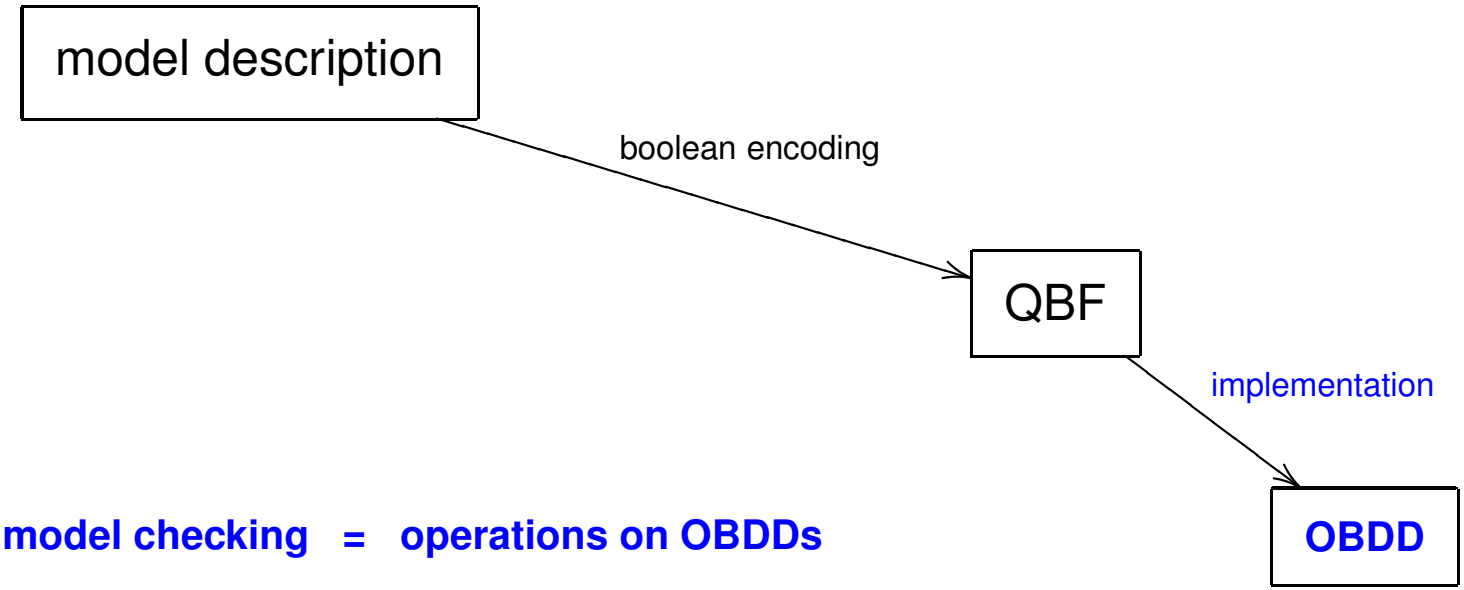
Computer aided verification

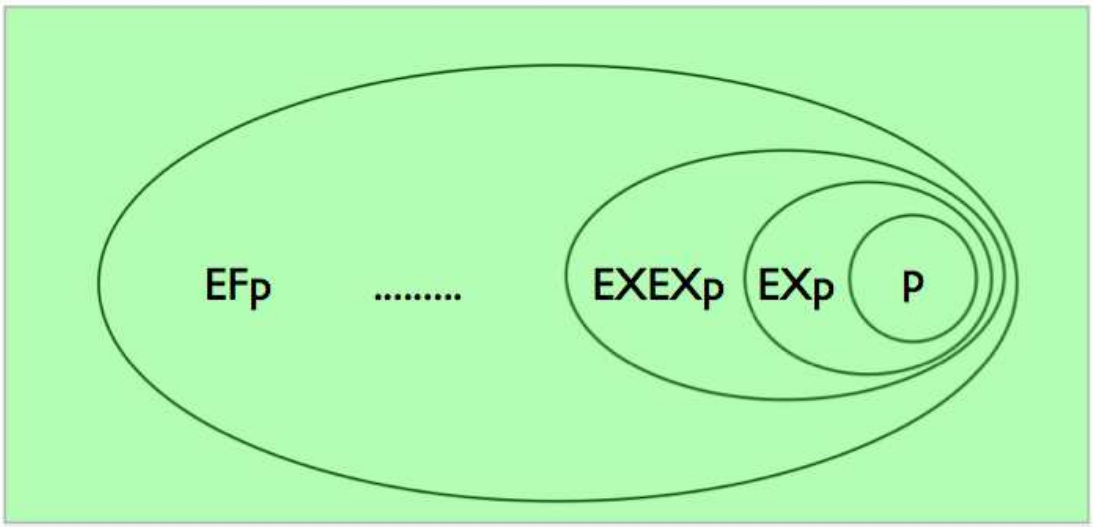
Lecture 7:

Bounded model checking

Sławomir Lasota
University of Warsaw



model checking = operations on OBDDs



SMC $\rightsquigarrow$ BMC

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

pre-SMC:

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

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

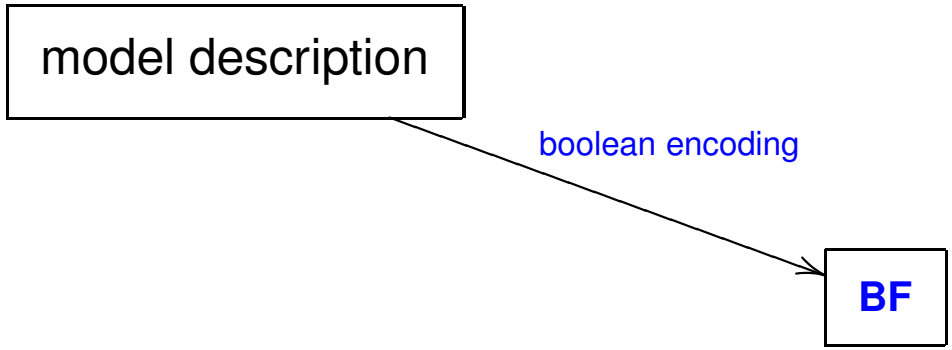
post-SMC:

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

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

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

Bounded model checking (BMC)



model checking = satisfiability of boolean formula

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

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

BMC for LTL

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

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:

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

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

Bound

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

$\Pi \models_k \phi$ — depth up to k is sufficient to decide whether $\Pi \models \phi$

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

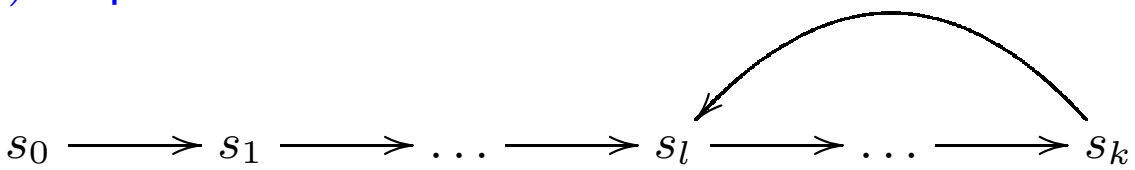
Lem.: $M \models \mathsf{E}\,\phi \implies \exists k \geq 0.\ M \models_k \mathsf{E}\,\phi$ (lasso)

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

k-loops and no-k-loops

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

(k, l)-loop:



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

(k-loop)

no-k-loop:

Bounded semantics for k-loops

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

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

Bounded semantics for no-k-loops

$\Pi \models_k \phi \iff \Pi \models_k^0 \phi$

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

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

Bounded semantics for no-k-loops

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

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

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

$\Pi \models_k^i \neg p \iff \Pi \not\models_k^i p$

$\Pi \models_k^i \phi_1 \wedge \phi_2 \iff \Pi \models_k^i \phi_1$ and $\Pi \models_k^i \phi_2$ $\qquad$ $\Pi \models_k^i \phi_1 \vee \phi_2 \iff \ldots$

$\Pi \models_k^i \mathsf{X}\phi \iff i{<}k$ 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

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

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

Bounded semantics of R

$$\phi \,\mathsf{R}\, \psi \;\equiv\; \neg(\neg\phi U \neg\psi)$$

$$\phi \,\mathsf{R}\, \psi \;\equiv\; \psi \,\mathsf{U}\, (\psi \wedge \phi) \;\vee\; \mathsf{G}\, \psi$$

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

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

Bounded model checking loop

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

Repeat:

(1) $k := k_0$

(2) if $M \models_k \mathsf{E}\,\phi$ then stop

(3) increment k and continue

Bounded model checking loop

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

Repeat:

(1) $k := k_0$

(2) if $M \models_k \mathsf{E}\,\phi$ then stop

(3) increment k and continue

- **What k is sufficiently big?**
- **No completeness !**
- **Efficiency of SMC depends on graph diameter. Is it so in case of BMC?**

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

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

$M \models_k \mathsf{E}\,\phi \iff [M, \phi]_k$ satisfiable

M described by boolean formulas as usual:

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

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

Boolean encoding

Symbolic path

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

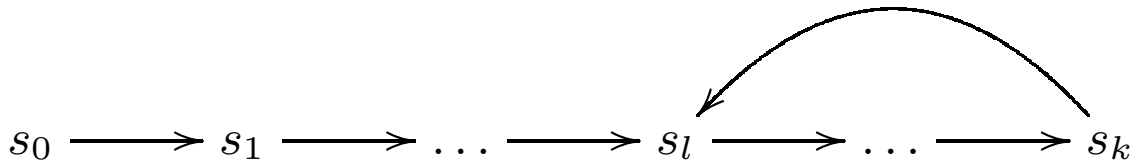
$$[M]_k \;:=\; S_0(\vec{z}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} R(\vec{z}_i, \vec{z}_{i+1})$$

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

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

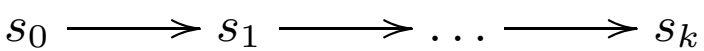
Boolean encoding

k-loop:



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

no-k-loop:



$$ \neg L_k $$

Boolean encoding

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

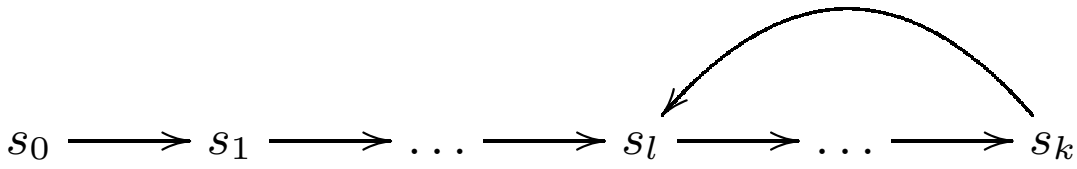
${}_l[\phi]_k^0$ – boolean encoding of semantics of ϕ on (k,l)-loop

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

$[M,\phi]_k$ is of size $\mathcal{O}(k^2 \cdot (|M| + |\phi|))$

Boolean encoding

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



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

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

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

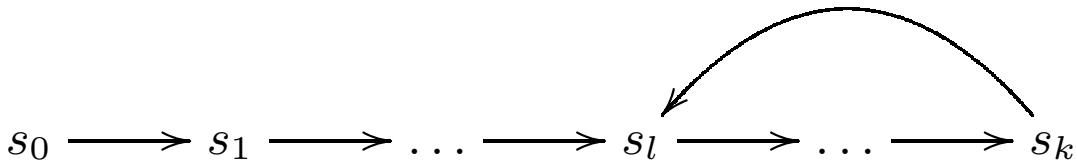
$$\mathrm{succ}(i) = \begin{cases} i+1 & i < k \\ l & i = k \end{cases}$$

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

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

Boolean encoding

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



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

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

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

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

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

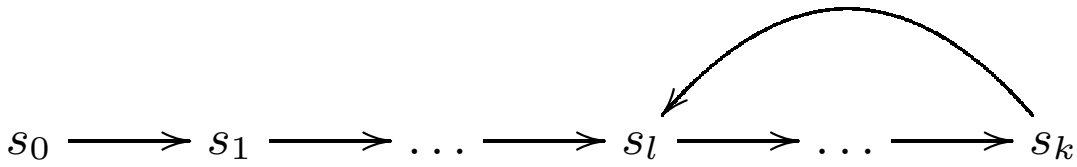
${}_l[\,\mathsf{G}\,\phi]^i_k \iff {}_l[\phi]^i_k \wedge {}_l[\,\mathsf{G}\,\phi]^{\mathrm{succ}(i)}_k$

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

${}_l[\phi\,\mathsf{R}\,\psi]^i_k \iff \ldots$

Boolean encoding

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



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

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

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

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

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

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

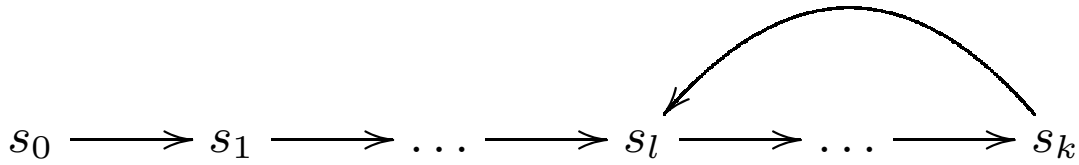
${}_l[\phi\,\mathsf{U}\,\psi]^i_k \iff {}_l[\psi]^i_k \vee ({}_l[\phi]^i_k \wedge {}_l[\phi\,\mathsf{U}\,\psi]^{\mathrm{succ}(i)}_k)$

${}_l[\phi\,\mathsf{R}\,\psi]^i_k \iff \ldots$

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

Boolean encoding

Examples:



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

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

$$\begin{aligned} _l[p\,\mathsf{U}\,q]_k^0 \iff\ & L_q(\vec{z}_0) \vee (L_p(\vec{z}_0) \wedge (L_q(\vec{z}_1) \vee (\ldots \\ & L_q(\vec{z}_{k-1}) \vee (L_p(\vec{z}_{k-1}) \wedge L_q(\vec{z}_k)) \ldots))) \end{aligned}$$

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

Boolean encoding

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

$s_0 \longrightarrow s_1 \longrightarrow \ldots \longrightarrow s_k$

$[p]_k^i \qquad [\neg p]_k^i \qquad [\phi \wedge \psi]_k^i \qquad [\phi \vee \psi]_k^i \iff \ldots$ as before

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

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

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

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

Boolean encoding

Example:

$$s_0 \longrightarrow s_1 \longrightarrow \ldots \longrightarrow s_k$$

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

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

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

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

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

BMC

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

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)
  - nested loops unfolding, heap model
- concurrent programs
- ...

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

CBMC example

```
while(B1) {
  S1;
  while(B2) {
   S2;
  }
}
```

$\longrightarrow$

```
while (vpc<=2) {
  switch (vpc) {
   case 1:
    if (B1)
     { S1; vpc = 2; }
    else
     vpc=3;
    break;
   case 2:
    if (B2) S2 else vpc=1;
    break; } }
```

```
x=x+y;
if(x!=1)
   x=2;
else
   x++;

assert(x<=3);
```

$\rightarrow$

```
x1=x0+y0;
if(x1!=1)
   x2=2;
else
   x3=x1+1;

x4=(x1!=1)?x2:x3;
assert(x4<=3);
```

$\rightarrow$

$$C := x_1 = x_0 + y_0 \ \wedge \ x_2 = 2 \ \wedge \ x_3 = x_1 + 1 \ \wedge \ x_4 = (x_1 \neq 1) ? x_2 : x_3$$

$$P := x_4 \leq 3$$

[Kroening, Clarke, Yorav 2003]

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 $\mathsf{G}\, p$)

Repeat:

(1) $k := k_0$

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

(3) increment k and continue

Completeness threshold for $\mathbf{G}\,p$

Reachability diameter – bound for k

the least i such that

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

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

Completeness threshold for G p

the longest loop-free path – bound for k

the greatest i such that

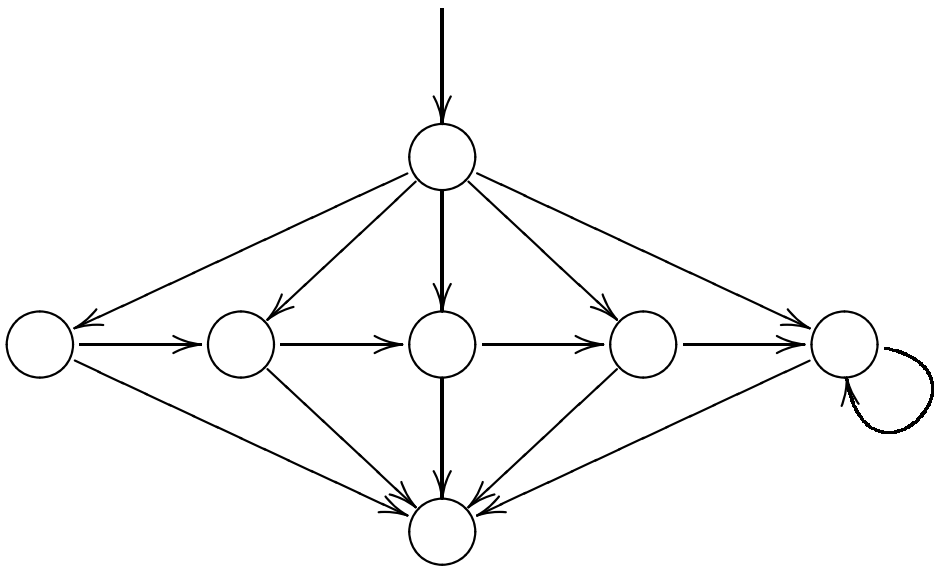
$$\exists \vec{y}_0, \ldots, \vec{z}_i.\ S_0(\vec{z}_0) \ \wedge \ \Big(\bigwedge_{j=0}^{i-1} R(\vec{z}_j, \vec{z}_{j+1}) \Big) \ \wedge \ \bigwedge_{j=0}^{i-1} \bigwedge_{l=j+1}^{i} \vec{z}_j \neq \vec{z}_l$$

Completeness threshold for G p

the longest loop-free path – bound for k

the greatest i such that

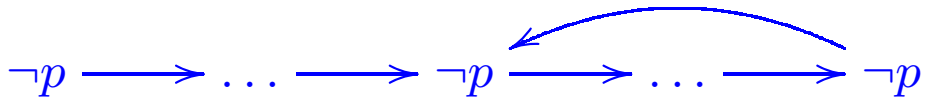
$$\exists \vec{y}_0, \ldots, \vec{z}_i.\ S_0(\vec{z}_0) \ \wedge \ \Big(\bigwedge_{j=0}^{i-1} R(\vec{z}_j, \vec{z}_{j+1}) \Big) \ \wedge \ \bigwedge_{j=0}^{i-1} \bigwedge_{l=j+1}^{i} \vec{z}_j \neq \vec{z}_l$$



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

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

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



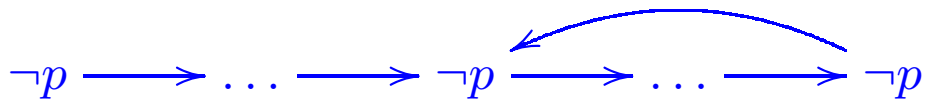
$M \models \mathsf{AF}\,p \iff \exists k$ s.t. the following formula is a tautology:

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

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

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

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



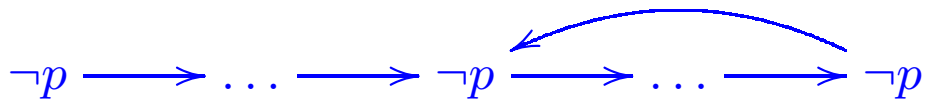
$M \models \mathsf{AF}\,p \iff \exists k$ s.t. the following formula is unsatisfiable:

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

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

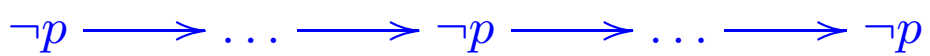
$M \models \mathsf{EG}\,\neg p \iff \exists k$ s.t. the following formula is satisfiable:

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



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

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



$$\mathsf{No}_k(\vec{z}_0,\ldots,\vec{z}_{k+1}) \equiv \neg\mathsf{Yes}_k(\vec{z}_0,\ldots,\vec{z}_{k+1}) \wedge L_k$$

Completeness for F p

Repeat:

(1) $k := 0$

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

(3) if $\neg p \longrightarrow \ldots \longrightarrow \neg p \longrightarrow \ldots \longrightarrow \neg p$ unsatisfiable then stop ($M \models \mathsf{F}\,p$)

(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:

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

Completeness for G p – induction

Repeat:

(1) $k := 0$

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

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

(4) increase k and continue

Completeness for G p – induction

Repeat:

(1) $k := 0$

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

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

(4) increase k and continue

Question: Why do we need $k > 1$?

Completeness for G p – induction

Repeat:

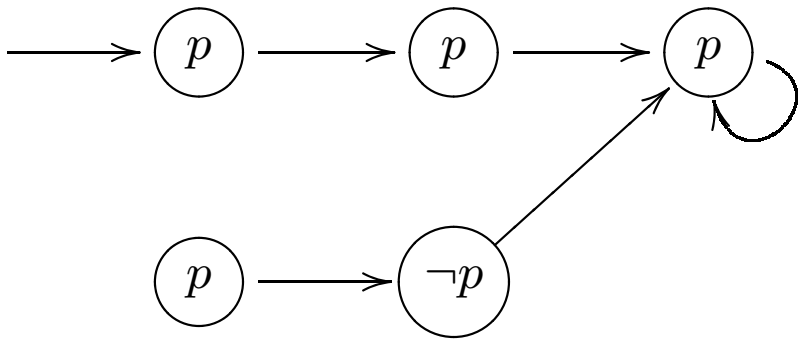
(1) $k := 0$

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

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

(4) increase k and continue

Question: Why do we need $k > 1$?



BDD or SAT ?

Comparison

– Hard problem for BDDs: 16x16 bit sequential shift and add multiplier

	SMV_1		SMV_2		SATO		PROVER	
bit	sec	MB	sec	MB	sec	MB	sec	MB
0	919	13	25	79	0	0	0	1
1	1978	13	25	79	0	0	0	1
2	2916	13	26	80	0	0	0	1
3	4744	13	27	82	0	0	1	2
4	6580	15	33	92	2	0	1	2
5	10803	25	67	102	12	0	1	2
6	43983	73	258	172	55	0	2	2
7	>17h		1741	492	209	0	7	3
8				>1GB	473	0	29	3
9					856	1	58	3
10					1837	1	91	3
11					2367	1	125	3
12					3830	1	156	4
13					5128	1	186	4
14					4752	1	226	4
15					4449	1	183	5
sum	71923		2202		23970		1066	

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

BDDs or SAT?

- SAT can outperform BDDs, but also BDDs may outperform SAT
- the methods seem complementary
- SAT more predictable
- BDDs memory-consuming while SAT time-consuming
- BDDs typically require more manual guidance
- BDDs complete, SAT no
- unbounded model checking UMC:

  CNF instead of BDDs in SMC
- BDDs + SAT

Progress

1981–1982: EMC — 10^5 states

1990': Spin — 10^{10} states

1990–1992: SMC — 10^{100} states

2000': SMC, BMC + CEGAR — 10^{1000} states

SAT-solvers

SAT-solvers

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

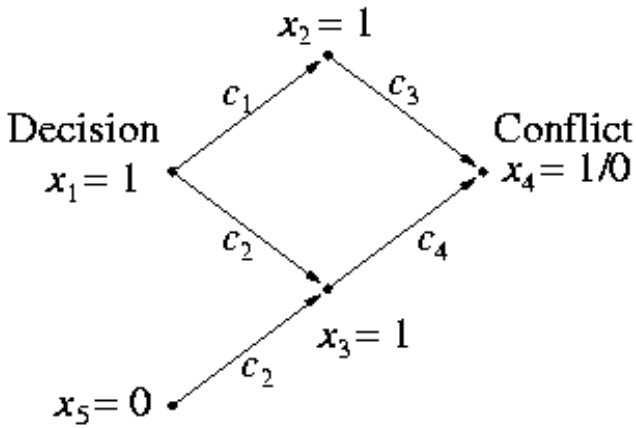
Implications graph and BCP

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

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

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

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



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

DPLL

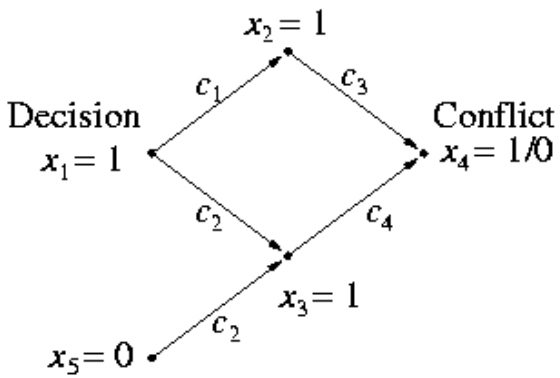
```
// Input arg: Current decision level d
// Return value:
//    SAT():      {SAT, UNSAT}
//    Decide():   {DECISION, ALL-DECIDED}
//    Deduce():   {OK, CONFLICT}
//    Diagnose():{SWAP, BACK-TRACK} also calculates β

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
- backtracking at conflict – pruning the partial valuations tree
- non-chronological backtracking
- heuristics for decisions
- efficient data structures
- incremental satisfiability

$$c_1 = (\neg x_1 \vee x_2)$$
$$c_2 = (\neg x_1 \vee x_3 \vee x_5)$$
$$c_3 = (\neg x_2 \vee x_4)$$
$$c_4 = (\neg x_3 \vee \neg x_4)$$



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