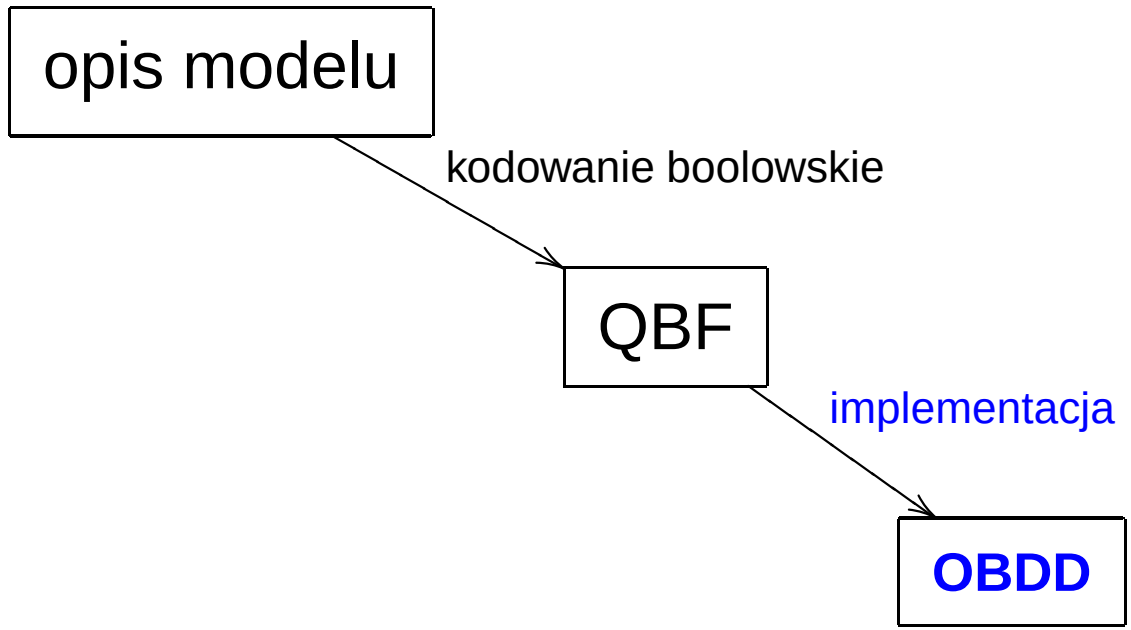
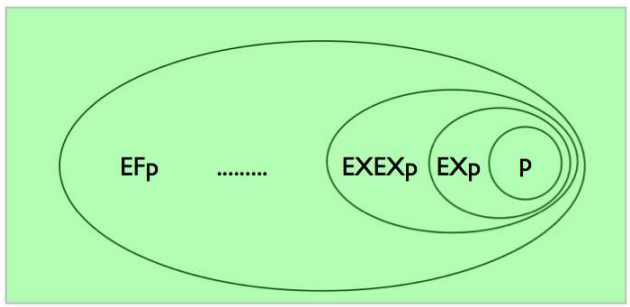
Weryfikacja wspomagana komputerowo

Wykład 8: Weryfikacja ograniczona

Symboliczna weryfikacja modelowa (SMC)





weryfikacja modelowa = operacje na OBDDs

Przykład: $\mathsf{EF}\, p$ (bezpieczeństwo)

pre-SMC:

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

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

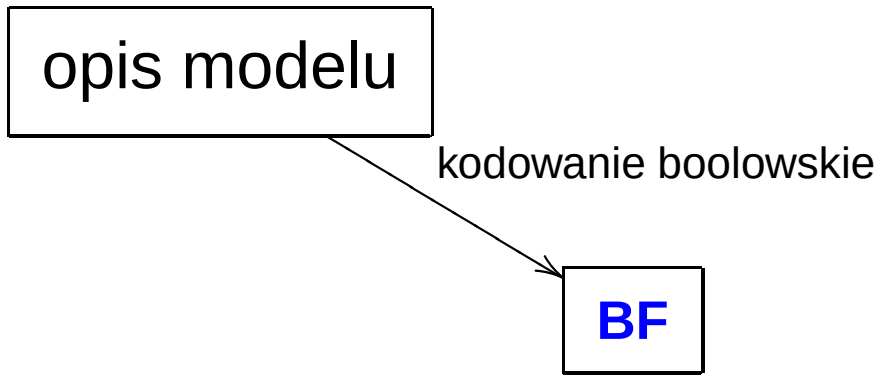
post-SMC:

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

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

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

Ograniczona weryfikacja modelowa (BMC)



weryfikacja modelowa = spełnialność formuły boolowskiej

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

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

Ograniczona weryfikacja modelowa (MBC)

- poszukiwanie kontrprzykładów ograniczonej długości
- symboliczna ścieżka
- wykorzystanie rozwiązywaczy SAT (ang. *SAT-solvers*)

I. Weryfikacja ograniczona

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

np. zamiast $M \models \mathsf{AG}\,\phi$, sprawdzamy $M \models \mathsf{EF}\,\neg\phi$

M opisany przez formuły boolowskie:

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

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

Ograniczenie

Pomysł: horyzont $k \geq 0$ k kroków systemu

$\Pi \models_k \phi$ k kroków wystarcza by stwierdzić, że $\Pi \models \phi$

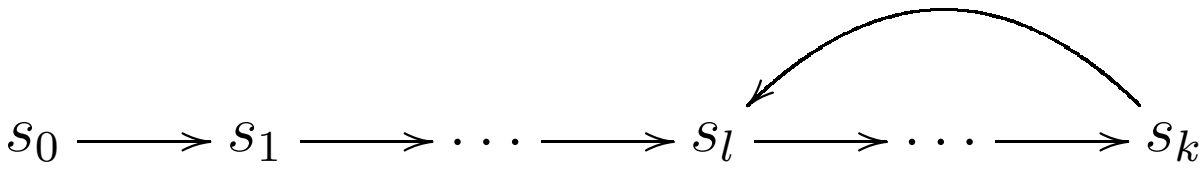
Lem.: $\Pi \models_k \phi \implies \Pi \models \phi$

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

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

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

(k, l) - pętla:



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

(k - pętla)

brak pętli:

Semantyka ograniczona dla k-pętli

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

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

Semantyka ograniczona dla nie-pętli

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

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

Semantyka ograniczona dla nie-pętli

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

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

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

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

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

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

$\Pi \models_k^i \mathsf{G}\,\phi$ nigdy

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

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

Ograniczenie

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

- **jakie k jest wystarczająco duże? średnica grafu?**
- **brak pełności !**
- **efektywność SMC zależy od średnicy grafu**

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

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

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

M opisany przez formuły boolowskie:

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

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

Kodowanie boolowskie

Ścieżka symboliczna

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

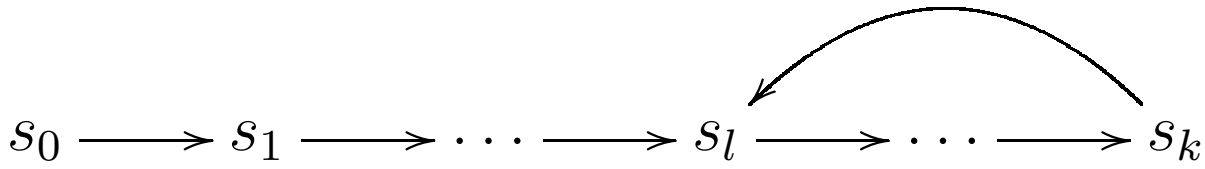
$$[M]_k \quad := \quad S_0(\vec{z}_0) \ \wedge \bigwedge_{i=0}^{k-1} R(\vec{z}_i, \vec{z}_{i+1})$$

$(k+1) \cdot m$ zmiennych boolowskich

rozmiar formuły boolowskiej $[M]_k$ jest $\mathcal{O}(k \cdot |M|)$

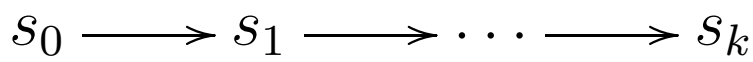
Kodowanie boolowskie

(k, l) - pętla:



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

brak pętli:



Kodowanie boolowskie

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

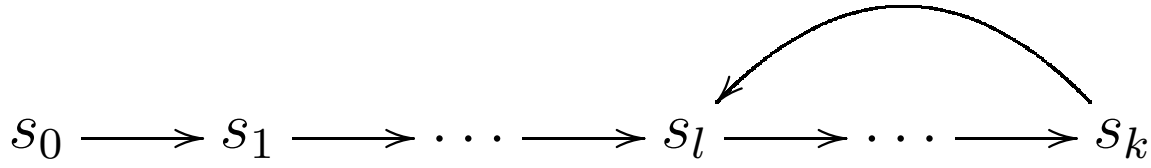
${}_l[\phi]_k^0$ – znaczenie ϕ na (k,l)-pętli

$[\phi]_k^0$ – znaczenie ϕ na nie-pętli

rozmiar formuły boolowskiej $[M,\phi]_k$ jest $\mathcal{O}(k \cdot (|M| + |\phi|))$

Kodowanie boolowskie

$_l[\phi]^i_k \qquad 0 \le l, i \le k$



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

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

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

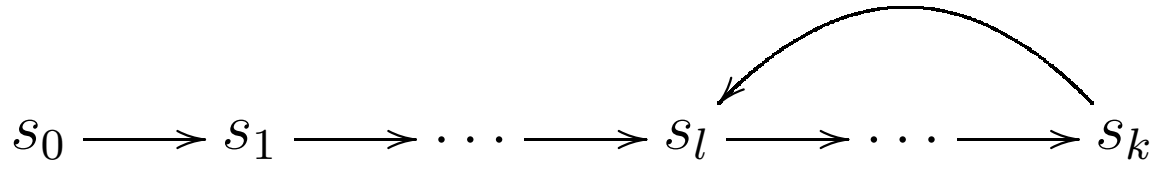
$$\mathrm{succ}(i) = \begin{cases} i+1 & i < k \\ l & i = k \end{cases}$$

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

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

Kodowanie boolowskie

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



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

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

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

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

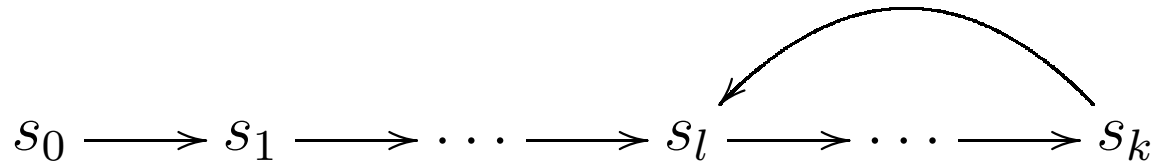
${}_l[\,\mathsf{F}\,\phi]^i_k \iff {}_l[\phi]^i_k \vee {}_l[\,\mathsf{F}\,\phi]^{\mathrm{succ}(i)}_k$

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

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

Kodowanie boolowskie

Przykład:



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

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

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

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

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

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

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

Kodowanie boolowskie

Rozmiar formuły boolowskiej ${}_l[\phi]_k^0$ jest $\mathcal{O}(k \cdot |\phi|)$

dzięki „dzieleniu podformuł” ${}_l[\phi]_k^i$

Kodowanie boolowskie

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

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

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

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

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

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

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

Kodowanie boolowskie

Przykład:

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

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

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

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

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

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

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

Tw.: $M \models_k \mathsf{E}\,\phi \iff [M,\phi]_k$ jest spełnialna.

Powtarzaj:

(1) $k := k_0$

(2) jeśli $[M,\phi]_k$ spełnialna to stop $\qquad M \models \mathsf{E}\,\phi$

(3) zwiększ k i kontynuuj

Inne kodowania boolowskie

- weryfikacji sprzętu:
  - zachowana jest informacja o strukturze układu
  - desygnowane rozwiązywacze SAT
- LTL z przeszłością
- weryfikacja oprogramowania (np. CBMC)
- programy współbieżne
- ...

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

CBMC

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

⟶

```
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;
   if(z) x++;
}
assert(x<=3);
```

→

```
x1=x0+y0;
if(x1!=1) {
   x2=2;
   if(z0) x3=x2+1;
}
assert(x3<=3);
```

→

$\mathcal{C} := (x_1 = x_0 + y_0) \wedge$
$x_2 = ((x_1 \neq 1)\,?\,2 : x_1) \wedge$
$x_3 = ((x_1 \neq 1 \wedge z_0)\,?\,x_2 + 1 : x_2)$
$\mathcal{P} := x_3 \leq 3$

[Kroening, Clarke, Yorav 2003]

II. Pełność ?

Jak uzyskać pełność weryfikacji ograniczonej?

- ograniczenie dla k (bezpieczeństwo $\mathsf{G}\, p$)
- równolegle sprawdzamy ϕ i $\neg\phi$ (żywotność $\mathsf{F}\, p$)
- indukcja (bezpieczeństwo $\mathsf{G}\, p$)

Głębokość – ograniczenie dla k

najmniejsze i t. że

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

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

Głębokość – ograniczenie dla k

najmniejsze i t. że

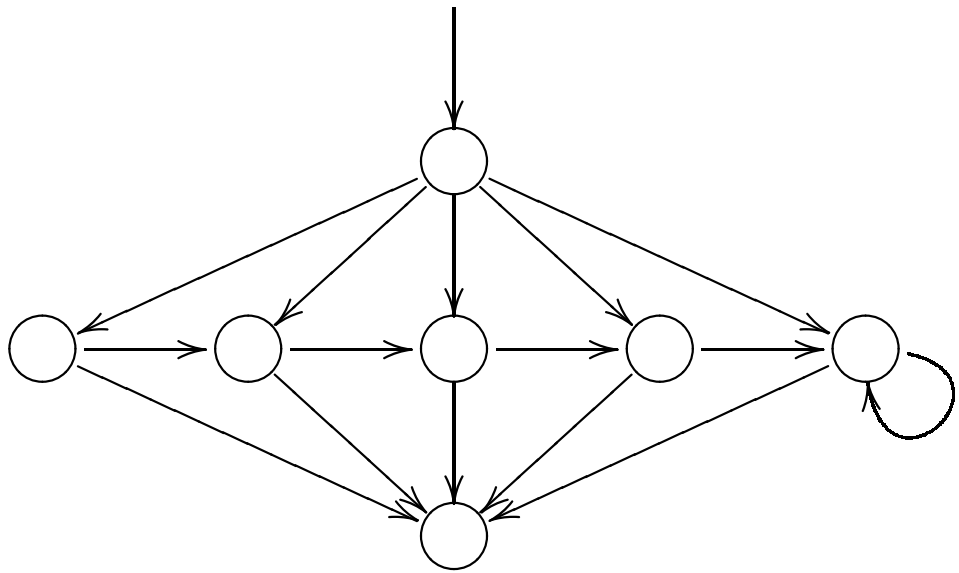
$$\forall \vec{z}_0, \ldots, \vec{z}_{i+1}.\ \exists \vec{z'}_0, \ldots, \vec{z'}_i.\ S_0(\vec{z}_0)\ \wedge\ \bigwedge_{j=0}^{i} R(\vec{z}_j, \vec{z}_{j+1})\ \Longrightarrow$$

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

najdłuższa ścieżka bez pętli – ograniczenie dla k

największe i t. że

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



Pełność dla $\mathsf{F}\,p$

$M \models \mathsf{EG}\,\neg p \iff \exists k$ t. że następująca formuła jest spełnialna:

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

$M \models \mathsf{AF}\,p \iff \exists k$ t. że następująca formuła jest tautologią:

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

$M \models \mathsf{EG}\,\neg p \iff \exists k$ t. że następująca formuła jest spełnialna:

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

$M \models \mathsf{AF}\,p \iff \exists k$ t. że następująca formuła jest niespełnialna:

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

Pełność dla $\mathsf{F}\,p$

Powtarzaj:

(1) $k := 0$

(2) jeśli No_k spełnialna to stop $\qquad M \models \mathsf{EG}\,\neg\phi$

(3) jeśli $\neg\text{Yes}_k$ niespełnialna to stop $\qquad M \models \mathsf{F}\,p$

(4) zwiększ k i kontynuuj

Pełność dla G p – indukcja

– warunki początkowe

sprawdzamy, że poniższa formuła jest niespełnialna:

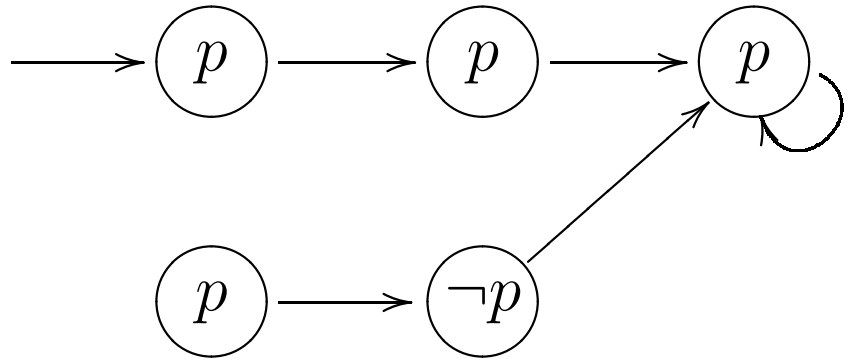
$$\mathsf{Base}_k(\vec{z}_0,\ldots,\vec{z}_k) \equiv S_0(\vec{z}_0) \wedge \bigwedge_{j=0}^{k-1} R(\vec{z}_j,\vec{z}_{j+1}) \wedge \bigvee_{j=0}^{k} \neg L_p(\vec{z}_j)$$

– krok indukcyjny

sprawdzamy, że poniższa formuła jest niespełnialna:

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

Po co $k > 1$?



Powtarzaj:

(1) $k := 0$

(2) jeśli Base$_k$ spełnialna to stop $\quad M \models \mathsf{EF}\,\neg\phi$

(3) jeśli Step$_k$ niespełnialna to stop $\quad M \models \mathsf{G}\,p$

(4) zwiększ k i kontynuuj

III. BDD czy SAT ?

Porównanie

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 czy SAT?

- metody są komplementarne
- SAT bardziej przewidywalny
- BDD zachłanne pamięciowo, a SAT czasowo
- BDD daje pełność, SAT nie
- nieograniczona weryfikacja modelowa UMC:

  CNF zamiast BDD w SMC
- BDD + SAT

Postęp

1981–1982: EMC — 10^5 stanów

1990–1992: SMV — 10^{100} stanów

2000': BMC + CEGAR — 10^{1000} stanów

IV. Rozwiązywacze SAT

Rozwiązywacze SAT

ang. *SAT-solver*

- CNF
- algorytm DPLL
  - przeszukiwanie drzewa wartościowań częściowych
  - BCP (ang. *Boolean Constraint Propagation*)
  - konflikty – „obcinanie” drzewa
- heurystyki

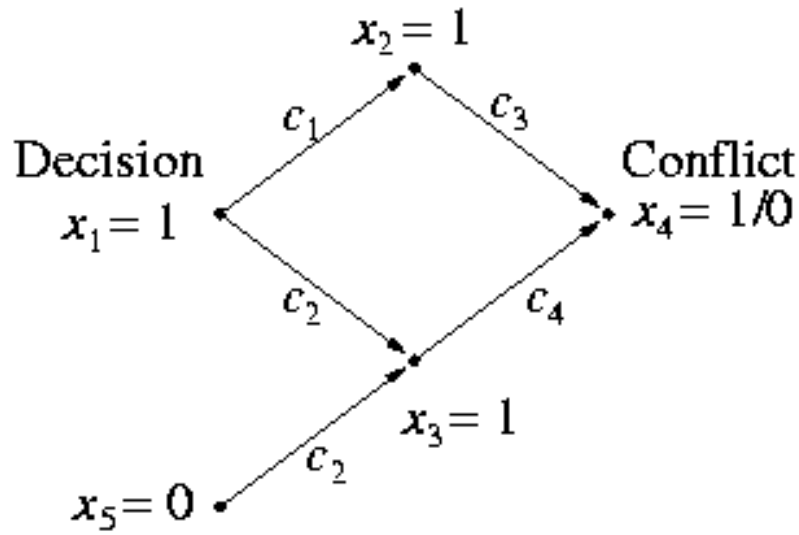
Graf implikacji i BCP

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

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

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

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



[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]

Dlaczego rozwiązywacze SAT są tak szybkie?

- uczenie konfliktów – dodajemy tzw. klauzule konfliktowe
- niechronologiczne powroty
- heurystyki dla decyzji
- szybkie struktury danych
- inkrementalna spełnialność