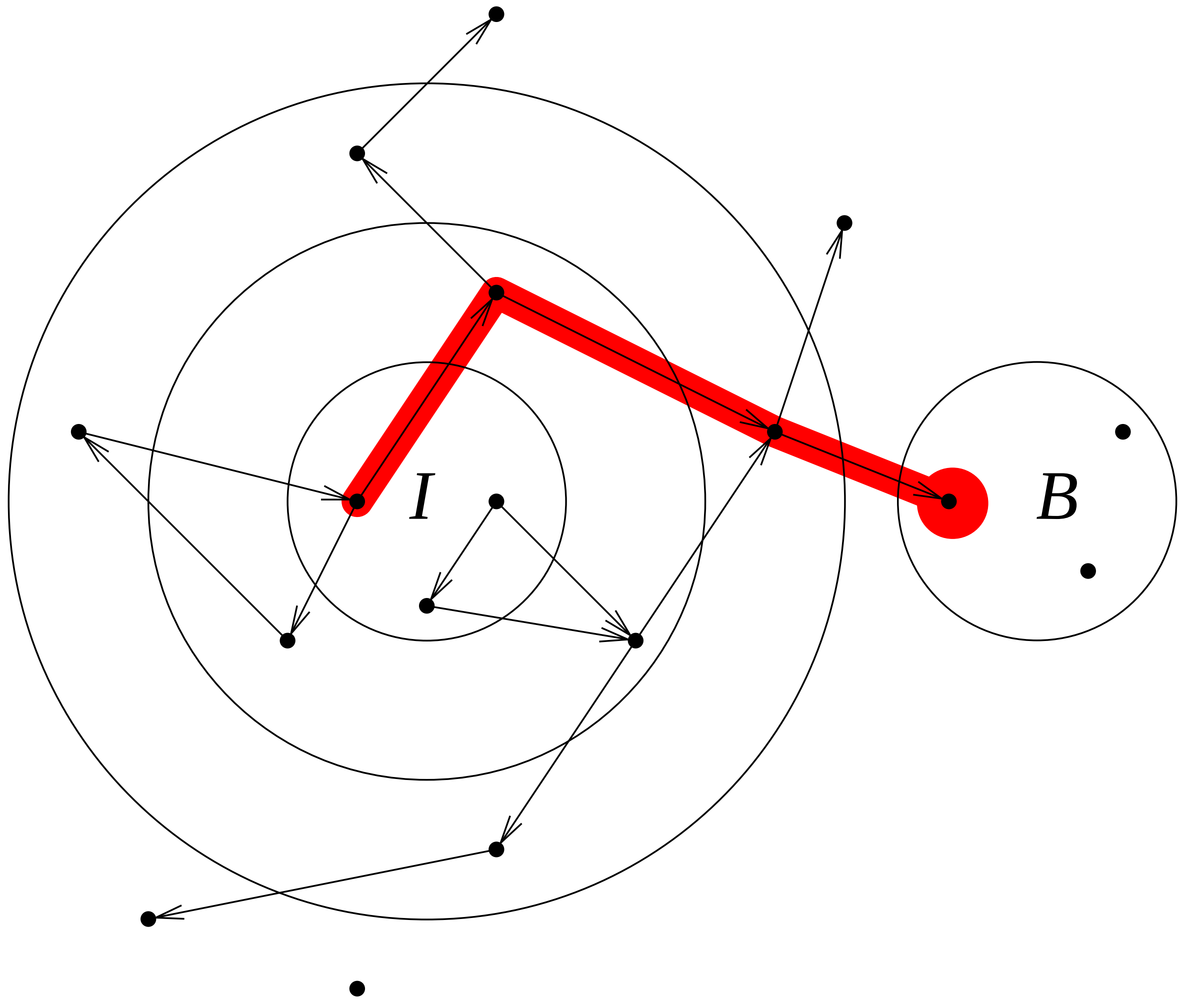
Model Checking

- 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

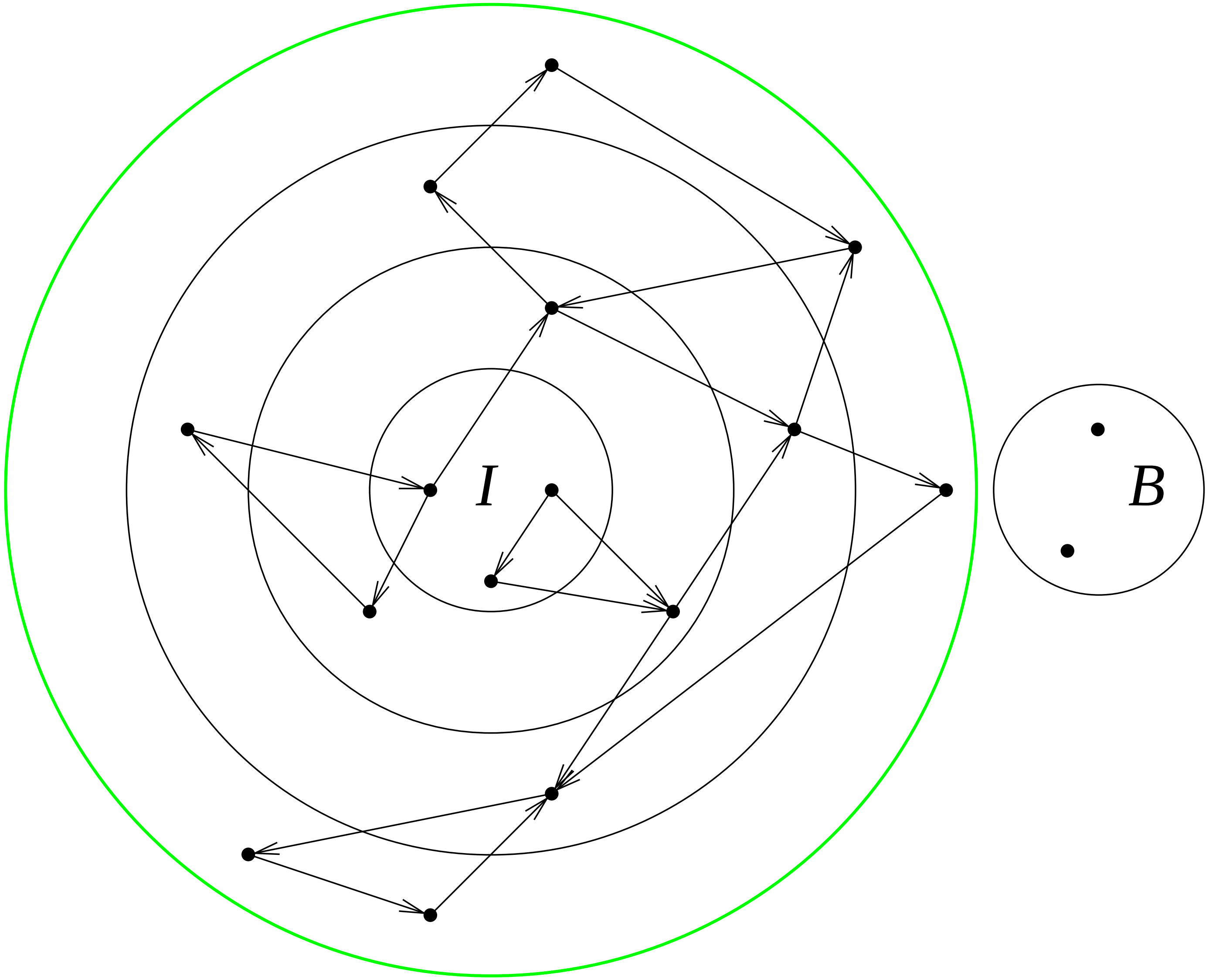
  $\Rightarrow$ State Explosion Problem (state space exponential in program size)

- 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



Forward Fixpoint Algorithm: Termination, No Bad State Reachable



Forward Least Fixpoint Algorithm for Model Checking Safety

initial states I, transition relation T, bad states B

<u>model-check</u>$^{\mu}_{\text{forward}}$ (I, T, B)

- $S_C = \emptyset$; $S_N = I$;
- **while** $S_C \neq S_N$ **do**
  - **if** $B \cap S_N \neq \emptyset$ **then**
    - **return** "found error trace to bad states";
  - $S_C = S_N$;
  - $S_N = S_C \cup Img(S_C)$;
- **done**;
- **return** "no bad state reachable";

symbolic model checking represents set of states in this BFS symbolically

Unrolling of Forward Least Fixpoint Algorithm

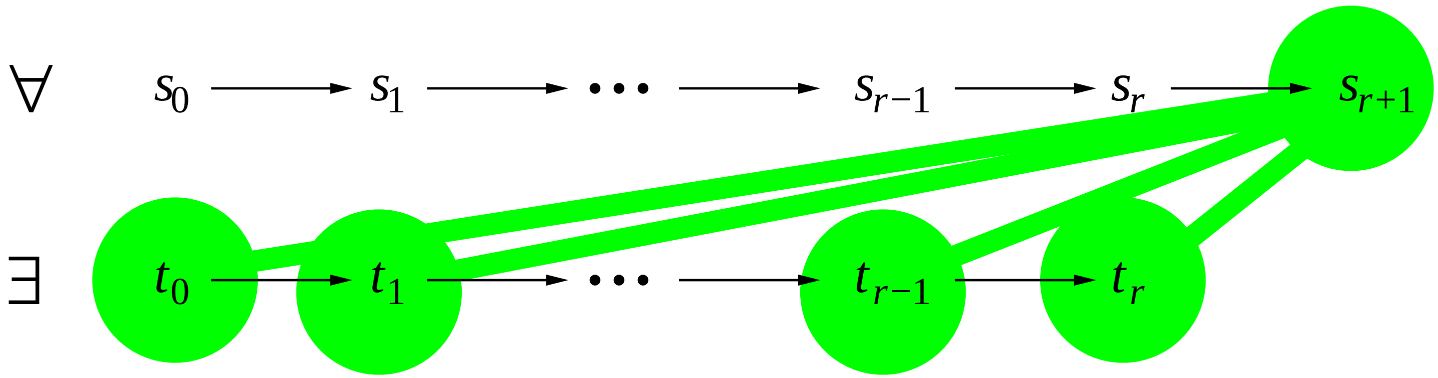
0: continue?	$S_C^0 \neq S_N^0$	$\exists s_0[I(s_0)]$
0: terminate?	$S_C^0 = S_N^0$	$\forall s_0[\neg I(s_0)]$
0: bad state?	$B \cap S_N^0 \neq \emptyset$	$\exists s_0[I(s_0) \wedge B(s_0)]$
1: continue?	$S_C^1 \neq S_N^1$	$\exists s_0, s_1[I(s_0) \wedge T(s_0, s_1) \wedge \neg I(s_1)]$
1: terminate?	$S_C^1 = S_N^1$	$\forall s_0, s_1[I(s_0) \wedge T(s_0, s_1) \rightarrow I(s_1)]$
1: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1[I(s_0) \wedge T(s_0, s_1) \wedge B(s_1)]$
2: continue?	$S_C^2 \neq S_N^2$	$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \neg(I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)])]$
2: terminate?	$S_C^2 = S_N^2$	$\forall s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]$
2: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

Termination Check = Determine Radius

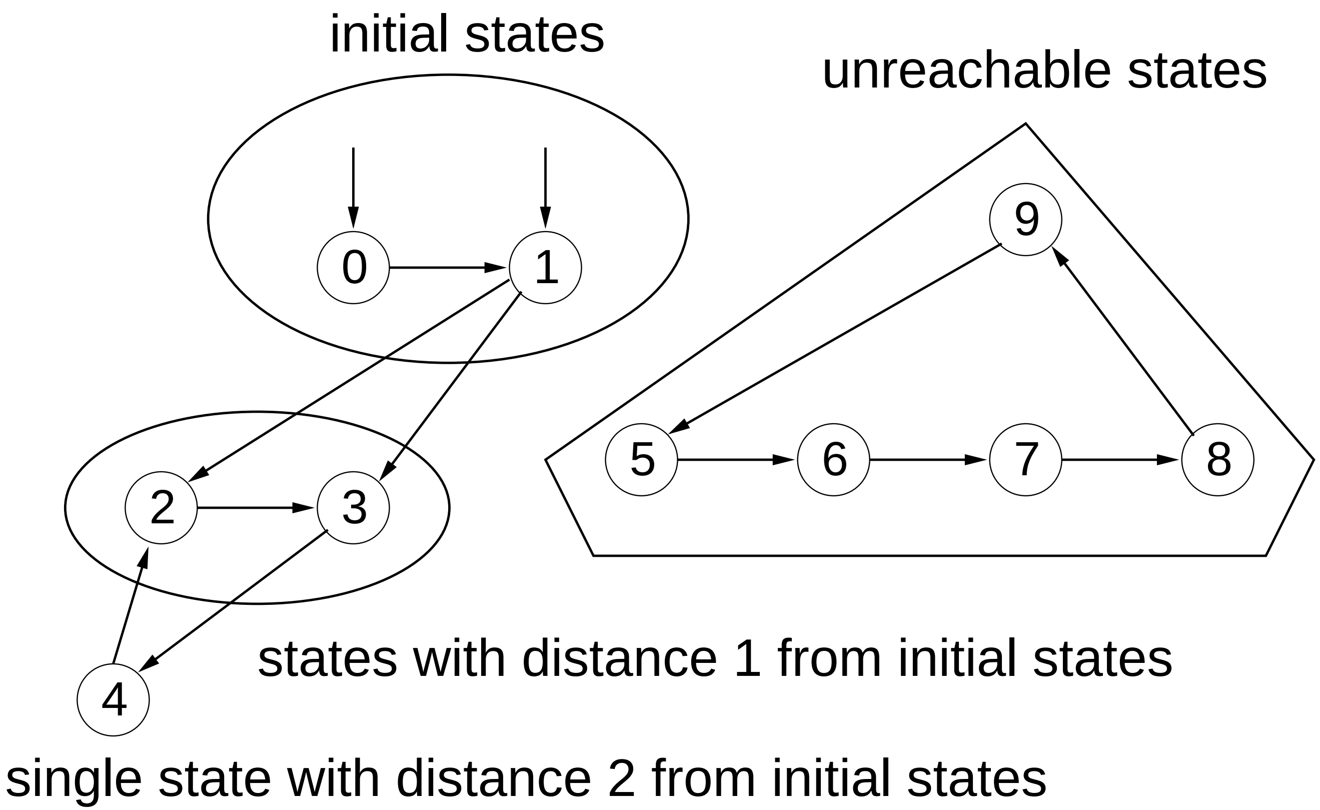
$$\forall s_0,\ldots,s_{r+1}\big[\,I(s_0)\wedge T(s_0,s_1)\wedge\cdots\wedge T(s_r,s_{r+1})\ \rightarrow$$

$$\exists t_0,\ldots,t_r,t_{r-1}[\,I(t_0)\wedge T(t_0,t_1)\wedge\cdots\wedge T(t_{r-1},t_r)\ \wedge$$

$$(t_0 = s_{r+1}\ \vee\ t_1 = s_{r+1}\ \vee\ \cdots\ \vee\ t_r = s_{r+1})]\big]$$



radius is smallest r for which formula is true


initial states
unreachable states
0
1
9
5
6
7
8
2
3
4
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 $\wedge, \neg, \rightarrow, \leftrightarrow, \ldots$
  - 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 $\exists x[\forall y[x \leftrightarrow y]]$

QBF Semantics

- semantics given as expansion of quantifiers

$$\exists x[f] \equiv f[0/x] \vee f[1/x] \qquad \forall x[f] \equiv f[0/x] \wedge f[1/x]$$

- 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 PSPACE = NP

QBF Application I: Termination Check

- checking $S_C = S_N$ in 2nd iteration results in QBF decision problem

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

- 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)

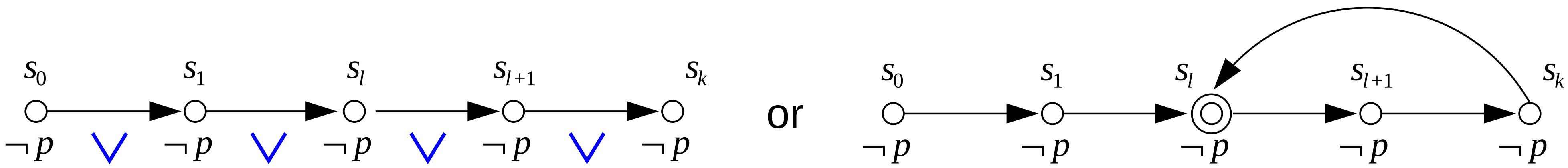
BMC Part of Fixpoint Algorithm

~~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?	$B \cap S_N^0 \neq \emptyset$	$\exists s_0[I(s_0) \wedge B(s_0)]$
~~1: continue?~~	~~$S_C^1 \neq S_N^1$~~	~~$\exists s_0, s_1[I(s_0) \wedge T(s_0, s_1) \wedge \neg I(s_1)]$~~
~~1: terminate?~~	~~$S_C^1 = S_N^1$~~	~~$\forall s_0, s_1[I(s_0) \wedge T(s_0, s_1) \rightarrow I(s_1)]$~~
1: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1[I(s_0) \wedge T(s_0, s_1) \wedge B(s_1)]$
~~2: continue?~~	~~$S_C^2 \neq S_N^2$~~	~~$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge$~~ $\neg(I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)])]$
~~2: terminate?~~	~~$S_C^2 = S_N^2$~~	~~$\forall s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow$ $I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]$~~
2: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

Bounded Model Checking (BMC)

[BiereCimattiClarkeZhu TACAS'99]

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



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

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

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

$$I(s_0) \wedge \left(\bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})\right) \wedge \left(\bigvee_{l=0}^{k} T(s_k, s_l)\right) \wedge \bigwedge_{i=0}^{k} \neg p(s_i)$$

Bounded Model Checking State-of-the-Art

- 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]

Induction with SAT

[SinghSheeranStålmarck FMCAD'00]

- more specifically k-induction
  - does there exist k such that the following formula is *unsatisfiable*

$$\overline{B(s_0)} \wedge \cdots \wedge \overline{B(s_{k-1})} \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge B(s_k) \wedge \bigwedge_{0 \le i < j \le k} s_i \neq s_j$$

  - if *unsatisfiable* and $\neg$BMC(k) then bad state unreachable

- backward version of reoccurrence radius

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

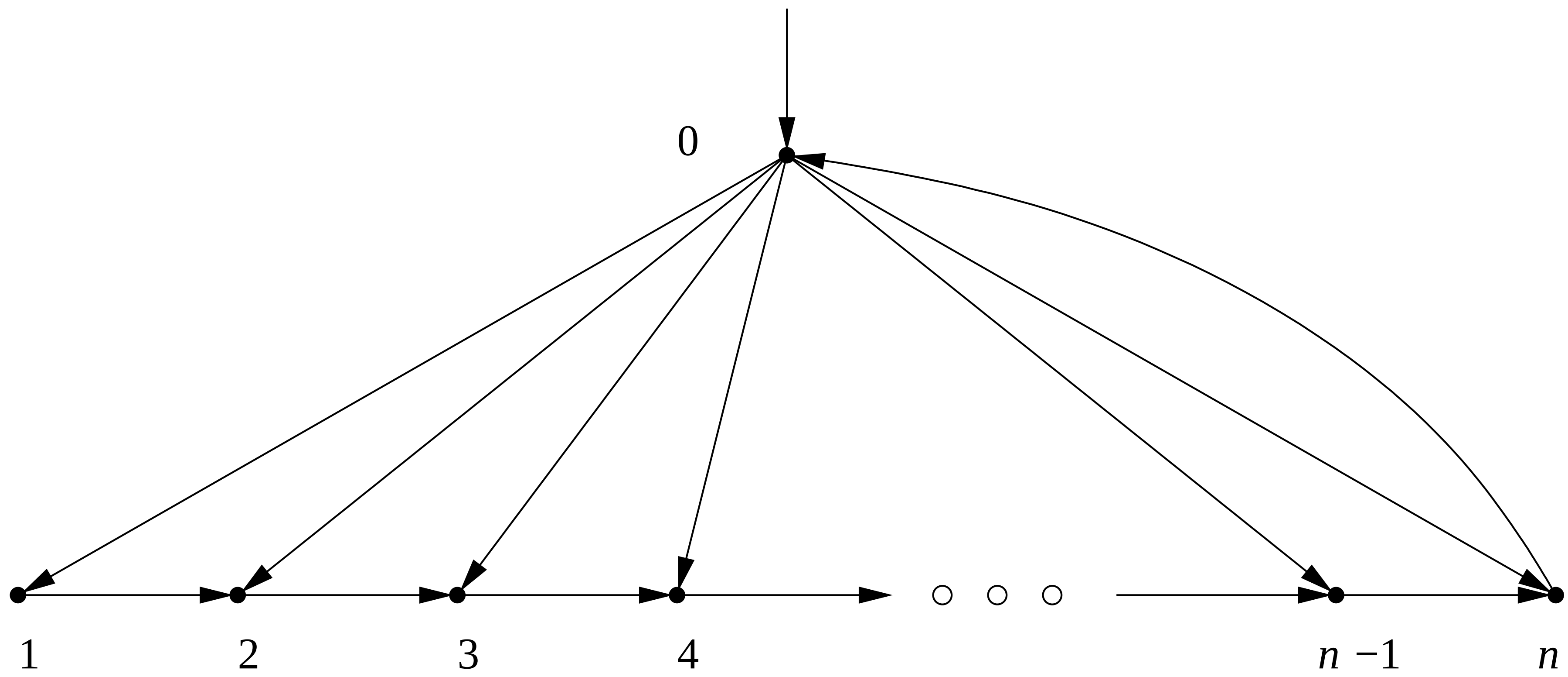
- $k = 1$ check whether $\neg B$ inductive for T

- **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 $O(1)$

QBF Application II: Transitive Closure

Transitive Closure

$$T^* \quad \equiv \quad T^{2^n}$$

(assuming $\quad = \subseteq T$)

Standard Linear Unfolding

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

Iterative Squaring via Copying

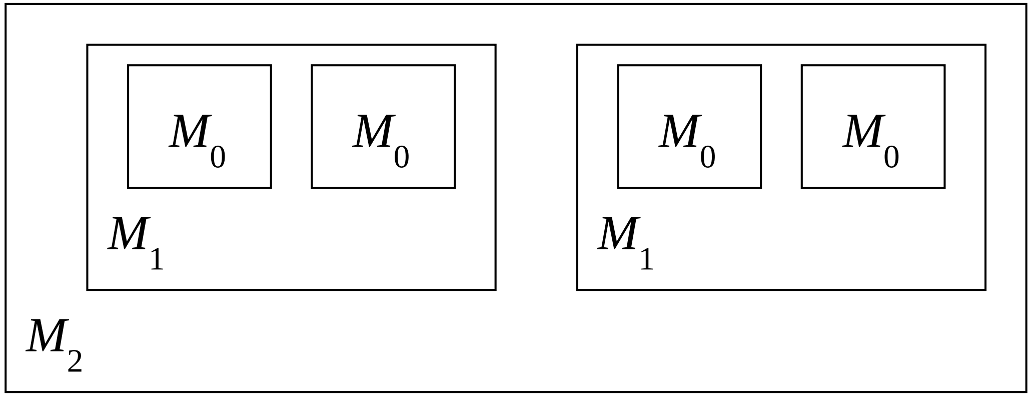
$$T^{2\cdot i}(s,t) \equiv \exists\, m[\, T^i(s,m) \wedge T^i(m,t)]$$

Non-Copying Iterative Squaring

$$T^{2\cdot i}(s,t) \quad \equiv \quad \exists\, m[\forall c[\exists l,r[(c \rightarrow (l,r) = (s,m)) \wedge (\bar{c} \rightarrow (l,r) = (m,t)) \wedge T^i(l,r)]]]$$

QBF Application III: Hierarchy

- 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 $\mathbf{AG}(p \rightarrow \mathbf{EX}q)$ (deadlock free)

$$\exists s0, s1[I(s_0) \wedge T(s_0, s_1) \wedge p(s_1) \wedge \forall s_2[T(s_1, s_2) \rightarrow \neg q(s_2)]]$$

- for counter example to check satisfiability of $\mathbf{AG}(p \rightarrow \mathbf{EF}q)$ (livelock free)

$$\exists s0, s1[I(s_0) \wedge T(s_0, s_1) \wedge p(s_1) \wedge \forall s_2[T(s_1, s_2) \rightarrow \neg q(s_1) \wedge \neg q(s_2)]$$

(assume $(\neg q)$-predicated diameter ≤ 2)

- similarly sequential equivalence checking $\mathbf{EFAG}(o_1 = o_2)$

QBF Application V: Sharing of Transition Relation

[DershowitzHannaKatz SAT'05]

- transition logic of industrial circuits can be very large
- use QBF to *share* transition relation T among time frames

$$
\begin{aligned}
&\exists s_0, s_1, s_2, s_3 [\\
&\quad \forall i = 0,1,2 [\\
&\qquad \exists l, r \; [\; (i = 0 \rightarrow (l = s_0 \wedge r = s_1) \wedge\\
&\qquad\qquad (i = 1 \rightarrow (l = s_1 \wedge r = s_2) \wedge\\
&\qquad\qquad (i = 2 \rightarrow (l = s_2 \wedge r = s_3) \wedge\\
&\qquad\qquad T(l, r) \wedge\\
&\qquad\qquad (B(s_0) \vee B(s_1) \vee B(s_2) \vee B(s_3))]]]
\end{aligned}
$$

- constant formula size reduction (only)
- experiments show space vs. time trade off

- rectification problem
  - parameters p
  - inputs i
  - generic circuit g
  - specification s

$$\exists p[\forall i[g(i,p) = s(i)]]$$

- 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 $O(k)$:

$$\bigwedge_{0 \leq i < j \leq k} s_i \neq s_j \quad \equiv \quad \forall\, j = 0, \ldots, k \left[\exists s [\bigwedge_{0 \leq i \leq k} \big(j = i \;\leftrightarrow\; s = s_i \big)] \right]$$

Reachability with QBF: Experiments

still work in progress

- bounded model checker for flat circuits with k induction 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-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 $\cup \{v \mapsto$ *false*$\}$) $\vee$ dpll-sat(S $\cup \{v \mapsto$ *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) ? $\vee$: $\wedge$

return dpll-sat(S $\cup \{v \mapsto$ *false*$\}$) @ dpll-sat(S $\cup \{v \mapsto$ *true*$\}$)

Why is QBF harder than SAT?

$$\models \quad \forall x \,.\, \exists y \,.\, (x \leftrightarrow y)$$

$$\not\models \quad \exists y \,.\, \forall x \,.\, (x \leftrightarrow y)$$

Decision order matters!

- 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 $\Rightarrow$ many opportunities for optimizations