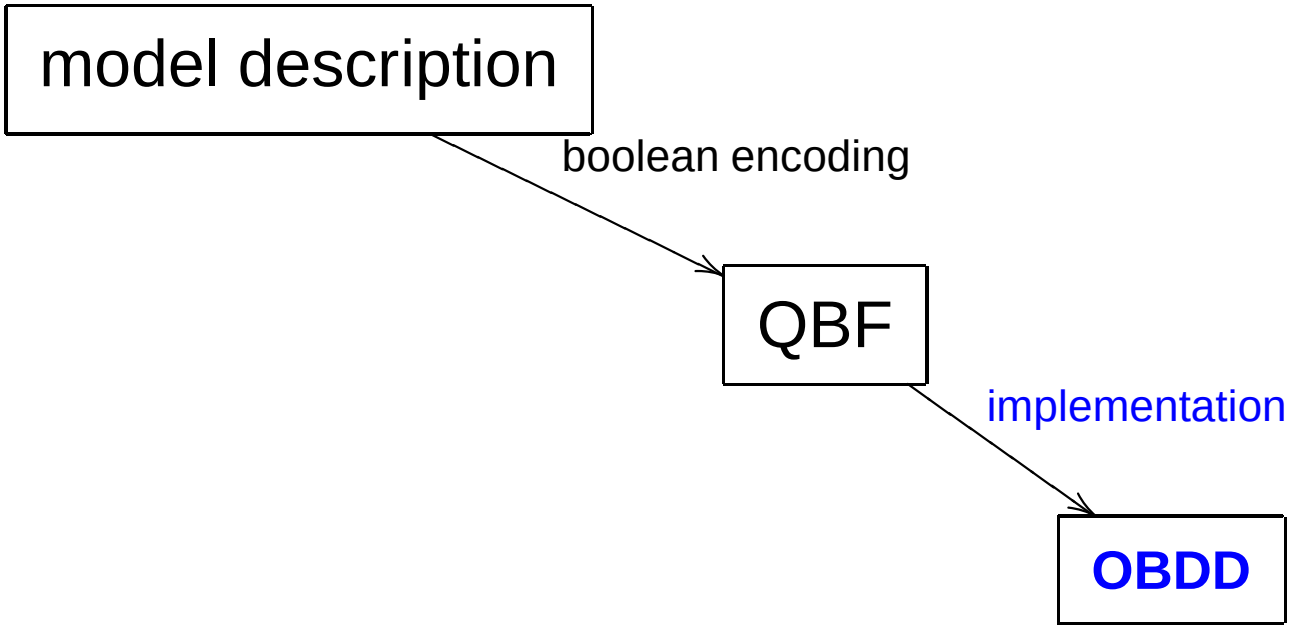
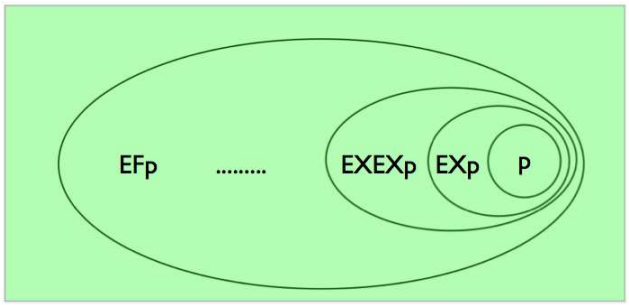
Computer aided verification

Lecture 8: Bounded model checking





model checking = operations on OBDDs

Example: $\mathsf{EF}\, p$ (safety)

pre-SMC:

$$\ldots \supseteq p \cup \mathsf{EX}(p \cup \mathsf{EX}\, p) \supseteq p \cup \mathsf{EX}\, p \supseteq p \supseteq \texttt{false}$$

$$\ldots \supseteq p \cup \mathrm{pre}(p \cup \mathrm{pre}(p)) \supseteq p \cup \mathrm{pre}(p) \supseteq p \supseteq \texttt{false}$$

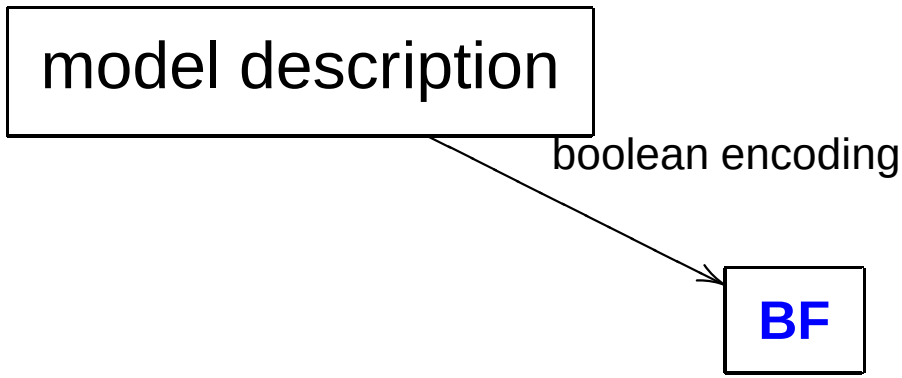
post-SMC:

$$S_0 \subseteq S_0 \cup \mathrm{post}(S_0) \subseteq S_0 \cup \mathrm{post}(S_0 \cup \mathrm{post}(S_0)) \subseteq \ldots$$

$$\exists \vec{z}_0 \exists \vec{z}_1 \ldots \exists \vec{z}_k \qquad \vec{z}_0 \longrightarrow \vec{z}_1 \longrightarrow \vec{z}_2 \longrightarrow \cdots \longrightarrow \vec{z}_{k-1} \longrightarrow \vec{z}_k$$

$$\vec{z} = \vec{z}_0 \vee \vec{z} = \vec{z}_1 \vee \ldots \vee \vec{z} = \vec{z}_k$$

Bounded model checking (BMC)



model checking = satisfiability of boolean formula

$$\vec{z}_0 \longrightarrow \vec{z}_1 \longrightarrow \vec{z}_2 \longrightarrow \quad \cdots \quad \longrightarrow \vec{z}_{k-1} \longrightarrow \vec{z}_k$$

$$p(\vec{z}_0) \ \vee \ p(\vec{z}_1) \ \vee \ p(\vec{z}_2) \ \vee \ \ldots \ \vee \ p(\vec{z}_k)$$

BMC

- searching for a counterexample of bounded length
- symbolic path
- application of SAT-solvers

I. BMC

$$M \models \mathsf{E}\,\phi \qquad (\phi \in \mathsf{LTL}^{+})$$

e.g.. instead of $M \models \mathsf{AG}\,\phi$, we check $M \models \mathsf{EF}\,\neg\phi$

M described by boolean formulas as usual:

$$R(\vec{z}, \vec{z'}), \quad L_p(\vec{z}), \quad S_0(\vec{z})$$

$$R(z_1, \ldots, z_m, z'_1, \ldots, z'_m), \quad L_p(z_1, \ldots, z_m), \quad S_0(z_1, \ldots, z_m)$$

Bound

Idea: Horizon $k \geq 0$ — k system steps

$\Pi \models_k \phi$ — k steps are sufficient to decide that $\Pi \models \phi$

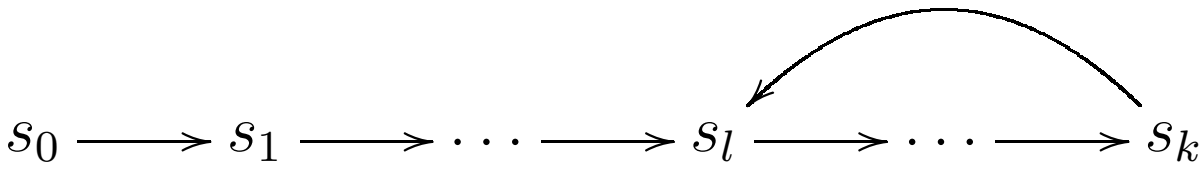
Lem.: $\Pi \models_k \phi \implies \Pi \models \phi$

Lem.: $M \models \mathsf{E}\,\phi \implies \exists k \geq 0.\ M \models_k \mathsf{E}\,\phi$ — loops!

Thm.: $M \models \mathsf{E}\,\phi \iff \exists k \geq 0.\ M \models_k \mathsf{E}\,\phi$

$$\Pi \quad = \quad s_0 \to s_1 \to s_2 \to \dots$$

(k, l) - loop:



$$s_{k+1+i} = s_{l+i}$$

(k - loop)

if there exists a back loop from s_k

no-loop:

$$s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k \longrightarrow \cdots$$

Bounded semantics for k-loops

$$\Pi \models_k \phi \iff \Pi \models \phi$$

$$\Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$$

Bounded semantics for no-loops

$$\Pi \models_k \phi \iff \Pi \models_k^0 \phi \qquad \Pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$$

$$\models_k^i , \quad 0 \leq i \leq k$$

Bounded semantics for no-loops

$\Pi \models_k \phi \iff \Pi \models_k^0 \phi$ $\qquad$ $\Pi = s_0 \to s_1 \to s_2 \to \ldots$

$$\models_k^i \ , \ \ 0 \leq i \leq k$$

$\Pi \models_k^i p \iff p \in L(s_i)$

$\Pi \models_k^i \neg p, \ \ \phi_1 \wedge \phi_2, \ \ \phi_1 \vee \phi_2 \iff \ldots$

$\Pi \models_k^i \mathsf{X}\,\phi \iff i{<}k \ \text{ i } \ \Pi \models_k^{i+1} \phi$

$\Pi \models_k^i \mathsf{F}\,\phi \iff \exists j, \ i{\leq}j{\leq}k.\ \Pi \models_k^j \phi$

$\Pi \models_k^i \mathsf{G}\,\phi$ never

$\Pi \models_k^i \phi\,\mathsf{U}\,\psi \iff \exists j, \ i{\leq}j{\leq}k.\ \Pi \models_k^j \psi \ \wedge \ \forall l, \ i{\leq}l{<}j.\ \Pi \models_k^l \phi$

$\Pi \models_k^i \phi\,\mathsf{R}\,\psi \iff$?

Bound

Tw.: $M \models \mathsf{E}\,\phi \iff \exists k \geq 0.\ M \models_k \mathsf{E}\,\phi$

- **what k is sufficiently big? graph diameter?**
- **no completeness !**
- **efficiency of SMC depends on graph diameter**

$$M \models_k \mathsf{E}\,\phi \qquad (\phi \in \mathsf{LTL}^+)$$

$M,\ \phi,\ k \quad \mapsto \quad [M, \phi]_k$

$M \models_k \phi \iff [M, \phi]_k$ satisfiable

M described by boolean formulas as usual:

$R(\vec{z}, \vec{z'}), \quad L_p(\vec{z}), \quad S_0(\vec{z})$

$R(z_1, \ldots, z_m, z'_1, \ldots, z'_m), \quad L_p(z_1, \ldots, z_m), \quad S_0(z_1, \ldots, z_m)$

Boolean encoding

Symbolic path

$$\vec{z}_0 \longrightarrow \vec{z}_1 \longrightarrow \vec{z}_2 \longrightarrow \quad \cdots \quad \longrightarrow \vec{z}_{k-1} \longrightarrow \vec{z}_k$$

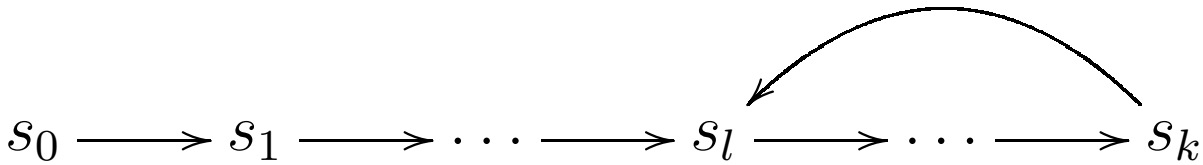
$$[M]_k \quad := \quad S_0(\vec{z}_0) \ \wedge \bigwedge_{i=0}^{k-1} R(\vec{z}_i, \vec{z}_{i+1})$$

$(k+1) \cdot m$ boolean variables

$[M]_k$ is of size $\mathcal{O}(k \cdot |M|)$

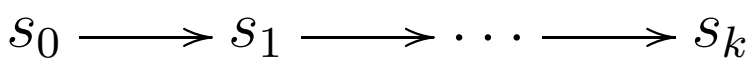
Boolean encoding

(k, l)-loop:



$$ {}_lL_k \equiv R(\vec{z}_k, \vec{z}_l) \qquad L_k \equiv \bigvee_{l=0}^{k} {}_lL_k $$

no-loop:



Boolean encoding

$$[M,\phi]_k \;:=\; [M]_k \;\wedge\; \Big((\neg L_k \wedge [\phi]^0_k) \vee \bigvee_{l=0}^{k} ({}_lL_k \wedge {}_l[\phi]^0_k)\Big)$$

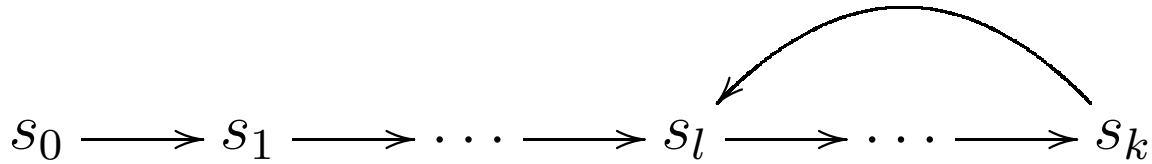
${}_l[\phi]^0_k$ – semantics of ϕ on (k,l)-loop

$[\phi]^0_k$ – semantics of ϕ on no-loop

$[M,\phi]_k$ is of size $\mathcal{O}(k \cdot (|M| + |\phi|))$

Boolean encoding

$${}_l[\phi]_k^i \qquad 0 \le l, i \le k$$



$${}_l[p]_k^i \iff L_p(\vec{z}_i)$$

$${}_l[\neg p]_k^i \qquad {}_l[\phi \wedge \psi]_k^i \qquad {}_l[\phi \vee \psi]_k^i \iff {}_l[\phi]_k^i \ \vee \ {}_l[\phi]_k^i$$

$${}_l[\,\mathsf{X}\,\phi]_k^i \iff {}_l[\phi]_k^{\mathrm{succ}(i)}$$

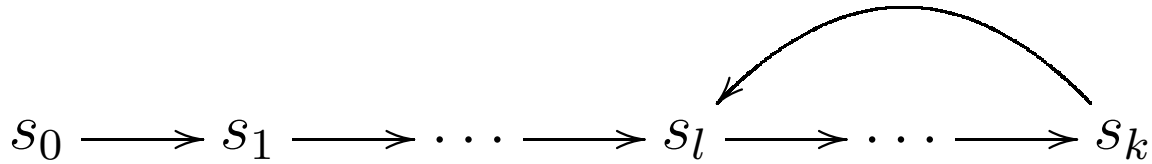
$$\mathrm{succ}(i) = \begin{cases} i+1 & i < k \\ l & i = k \end{cases}$$

$$\mathsf{F}\,\phi \equiv \phi \vee \mathsf{X}\,\mathsf{F}\,\phi \qquad \phi\,\mathsf{U}\,\psi \equiv \psi \vee (\phi \wedge \mathsf{X}\,\phi\,\mathsf{U}\,\psi)$$

$$\mathsf{G}\,\phi \equiv \phi \wedge \mathsf{X}\,\mathsf{G}\,\phi \qquad \phi\,\mathsf{R}\,\psi \equiv \psi \wedge (\phi \vee \mathsf{X}\,\phi\,\mathsf{R}\,\psi)$$

Boolean encoding

${}_l[\phi]_k^i \qquad 0 \leq l, i \leq k$



${}_l[p]_k^i \iff L_p(\vec{z}_i)$

${}_l[\neg p]_k^i \qquad {}_l[\phi \wedge \psi]_k^i \qquad {}_l[\phi \vee \psi]_k^i \iff {}_l[\phi]_k^i \vee {}_l[\phi]_k^i$

${}_l[\,\mathsf{X}\,\phi]_k^i \iff {}_l[\phi]_k^{\mathrm{succ}(i)}$

$$\mathrm{succ}(i) = \begin{cases} i+1 & i < k \\ l & i = k \end{cases}$$

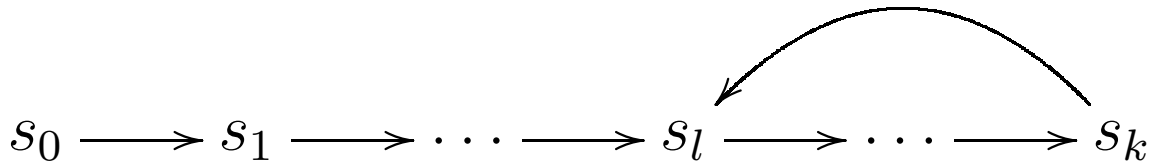
${}_l[\,\mathsf{F}\,\phi]_k^i \iff {}_l[\phi]_k^i \vee {}_l[\,\mathsf{F}\,\phi]_k^{\mathrm{succ}(i)}$

${}_l[\,\mathsf{G}\,\phi]_k^i \iff {}_l[\phi]_k^i \wedge {}_l[\,\mathsf{G}\,\phi]_k^{\mathrm{succ}(i)}$

${}_l[\phi\,\mathsf{U}\,\psi]_k^i \iff {}_l[\psi]_k^i \vee ({}_l[\phi]_k^i \wedge {}_l[\phi\,\mathsf{U}\,\psi]_k^{\mathrm{succ}(i)})$

Boolean encoding

Example:



$$_l[\,\mathsf{F}\,p]^0_k \iff L_p(\vec{z}_0) \vee \ldots \vee L_p(\vec{z}_k)$$

$$_l[\,\mathsf{G}\,p]^0_k \iff L_p(\vec{z}_0) \wedge \ldots \wedge L_p(\vec{z}_k)$$

$$_l[p\,\mathsf{U}\,q]^0_k \iff L_q(\vec{z}_0) \vee (L_p(\vec{z}_0) \wedge (L_q(\vec{z}_1) \vee (\ldots$$

$$L_q(\vec{z}_{k-1}) \vee (L_p(\vec{z}_{k-1}) \wedge L_q(\vec{z}_k))\ldots)))$$

$$_l[\,\mathsf{F}\,\mathsf{G}\,p]^0_k \iff (L_p(\vec{z}_0) \wedge L_p(\vec{z}_1) \wedge \ldots \wedge L_p(\vec{z}_k)) \vee$$

$$(L_p(\vec{z}_1) \wedge \ldots \wedge L_p(\vec{z}_k)) \vee \ \ldots \ \vee (L_p(\vec{z}_l) \wedge \ldots \wedge L_p(\vec{z}_k))$$

$$\iff (L_p(\vec{z}_l) \wedge \ldots \wedge L_p(\vec{z}_k))$$

Boolean encoding

$_l[\phi]^0_k$ is of size $\mathcal{O}(k \cdot |\phi|)$

Boolean encoding

$[\phi]_k^i \qquad 0 \leq i \leq k$

$s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k$

$[p]_k^i \qquad [\neg p]_k^i \qquad [\phi \wedge \psi]_k^i \qquad [\phi \vee \psi]_k^i \iff \ldots$ j. w.

$[\mathsf{X}\,\phi]_k^i \iff [\phi]_k^{i+1}$

$[\phi]_k^{k+1} \iff \texttt{false}$

$\mathsf{F}\,\phi \equiv \phi \vee \mathsf{X}\,\mathsf{F}\,\phi \qquad \phi\,\mathsf{U}\,\psi \equiv \psi \vee (\phi \wedge \mathsf{X}\,\phi\,\mathsf{U}\,\psi)$

$\mathsf{G}\,\phi \equiv \phi \wedge \mathsf{X}\,\mathsf{G}\,\phi \qquad \phi\,\mathsf{R}\,\psi \equiv \psi \wedge (\phi \vee \mathsf{X}\,\phi\,\mathsf{R}\,\psi)$

Boolean encoding

Example:

$$s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k$$

$$[\,\mathsf{F}\,p]_k^0 \iff L_p(\vec{z}_0) \vee \ldots \vee L_p(\vec{z}_k) \vee \texttt{false}$$

$$[\,\mathsf{G}\,p]_k^0 \iff L_p(\vec{z}_0) \wedge \ldots \wedge L_p(\vec{z}_k) \wedge \texttt{false} \equiv \texttt{false}$$

$$[p\,\mathsf{U}\,q]_k^0 \iff L_q(\vec{z}_0) \vee (L_p(\vec{z}_0) \wedge (L_q(\vec{z}_1) \vee (\ldots$$

$$L_q(\vec{z}_{k-1}) \vee (L_p(\vec{z}_{k-1}) \wedge (L_q(\vec{z}_k) \vee (L_p(\vec{z}_k) \wedge \texttt{false}))) \ldots)))$$

$$[\,\mathsf{F}\;\mathsf{G}\,p]_k^0 \iff \texttt{false}$$

$$[M,\phi]_k \;:=\; [M]_k \;\wedge\; \Big((\neg L_k \wedge [\phi]_k^0) \vee \bigvee_{l=0}^{k} ({}_l L_k \wedge {}_l[\phi]_k^0)\Big)$$

Thm.: $M \models_k \mathsf{E}\,\phi \iff [M,\phi]_k$ is satisfiable.

Repeat:

(1) $k := k_0$

(2) if $[M,\phi]_k$ satisfiable then stop $\qquad M \models \mathsf{E}\,\phi$

(3) increment k and continue

Other boolean encodings

- hardware verification:
  - circuit structure kept, designated SAT-solvers
- LTL with past
- software verification (e.g. CBMC)
  - loop unfolding, heap model
- concurrent programs
- ...

$$\vec{z}_0 \longrightarrow \vec{z}_1 \longrightarrow \vec{z}_2 \longrightarrow \quad \cdots \quad \longrightarrow \vec{z}_{k-1} \longrightarrow \vec{z}_k$$

II. Completeness ?

Is completeness achievable ?

- completeness threshold for k (safety $\mathsf{G}\, p$)
- simultaneous checks of ϕ and $\neg\phi$ (liveness $\mathsf{F}\, p$)
- induction (safety $\mathsf{G}\, p$)

Completeness for G p

Reachability diameter – bound for k

the least i such that

$$\forall \vec{z}_0, \ldots, \vec{z}_n.\ \exists \vec{z'}_0, \ldots, \vec{z'}_t,\ t \leq i.\ S_0(\vec{z}_0)\ \wedge \bigwedge_{j=0}^{n-1} R(\vec{z}_j, \vec{z}_{j+1})\ \Longrightarrow$$

$$S_0(\vec{z'}_0)\ \wedge\ \Big(\bigwedge_{j=0}^{t-1} R(\vec{z'}_j, \vec{z'}_{j+1})\Big) \wedge \vec{z'}_t = \vec{z}_n$$

Reachability diameter – bound for k

the least i such that

$$\forall \vec{z}_0, \ldots, \vec{z}_{i+1}.\ \exists \vec{z'}_0, \ldots, \vec{z'}_i.\ S_0(\vec{z}_0)\ \wedge\ \bigwedge_{j=0}^{i} R(\vec{z}_j, \vec{z}_{j+1})\ \Longrightarrow$$

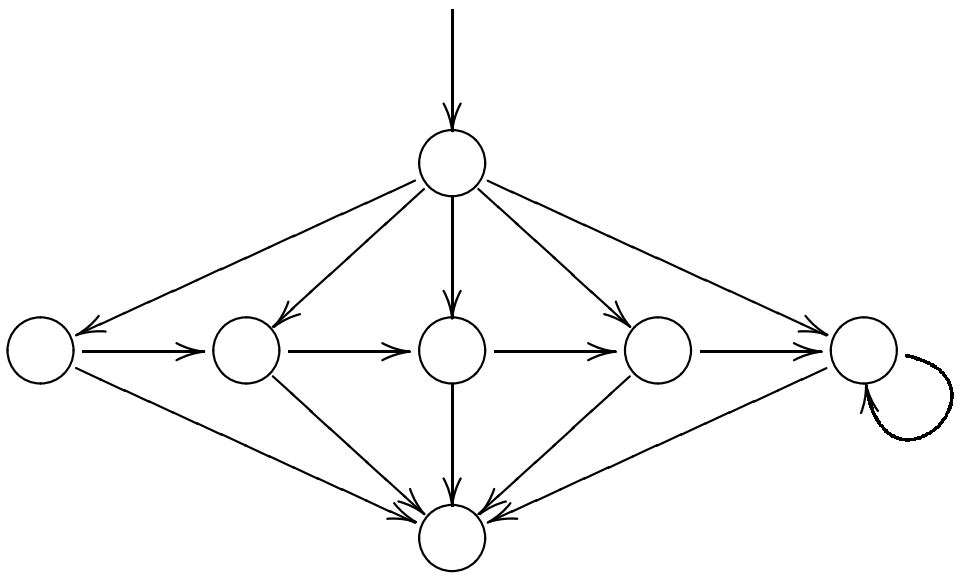
$$S_0(\vec{z'}_0)\ \wedge\ \Big(\bigwedge_{j=0}^{i-1} R(\vec{z'}_j, \vec{z'}_{j+1})\Big)\ \wedge\ \bigvee_{j=0}^{i} \vec{z'}_j = \vec{z}_{i+1}$$

Completeness for G p

the longest loop-free path – bound for k

the greatest i such that

$$\exists \vec{z'}_0, \ldots, \vec{z}_i.\ S_0(\vec{z}_0)\ \wedge\ \Big(\bigwedge_{j=0}^{i-1} R(\vec{z}_j, \vec{z}_{j+1})\Big)\ \wedge\ \bigwedge_{j=0}^{i-1}\bigwedge_{l=j+1}^{i} \vec{z}_j \neq \vec{z}_l$$



Completeness for $\mathsf{F}\,p$

$M \models \mathsf{EG}\,\neg p \iff \exists k$ s.t. the following formula is satisfiable:

$$\mathsf{No}_k(\vec{z}_0, \ldots, \vec{z}_{k+1}) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j, \vec{z}_{j+1}) \wedge \Big(\bigvee_{l=0}^{k} {}_l L_k\Big) \wedge \bigwedge_{j=0}^{k} \neg L_p(\vec{z}_j)$$

$M \models \mathsf{AF}\,p \iff \exists k$ s.t. the following formula is a tautology:

$$\mathsf{Yes}_k(\vec{z}_0, \ldots, \vec{z}_{k+1}) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j, \vec{z}_{j+1}) \Longrightarrow \bigvee_{j=0}^{k} L_p(\vec{z}_j)$$

Completeness for $\mathsf{F}\,p$

$M \models \mathsf{EG}\,\neg p \iff \exists k$ s.t. the following formula is satisfiable:

$$\mathsf{No}_k(\vec{z}_0,\ldots,\vec{z}_{k+1}) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j,\vec{z}_{j+1}) \wedge \Big(\bigvee_{l=0}^{k} {}_l L_k\Big) \wedge \bigwedge_{j=0}^{k} \neg L_p(\vec{z}_j)$$

$M \models \mathsf{AF}\,p \iff \exists k$ s.t. the following formula is unsatisfiable:

$$\neg\mathsf{Yes}_k(\vec{z}_0,\ldots,\vec{z}_{k+1}) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j,\vec{z}_{j+1}) \wedge \bigwedge_{j=0}^{k} \neg L_p(\vec{z}_j)$$

Completeness for F p

Repeat:

(1) $k := 0$

(2) if No_k satisfiable then stop $\qquad M \models \mathsf{EG}\,\neg\phi$

(3) if $\neg\text{Yes}_k$ unsatisfiable then stop $\qquad M \models \mathsf{F}\,p$

(4) increase k and continue

Completeness for G p – induction

– induction base

check that the following formula is unsatisfiable:

$$\text{Base}_k(\vec{z}_0, \ldots, \vec{z}_k) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j, \vec{z}_{j+1}) \wedge \bigvee_{j=0}^{k} \neg L_p(\vec{z}_j)$$

– induction step

check that the following formula is unsatisfiable:

$$\text{Step}_k(\vec{z}_0, \ldots, \vec{z}_{k+1}) \equiv \bigwedge_{j=0}^{k} (L_p(\vec{z}_j) \wedge R(\vec{z}_j, \vec{z}_{j+1})) \wedge \neg L_p(\vec{z}_{k+1})$$

Why do we need $k > 1$?

Repeat:

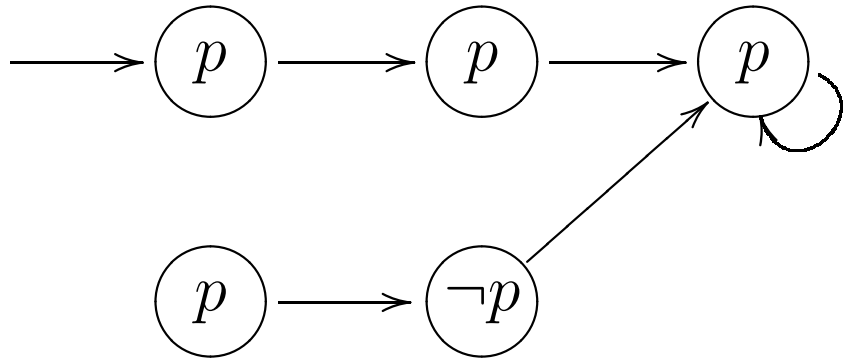
(1) $k := 0$

(2) if Base$_k$ satisfiable then stop $\qquad M \models \mathsf{EF}\,\neg\phi$

(3) if Step$_k$ unsatisfiable then stop $\qquad M \models \mathsf{G}\,p$

(4) increase k and continue

Why do we need $k > 1$?



Repeat:

(1) $k := 0$

(2) if Base$_k$ satisfiable then stop $\quad M \models \mathsf{EF}\,\neg\phi$

(3) if Step$_k$ unsatisfiable then stop $\quad M \models \mathsf{G}\,p$

(4) increase k and continue

III. BDD or SAT ?

Comparison

bit	k	SMV$_2$	MB	PROVER	MB
0	1	25	79	< 1	1
1	2	25	79	< 1	1
2	3	26	80	< 1	1
3	4	27	82	1	2
4	5	33	92	1	2
5	6	67	102	1	2
6	7	258	172	2	2
7	8	1741	492	7	3
8	9		$>$1GB	29	3
9	10			58	3
10	11			91	3
11	12			125	3
12	13			156	4
13	14			186	4
14	15			226	4
15	16			183	5

[Biere, Cimatti, Clarke, Strichman, Zhu 2003]

BDD or SAT?

- the methods seem complementary
- SAT more predictable
- BDD is memory-consuming and SAT time-consuming
- BDD is complete, SAT no
- unbounded model checking UMC:

  CNF instead of BDD in SMC
- BDD + SAT

Progress

1981–1982: EMC — 10^5 states

1990–1992: SMV — 10^{100} states

2000': BMC + CEGAR — 10^{1000} states

IV. SAT-solvers

SAT-solvers

- CNF
- algorithm DPLL
  - searching through partial valuations tree
  - Boolean Constraint Propagation (BCP)
  - conflicts – pruning of the tree
- heuristics

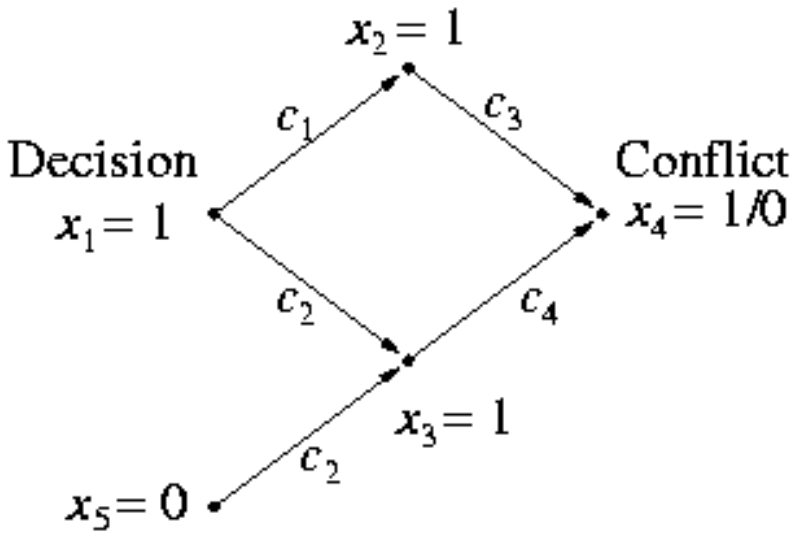
Implications graph and BCP

$$c_1 = (\neg x_1 \vee x_2)$$

$$c_2 = (\neg x_1 \vee x_3 \vee x_5)$$

$$c_3 = (\neg x_2 \vee x_4)$$

$$c_4 = (\neg x_3 \vee \neg x_4)$$



[Biere, Cimatti, Clarke, Strichman, Zhu 2003]

DPLL

```
// Input arg: Current decision level d
// Return value:
//   SAT():      {SAT, UNSAT}
//   Decide():   {DECISION, ALL-DECIDED}
//   Deduce():   {OK, CONFLICT}
//   Diagnose():{SWAP, BACK-TRACK} also calculates β

SAT (d)
   {
l1:    if (Decide (d) == ALL-DECIDED) return SAT;
l2:    while (TRUE) {
l3:       if (Deduce(d) != CONFLICT) {
l4:          if (SAT (d+1) == SAT) return SAT;
l5:          else if (β < d || d == 0)
l6:             { Erase (d); return UNSAT; }
          }
l7:       if (Diagnose (d) == BACK-TRACK) return UNSAT;
       }
   }
```

[Biere, Cimatti, Clarke, Strichman, Zhu 2003]

Why SAT-solvers are so fast?

- conflict learning – adding conflict clauses
- non-chronological backtracking
- heuristics for decisions
- efficient data structures
- incremental satisfiability