QBF in Formal Verification

Armin Biere

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

Alpine Verification Meeting 2005

Laussanne, Switzerland
6. October 2006

Quantified Boolean Formulae (QBF)

A. Biere, FMV, JKU Linz

- 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 $\exists x[\forall y[x \leftrightarrow y]]$

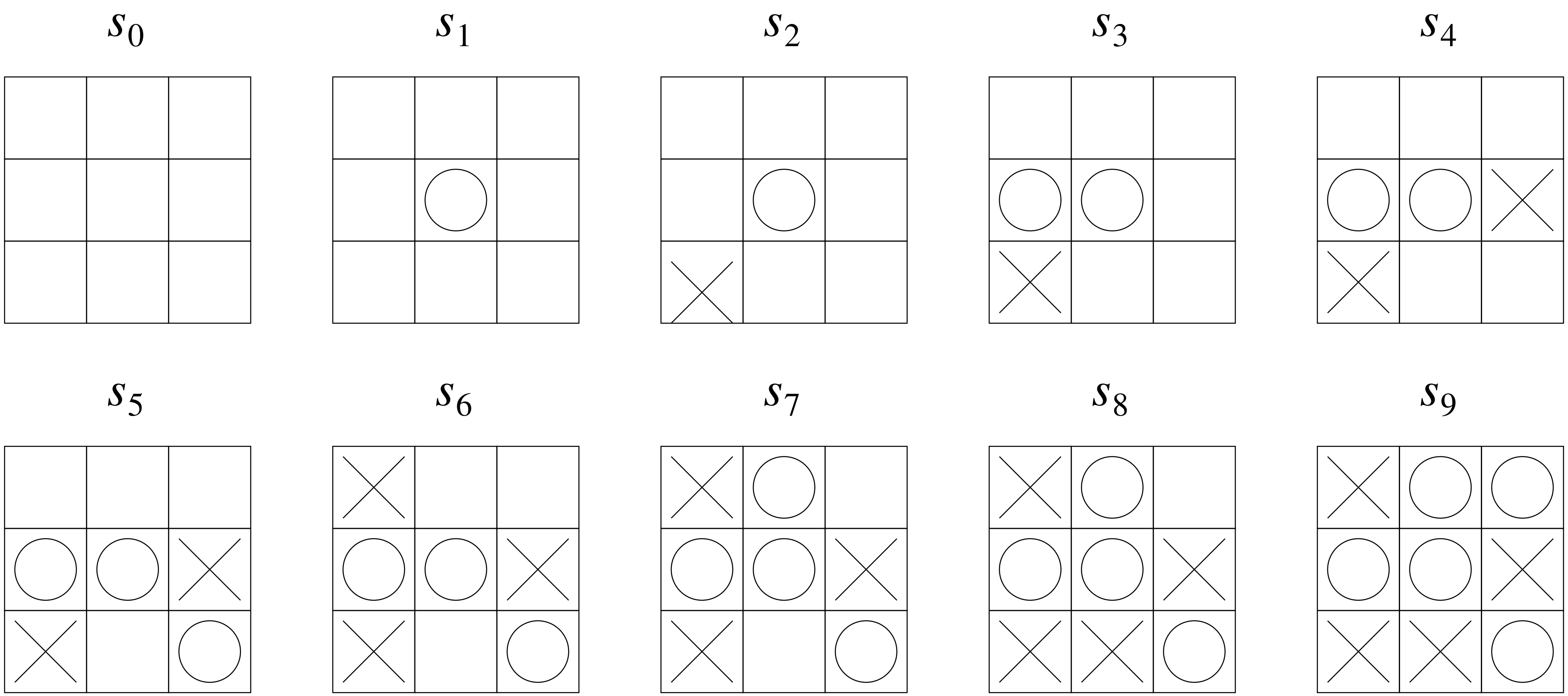
- semantics given as expansion of quantifiers

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

- 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

Tic-Tac-Toe

A. Biere, FMV, JKU Linz



No Winning Strategy for Tic-Tac-Toe

A. Biere, FMV, JKU Linz

$$
\begin{array}{l}
\not\models \quad \forall s_0[\mathit{empty}(s_0) \rightarrow \\
\quad \exists x_1[\mathit{circle}(s_0,x_1,s_1) \wedge \\
\quad\quad \forall y_2[\mathit{cross}(s_1,y_2,s_2) \rightarrow \\
\quad\quad\quad \exists x_3[\mathit{circle}(s_2,x_3,s_3) \wedge \\
\quad\quad\quad\quad \forall y_4[\mathit{cross}(s_3,y_4,s_4) \rightarrow \\
\quad\quad\quad\quad\quad \exists x_5[\mathit{circle}(s_4,x_5,s_5) \wedge \\
\quad\quad\quad\quad\quad\quad \forall y_6[\mathit{cross}(s_5,y_6,s_6) \rightarrow \\
\quad\quad\quad\quad\quad\quad\quad \exists x_7[\mathit{circle}(s_6,x_7,s_7) \wedge \\
\quad\quad\quad\quad\quad\quad\quad\quad \forall y_8[\mathit{cross}(s_7,y_8,s_8) \rightarrow \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \exists x_9[\mathit{circle}(s_8,x_9,s_9) \wedge \mathit{win}_{\mathit{circle}}(s_9)]]]]]]]]]
\end{array}
$$

x_i, y_i plays (4 bits each)

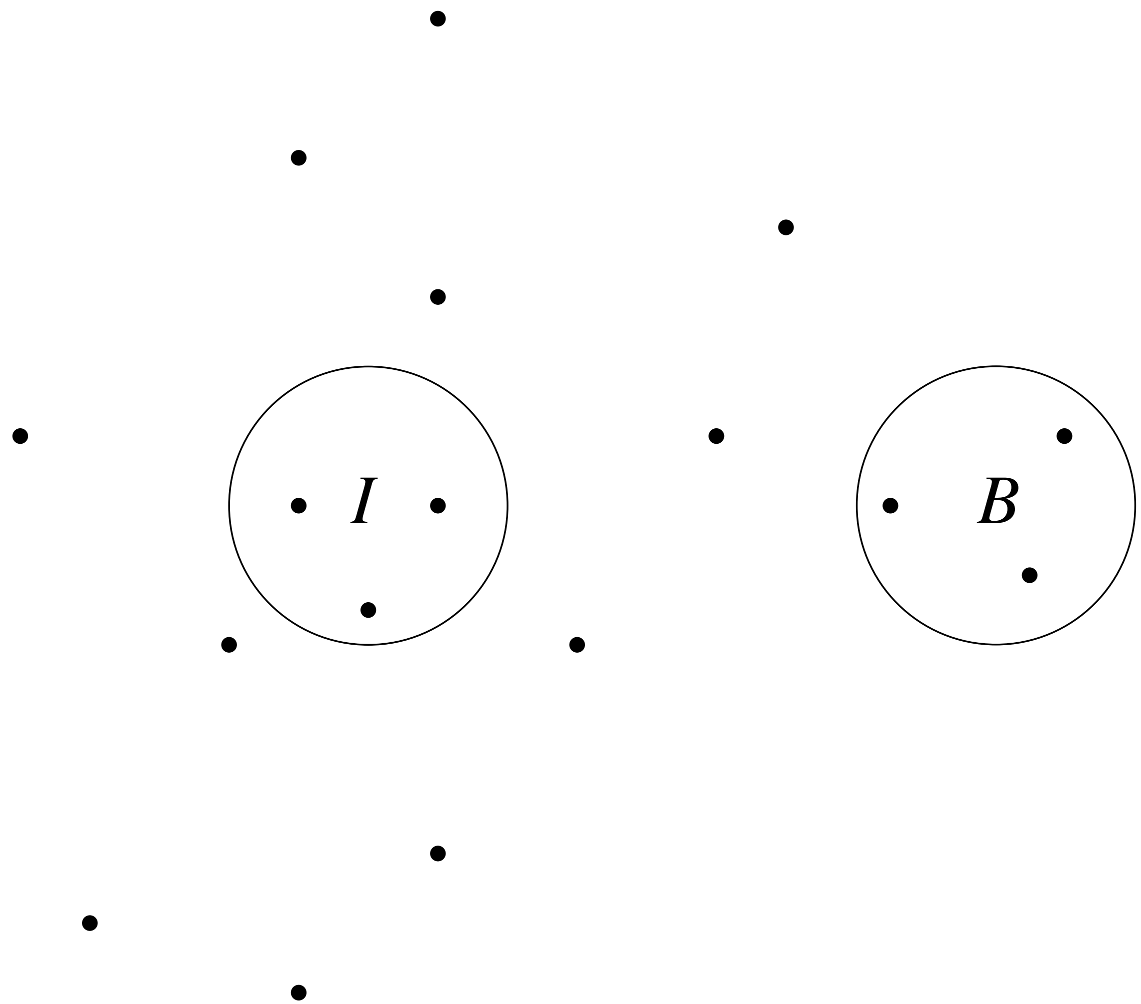
s_i configurations

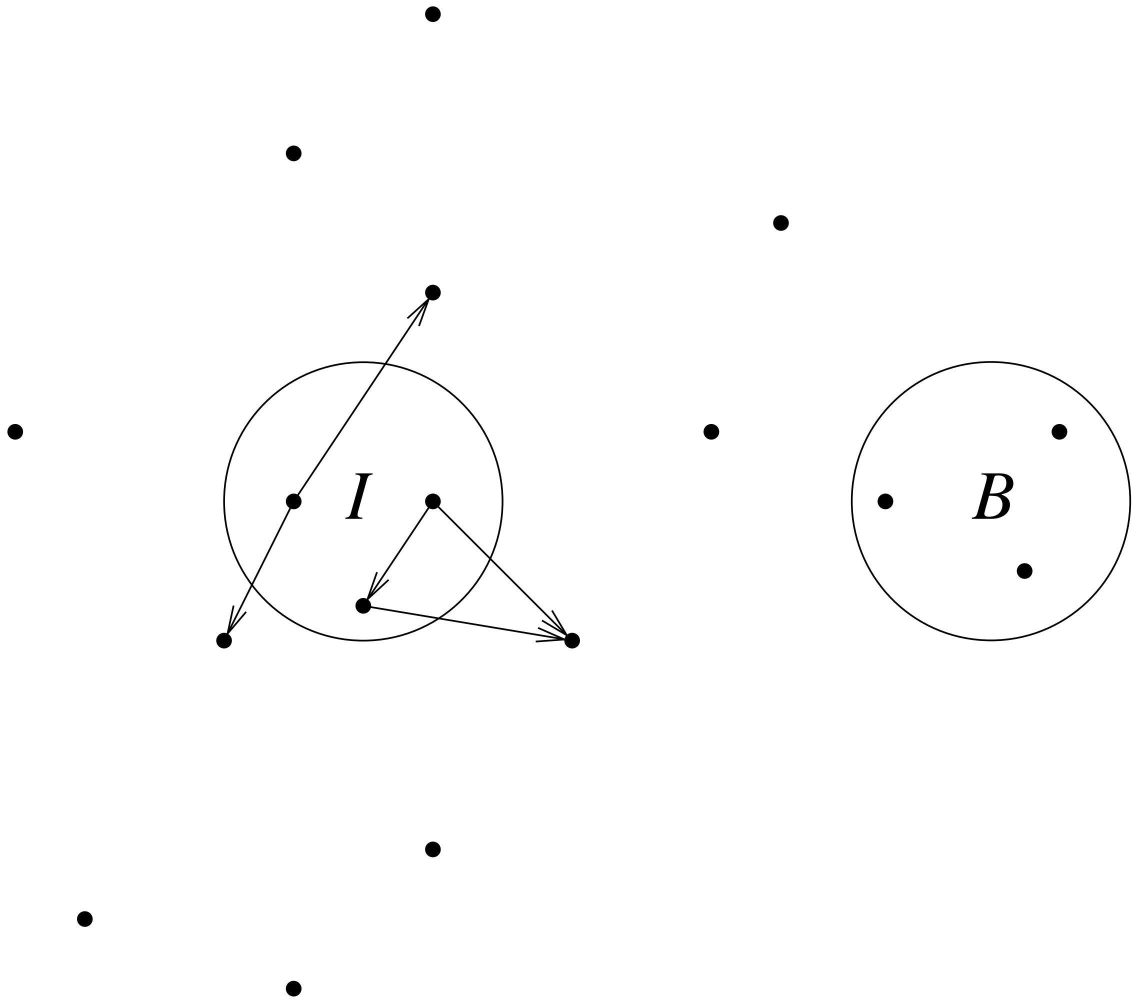
(9×3 bits each)

- 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)

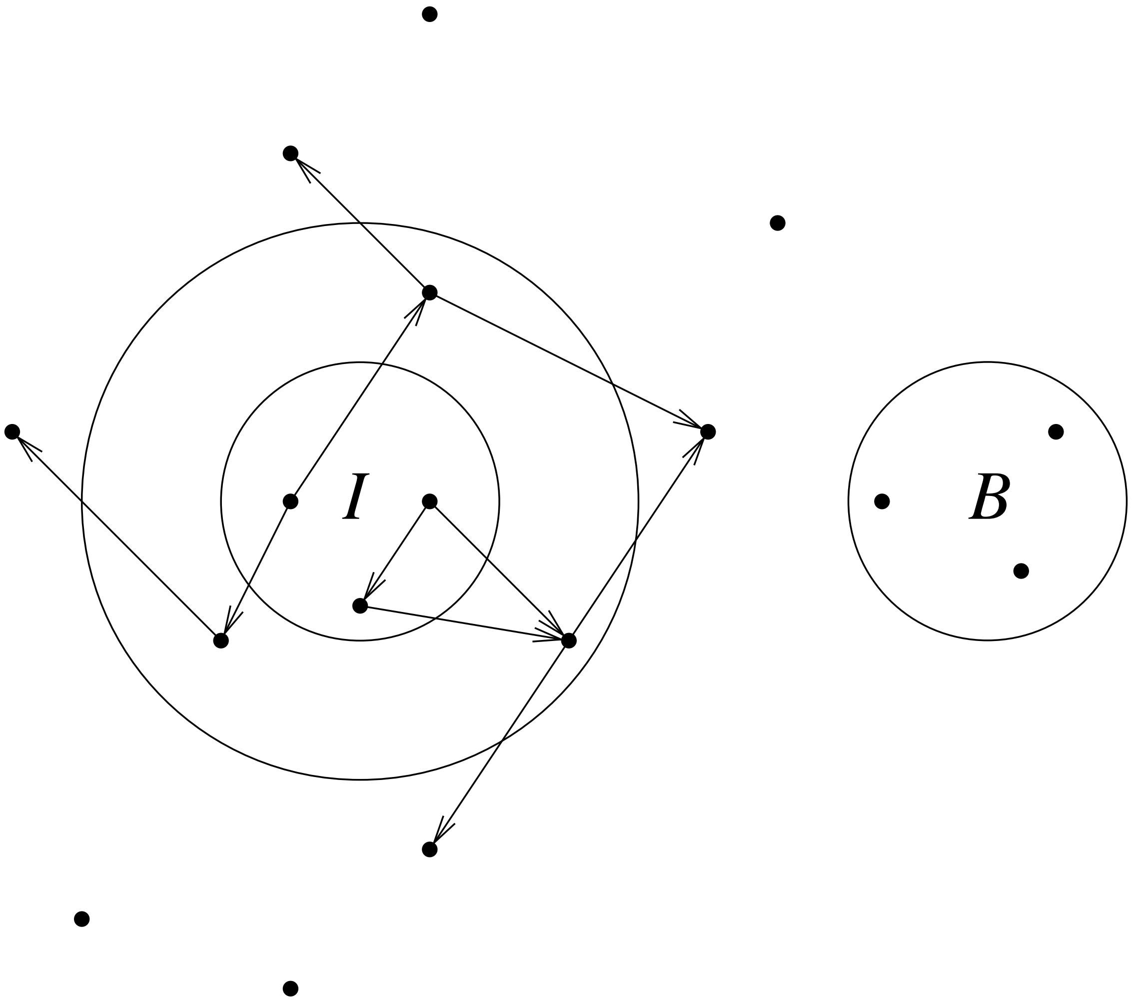
- symbolic model checking [McMillan Thesis'93], [CoudertMadre'89]
  - use symbolic representations for sets of states
  - originally with Binary Decision Diagrams [Bryant'86]
  - Bounded Model Checking using SAT [BiereCimattiClarkeZhu'99]


I
B

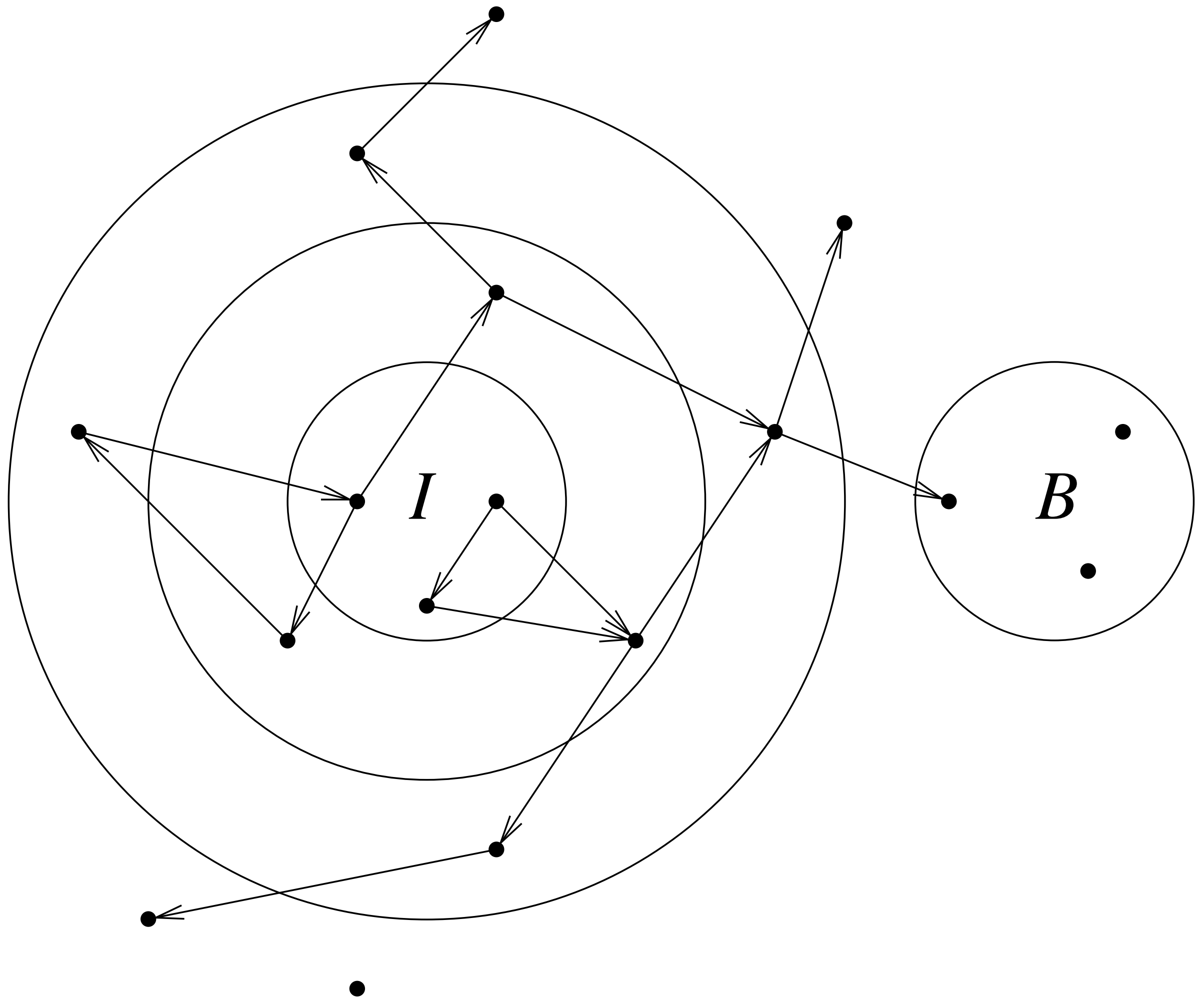

I
B

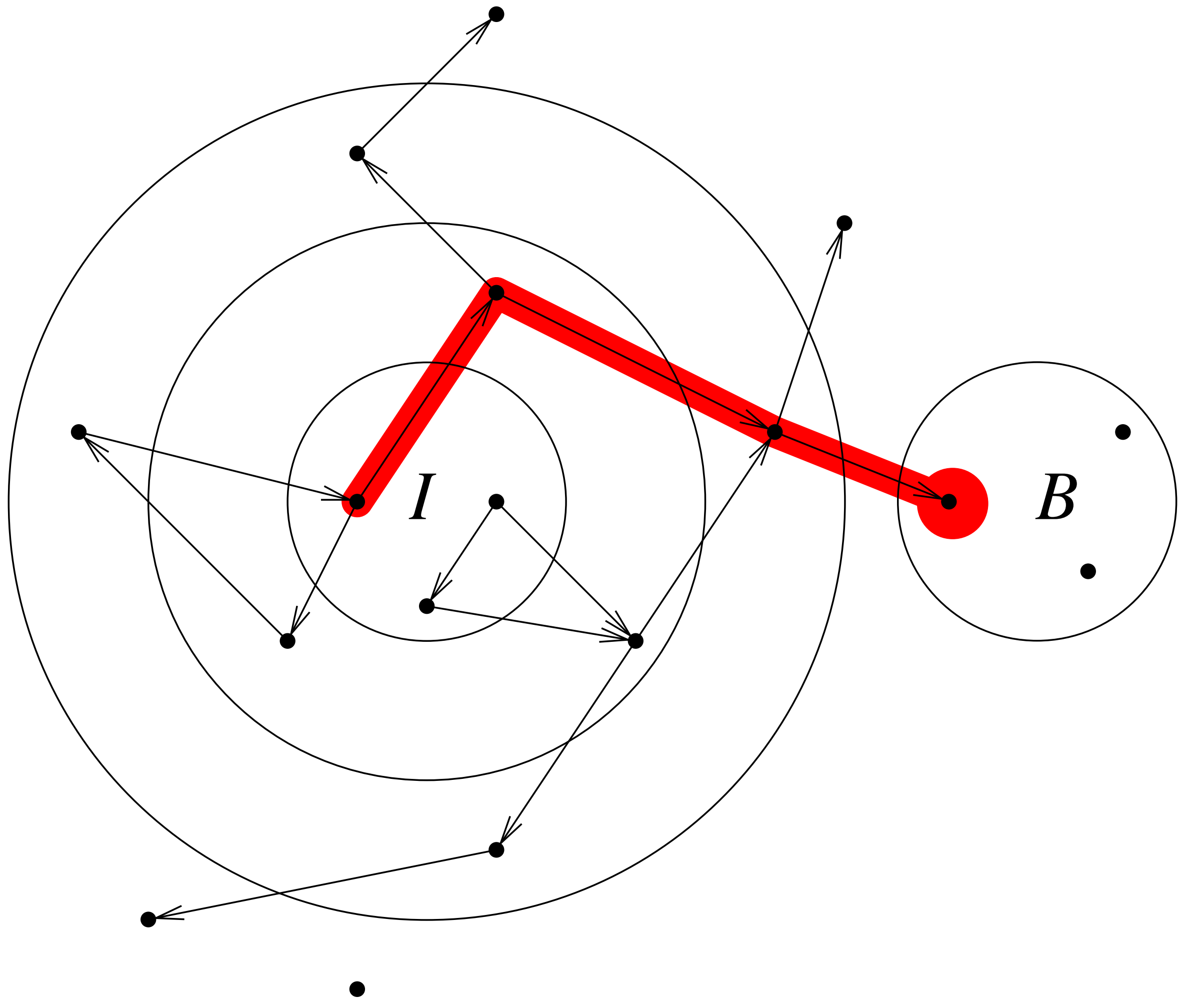
Forward Fixpoint Algorithm: Step 2

A. Biere, FMV, JKU Linz



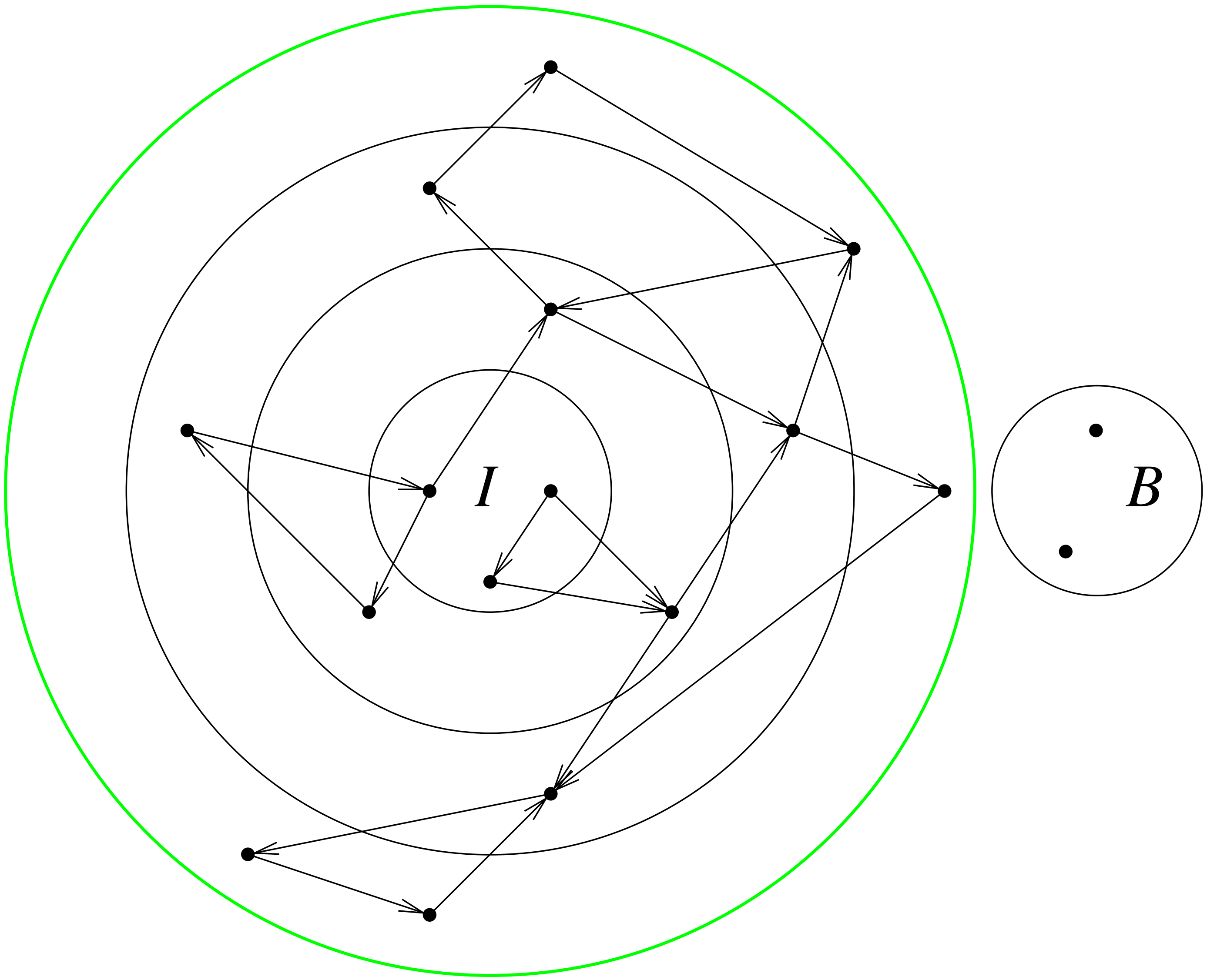
Forward Fixpoint Algorithm: Step 3




I
B

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

<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;

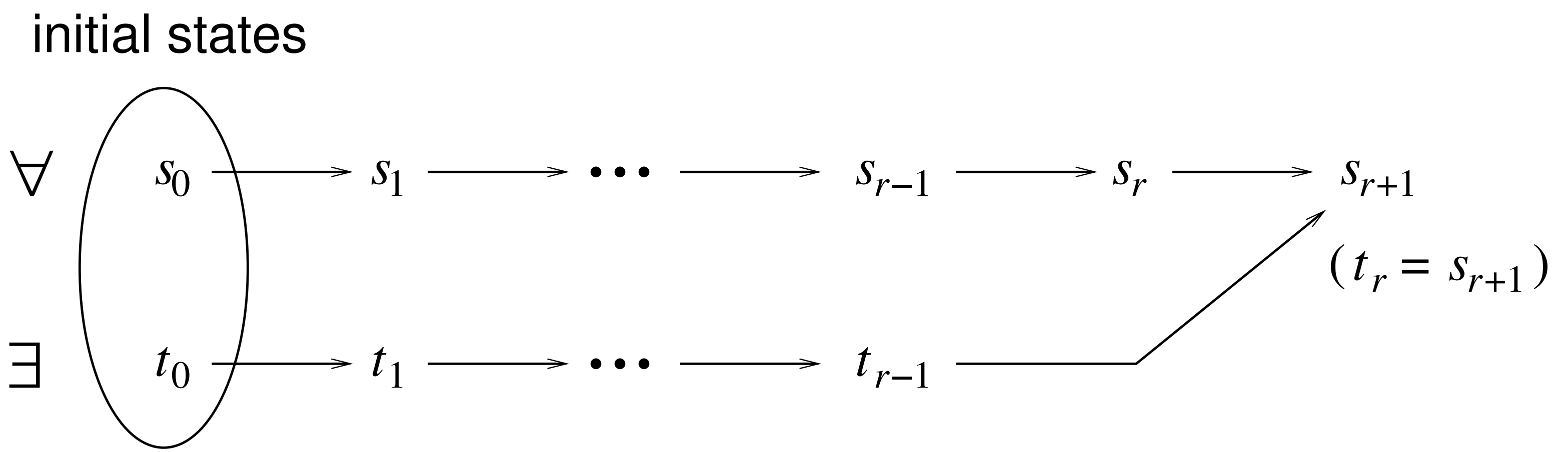
return "no bad state reachable";

symbolic model checking represents set of states in this BFS symbolically

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?	$S_C^2 \neq S_N^2$	$\exists s_0, s_1, s2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \neg (I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)])]$
2: terminate?	$S_C^2 = S_N^2$	$\forall s_0, s_1, s2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]$
2: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

$$\forall s_0,\ldots,s_{r+1}[\,I(s_0)\wedge\bigwedge_{i=0}^{r}T(s_i,s_{i+1})\rightarrow$$

$$\exists t_0,\ldots,t_r[\,I(t_0)\ \wedge\ s_{r+1}=t_r\ \wedge\ \bigwedge_{i=0}^{r-1}(t_i=t_{i+1}\ \vee\ T(t_i,t_{i+1}))]]$$

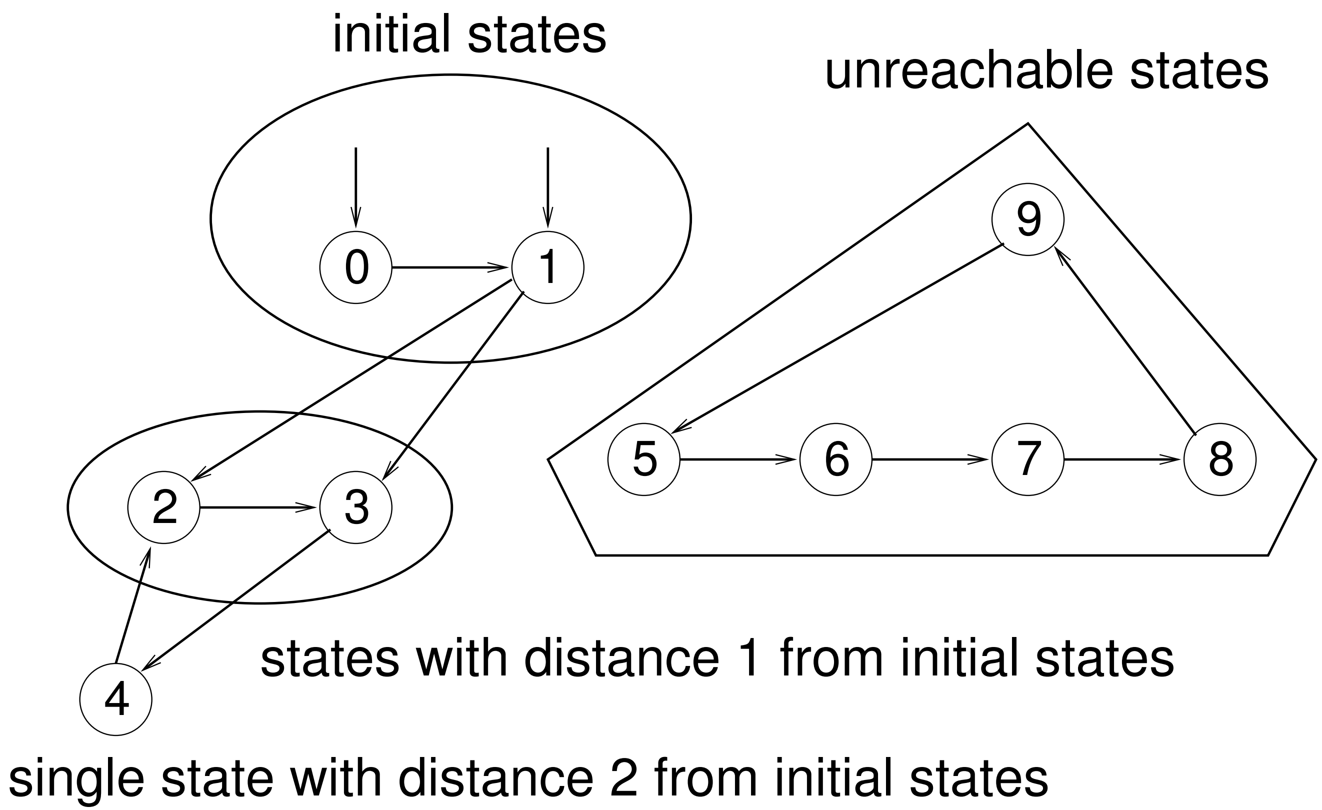


(we allow t_{i+1} to be identical to t_i in the lower path)

radius is smallest r for which formula is true

Radius Example

A. Biere, FMV, JKU Linz



Termination Check in Symbolic Reachability is in QBF

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

$$\forall s_0, s_1, s2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]$$

- not **eliminating quantifiers** results in QBF with one alternation
  - checking whether bad state is reached only needs SAT
  - number iterations bounded by radius $r = O(2^n)$
- so why not forget about termination and concentrate on bug finding?

$\Rightarrow$ Bounded Model Checking (BMC)

BMC Part of Fixpoint Algorithm

A. Biere, FMV, JKU Linz

~~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? $B \cap S_N^0 \neq \emptyset$ $\boxed{\exists s_0[I(s_0) \wedge B(s_0)]}$

~~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? $B \cap S_N^1 \neq \emptyset$ $\boxed{\exists s_0, s_1[I(s_0) \wedge T(s_0, s_1) \wedge B(s_1)]}$

~~2: continue?~~ $S_C^2 \neq S_N^2$ $\exists s_0, s_1, s2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge$
$\neg(I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]])]$

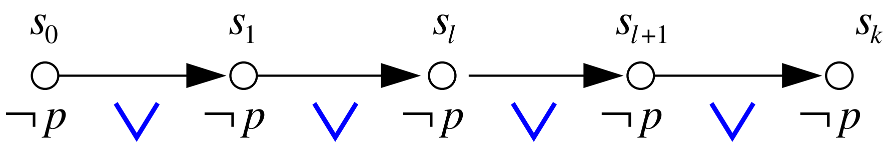
~~2: terminate?~~ $S_C^2 = S_N^2$ $\forall s_0, s_1, s2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow$
$I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]]$

2: bad state? $B \cap S_N^1 \neq \emptyset$ $\boxed{\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]}$

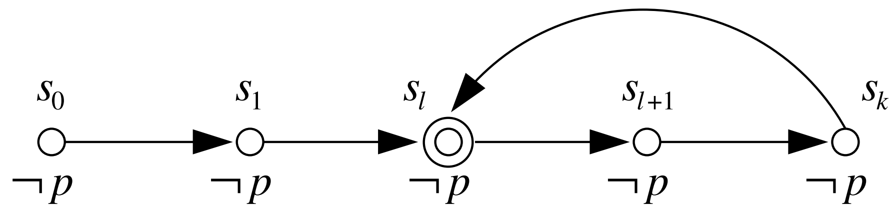
Bounded Model Checking (BMC)

[BiereCimattiClarkeZhu TACAS'99]

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



or



- 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 $\mathbf{F}p$

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

- 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]

Transitive Closure

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

(assuming $\quad = \subseteq T$)

Standard Linear Unfolding

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

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

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

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 $\cup \{v \mapsto$ *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

- 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