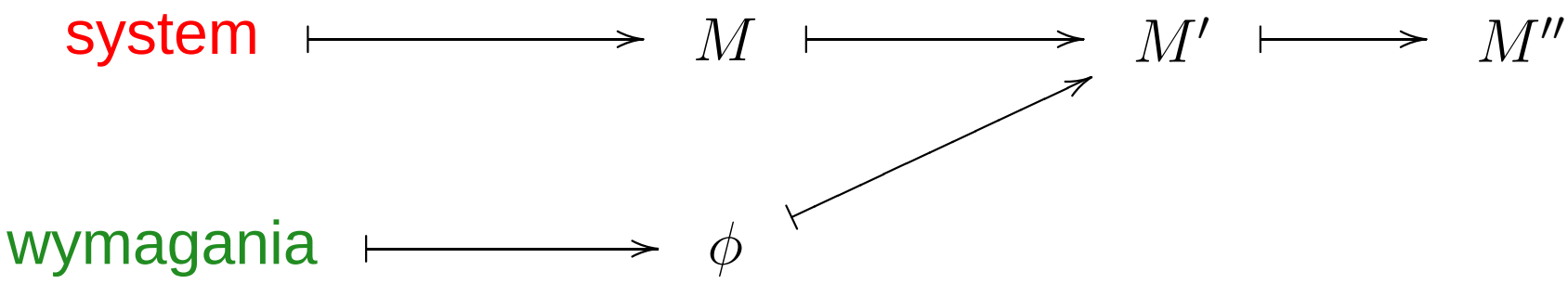
Weryfikacja wspomagana komputerowo

Wykład 9: Abstrakcja

Od rzeczywistości do modelu



$M'' \models \phi$?

Abstrakcja = usuwanie z modelu

zbędnej informacji

$$M \longmapsto M'$$

– często przydatna w przypadku układów sprzętowych

– na ogół niezbędna w przypadku oprogramowania

(nieskończone sterowanie, nieskończone dane)

Model

Sprzęt: $M = (S, S_0, L, R)$ $V = \{x_1, \ldots, x_n\}$ $S = D^n$

R, S_0, L reprezentowane przez formuły

abstrakcja dotyczy opisu modelu

Oprogramowanie: program źródłowy

I. Poprawność

Symulacja: gra $G_{M\ M'}$

– gracze: Spoiler, Duplikator

– konfiguracje: $\langle s, s' \rangle$

– konfiguracja pocz.: $\langle s_0, s'_0 \rangle$

– jeśli $L(s) \neq L(s')$, to wygrywa Spoiler

Symulacja: gra $G_{M\ M'}$

– gracze: Spoiler, Duplikator

– konfiguracje: $\langle s, s' \rangle$ – konfiguracja pocz.: $\langle s_0, s'_0 \rangle$

– jeśli $L(s) \neq L(s')$, to wygrywa Spoiler

– Spoiler wybiera tranzycję $s \longrightarrow r$

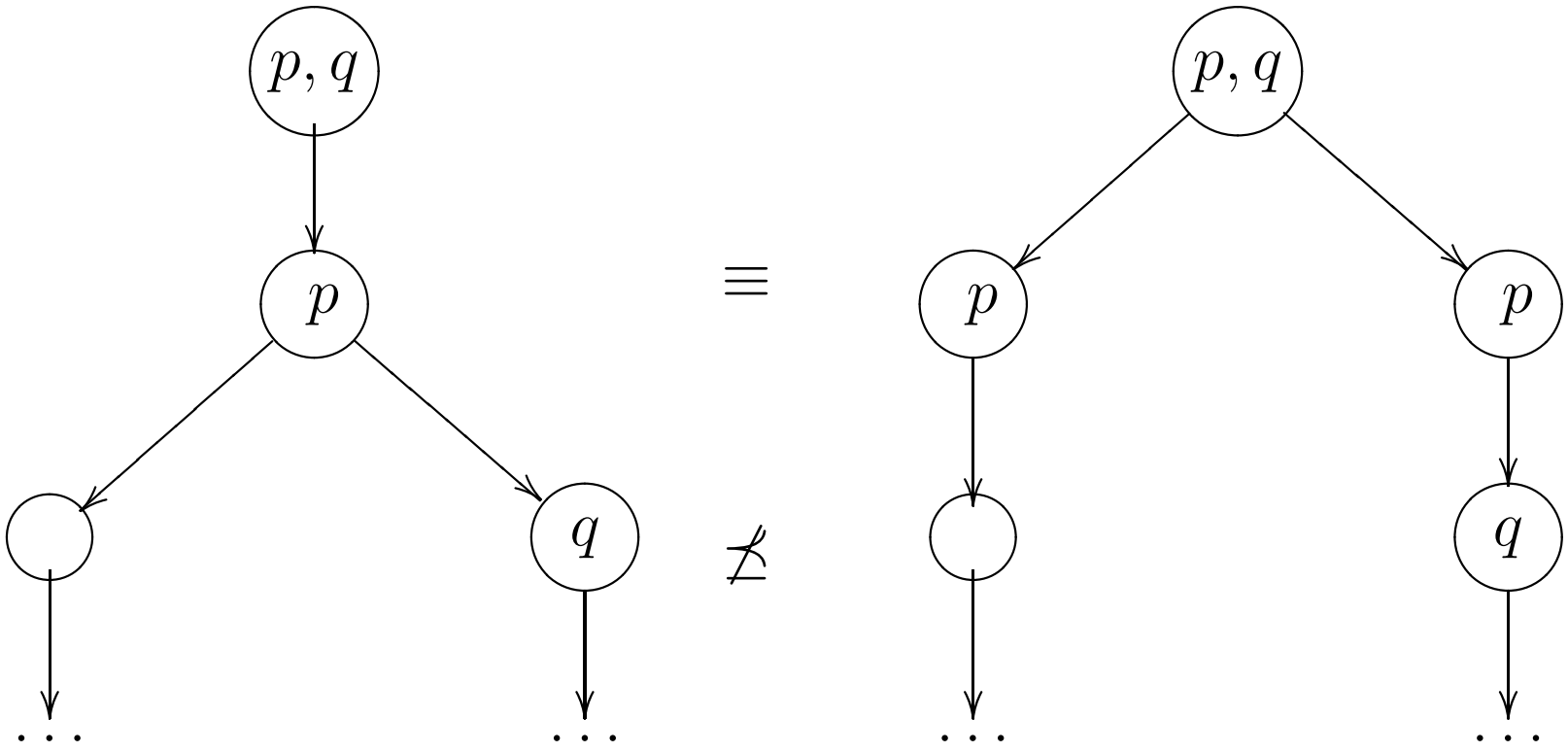
– Duplicator odpowiada po drugiej stronie ($s' \longrightarrow r'$)

– gra kontynuuje się od konfiguracji $\langle r, r' \rangle$

– kto wygrywa?

Def.: $M \preceq M' \iff$ Duplikator ma strategię wygrywającą

Przykład:



Symulacja – poprawność

Tw.: $M \preceq M' \implies (\forall \phi \in \mathrm{ACTL}^*.\ M' \models \phi \implies M \models \phi)$

Tw.: $M \preceq M' \implies (\forall \phi \in \mathrm{ECTL}^*.\ M \models \phi \implies M' \models \phi)$

Def.: Równoważność: $\simeq\ =\ \preceq \cap \succeq$

Sprawiedliwa symulacja: gra $G_{M\ M'}$

– gracze: Spoiler, Duplikator

– konfiguracje: $\langle s, s' \rangle$

– konfiguracja pocz.: $\langle s_0, s'_0 \rangle$

– jeśli $L(s) \neq L(s')$, to wygrywa Spoiler

Sprawiedliwa symulacja: gra $G_{M\ M'}$

– gracze: Spoiler, Duplikator

– konfiguracje: $\langle s, s' \rangle$ – konfiguracja pocz.: $\langle s_0, s'_0 \rangle$

– jeśli $L(s) \neq L(s')$, to wygrywa Spoiler

– Spoiler wybiera sprawiedliwą ścieżkę $s \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots$

– Duplicator odpowiada $s' \longrightarrow s'_1 \longrightarrow s'_2 \longrightarrow \ldots$

– Spoiler wybiera i

– gra kontynuuje się od konfiguracji $\langle s_i, s'_i \rangle$

– kto wygrywa?

Podsumowanie

$M \preceq M'$	$\forall \phi \in \mathrm{ACTL}^*.\ M' \models \phi \implies M \models \phi$
$M \preceq_F M'$	$\forall \phi \in \mathrm{ACTL}^*.\ M' \models_F \phi \implies M \models_F \phi$
$M \simeq M'$	$\forall \phi \in \mathrm{ACTL}^*.\ M \models \phi \iff M' \models \phi$
$M \simeq_F M'$	$\forall \phi \in \mathrm{ACTL}^*.\ M \models_F \phi \iff M' \models_F \phi$
$M \sim M'$	$\forall \phi \in \mathrm{CTL}^*.\ M \models \phi \iff M' \models \phi$
$M \sim_F M'$	$\forall \phi \in \mathrm{CTL}^*.\ M \models_F \phi \iff M' \models_F \phi$
$L_\omega(M) \subseteq L_\omega(M')$	$\forall \phi \in \mathrm{LTL}.\ M' \models \phi \implies M \models \phi$
$L_\omega(M) = L_\omega(M')$	$\forall \phi \in \mathrm{LTL}.\ M \models \phi \iff M' \models \phi$

II. Abstrakcja danych

Przykład:

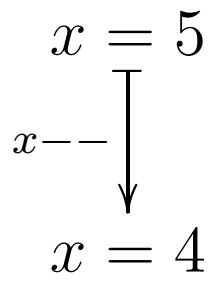
$$h(d) = \begin{cases} a_+ & x > 0 \\ a_0 & x = 0 \\ a_- & x < 0 \end{cases} \qquad h : D \to A$$

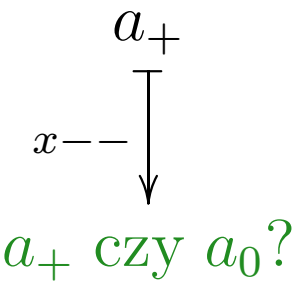
$$x = 5 \xrightarrow{x--} x = 4$$

$$a_+ \xrightarrow{x--} a_+ \text{ czy } a_0?$$

Przykład:

$$h(d) = \begin{cases} a_+ & x > 0 \\ a_0 & x = 0 \\ a_- & x < 0 \end{cases} \qquad h : D \to A$$





– model abstrakcyjny ma więcej zachowań – symulacja

– niedeterminizm

Przykładowe funkcje abstrakcji:

- znak
- logarytm
- wybrany bit
- kongruencja $\mod m$
- $h = \langle h_1, h_2 \rangle$
- predykaty ($\rightsquigarrow$ programy boolowskie)

$$h = \langle h_1, h_2, \ldots, h_n \rangle : D^n \to \{0,1\}^n$$

Predykaty – przykład

init(y) := 1;
next(y) := **case**
$(r = 1) : 0;$
$(x < y) \wedge \neg(y = 2) : y + 1;$
$(x = y) : 0;$
else : $y;$
esac

predykaty:
$(r = 1), (x < y)$

$$h(r, x, y) = \langle (r = 1), (x < y) \rangle$$

$$\langle r, x, y \rangle \approx_h \langle r', x', y' \rangle \iff (r = 1 \iff r' = 1) \wedge (x < y \iff x' < y')$$

Programy boolowskie

```
    int x, y, z, w;

    void foo()
    {
[1]   do {
[2]     z = 0;
[3]     x = y;
[4]     if (w){
[5]       x++;
[6]       z = 1;
        }
[7]   } while(x!=y)
[8]   if(z){
[9]        assert(0);
      }
    }
```

```
    decl b1, b2;
   /* b1 stands for predicate (z=0) and
      b2 stands for predicate (x=y) */
     void foo()
     begin
[1]    do
[2]      b1 := 1;
[3]      b2 := 1;
[4]      if (*)
         begin
[5]        b2 := H(0,b2);
[6]        b1 := 0;
         end
[7]    while(b2)
[8]    if (!b1)
[9]      assert(0);
     end

     boolean H(e1,e2)
     begin
[10]   if (e1) then
[11]     return(1);
[12]   elsif (e2) then
[13]     return(0);
[14]   else
[15]     return(*);
       fi
    end
```

[T. Ball, A. Podelski, S. K. Rajamani 2000]

Abstrakcja danych: model ilorazowy

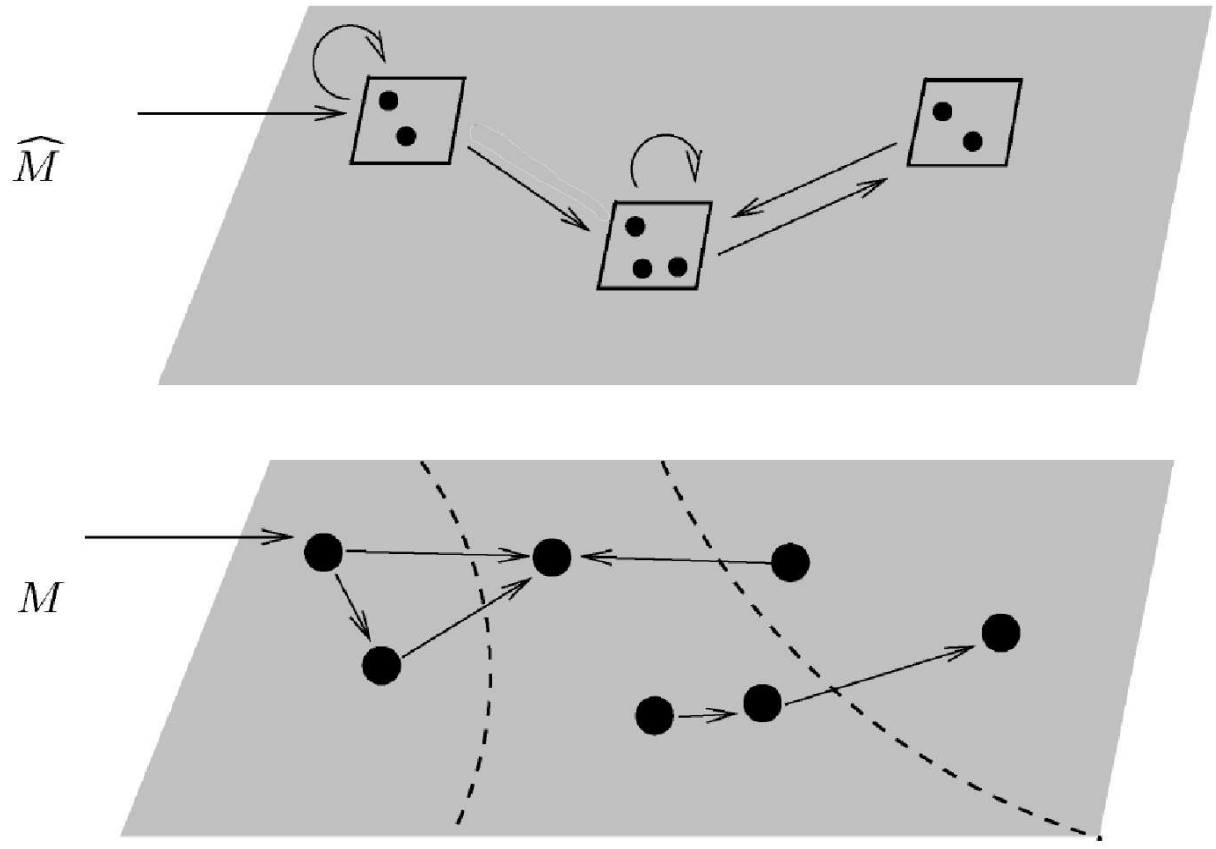
$$M \quad \mapsto \quad \widehat{M} \;=\; M/\approx_h$$

$$\approx_h \;=\; \ker(h)$$

$$R(\vec{d}_1, \vec{d}_2) \implies \widehat{R}([\vec{d}_1], [\vec{d}_2])$$

$$S_0(\vec{d}) \implies \widehat{S}_0([\vec{d}])$$

Abstrakcja egzystencjalna



[Clarke, Grumberg, Jha, Lu, Veith 2003]

Tw.: $M \preceq \widehat{M}$ (o ile $\approx_h$ zachowuje L)

III. CEGAR

Uszczegółowienie abstrakcji – CEGAR

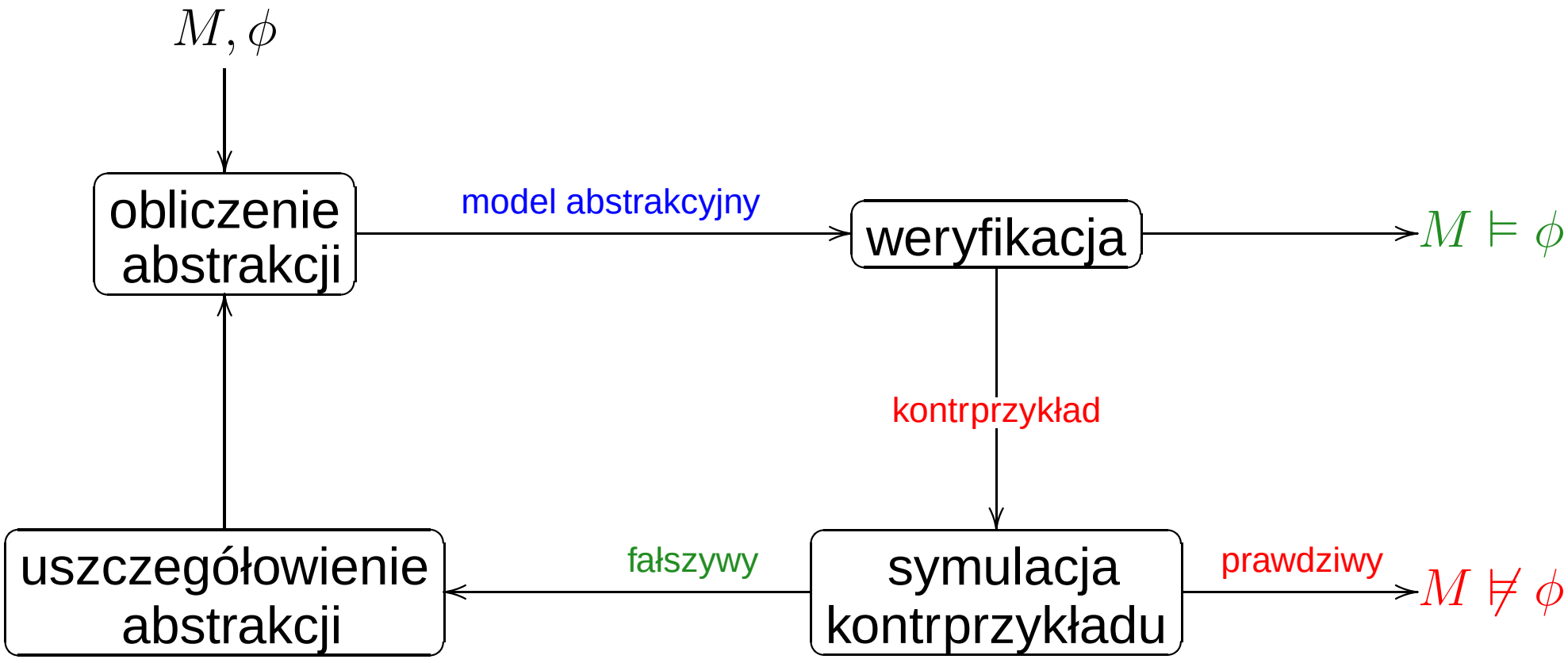
(ang. Counter-example guided abstraction refinement)

(1) Znajdź abstrakcję początkową

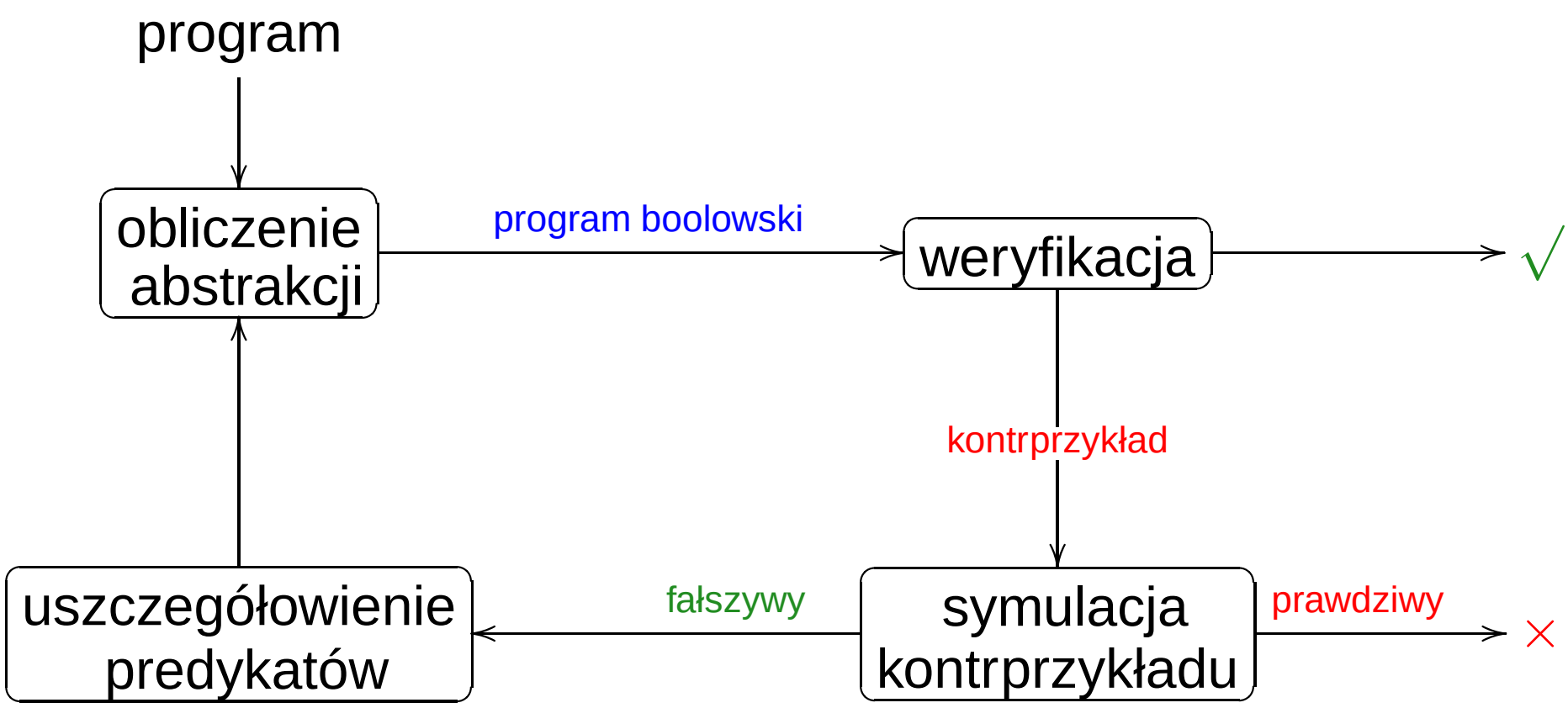
(2) Powtarzaj:

- weryfikuj model abstrakcyjny
  - jeśli wynik pozytywny – zakończ
  - jeśli wynik negatywny – kontrprzykład abstrakcyjny
- jesli kontrprzykład jest rzeczywisty – zakończ
- uszczegółów abstrakcję na podstawie kontrprzykładu

CEGAR


M, ϕ
obliczenie abstrakcji
model abstrakcyjny
weryfikacja
$M \models \phi$
kontrprzykład
uszczegółowienie abstrakcji
fałszywy
symulacja kontrprzykładu
prawdziwy
$M \not\models \phi$

CEGAR


program
obliczenie abstrakcji
program boolowski
weryfikacja
√
kontrprzykład
uszczegółowienie predykatów
fałszywy
symulacja kontrprzykładu
prawdziwy
×

Abstrakcja początkowa

$s \equiv_h s'$ jeśli

– $L(s) = L(s')$

– s i s' nie są rozróżniane przez dozory

Abstrakcja początkowa

init(y) := 1;
next(y) := **case**
$(r = 1) : 0;$
$(x < y) \wedge \neg(y = 2) : y + 1;$
$(x = y) : 0;$
else : $y;$
esac

predykaty:
$(r = 1)$, $(x = y)$, $(x < y)$,
$(y = 2)$

$$h(r, x, y) = \langle (r = 1), (x = y), (x < y), (y = 2) \rangle$$

$$\langle r, x, y \rangle \approx_h \langle r', x', y' \rangle \iff \begin{array}{l} (r = 1 \iff r' = 1) \wedge \\ (x = y \iff x' = y') \wedge \\ (x < y \iff x' < y') \wedge \\ (y = 2 \iff y' = 2) \end{array}$$

Abstrakcja początkowa

```
   numUnits: int;
   level: int;
   void getUnit() {
[1]  canEnter: bool := F;
[2]  if (numUnits = 0) {
[3]    if (level > 10){
[4]      NewUnit();
[5]      numUnits := 1;
[6]      canEnter := T;
       }
     } else
[7]    canEnter := T;

[8]  if (canEnter)
[9]    if (numUnits = 0)
[10]     assert(F);
       else
[11]     gotUnit();
   }
```

P

```
   void getUnit() {
[1]  ...;
[2]  if (?) {
[3]    if (?){
[4]      ...;
[5]      ...;
[6]      ...;
       }
     } else
[7]    ...;

[8]  if (?)
[9]    if (?)
[10]     ...;
       else
[11]     ...;
   }
```

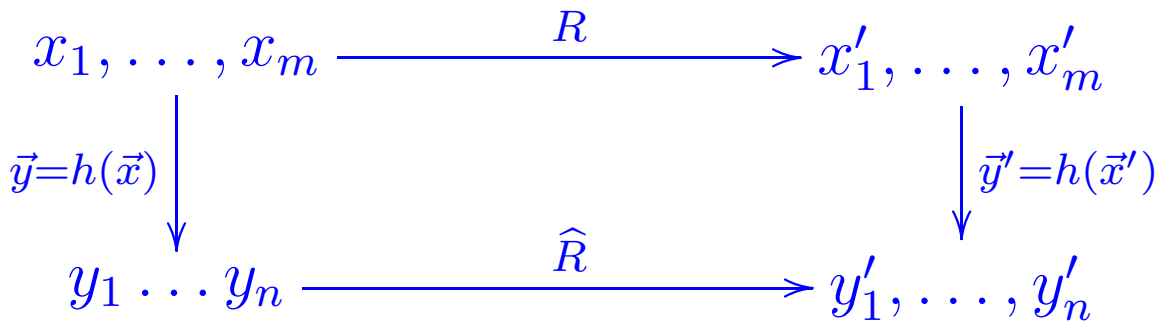
B_1

[T. Ball, S. K. Rajamani 2000]

Obliczanie abstrakcji

$$h : D^m \to A^n$$

- $\widehat{S} = A^n$
- $\widehat{R}(\vec{y}, \vec{y}') \equiv \exists \vec{x}, \vec{x}'.\ R(\vec{x}, \vec{x}') \wedge \vec{y}{=}h(\vec{x}) \wedge \vec{y}'{=}h(\vec{x}')$



- $\widehat{S}_0(\vec{y}) \equiv \exists \vec{x}.\ S_0(\vec{x}) \wedge \vec{y}{=}h(\vec{x})$

$[\phi](\vec{y}) \equiv \exists \vec{x}.\ \phi(\vec{x}) \wedge \vec{y}{=}h(\vec{x})$

Obliczanie abstrakcji – przybliżenie

Obliczamy przybliżenie $[\phi]$

- $\mathcal{A}(x' = x-1) \;=\; (y = a_+ \wedge (y' = a_+ \vee y' = a_0)) \vee \ldots$
- $\mathcal{A}(\phi_1 \wedge \phi_2) \;=\; \mathcal{A}(\phi_1) \wedge \mathcal{A}(\phi_2)$
- $\ldots$

Fakt: $[\phi] \implies \mathcal{A}(\phi)$

Tw.: $M \preceq \widehat{M} \preceq M^{\mathcal{A}}$

Programy boolowskie – obliczanie abstrakcji

```
   numUnits: int;
   level: int;
   void getUnit() {
[1]  canEnter: bool := F;
[2]  if (numUnits = 0) {
[3]    if (level > 10){
[4]      NewUnit();
[5]      numUnits := 1;
[6]      canEnter := T;
       }
     } else
[7]    canEnter := T;

[8]  if (canEnter)
[9]    if (numUnits = 0)
[10]     assert(F);
       else
[11]     gotUnit();
   }
```

P

```
   void getUnit() {
[1]  ...;
[2]  if (?) {
[3]    if (?){
[4]      ...;
[5]      ...;
[6]      ...;
       }
     } else
[7]    ...;

[8]  if (?)
[9]    if (?)
[10]     ...;
       else
[11]     ...;
   }
```

B_1

```
   nU0: bool;

   void getUnit() {
[1]  ...;
[2]  if (nU0) {
[3]    if (?){
[4]      ...;
[5]      nU0:=F;
[6]      ...;
       }
     } else
[7]    ...;

[8]  if (?){
[9]    if (nU0)
[10]     ...;
       else
[11]     ...;
   }
```

B_2

```
   nU0: bool;

   void getUnit() {
[1]  cE: bool :=F;
[2]  if (nU0) {
[3]    if (?){
[4]      ...;
[5]      nU0:=F;
[6]      cE:=T;
       }
     } else
[7]    cE:=T;

[8]  if (cE)
[9]    if (nU0)
[10]     ...;
       else
[11]     ...;
   }
```

B_3

[T. Ball, S. K. Rajamani 2000]

– system dowodzący (rozwiązywacz SMT)

– rozwiązywacz SAT

Obliczanie abstrakcji – przybliżenie

$\mathtt{x} := \mathtt{y}$

predykaty: $x < 5,\ x > 5,\ y = 5$

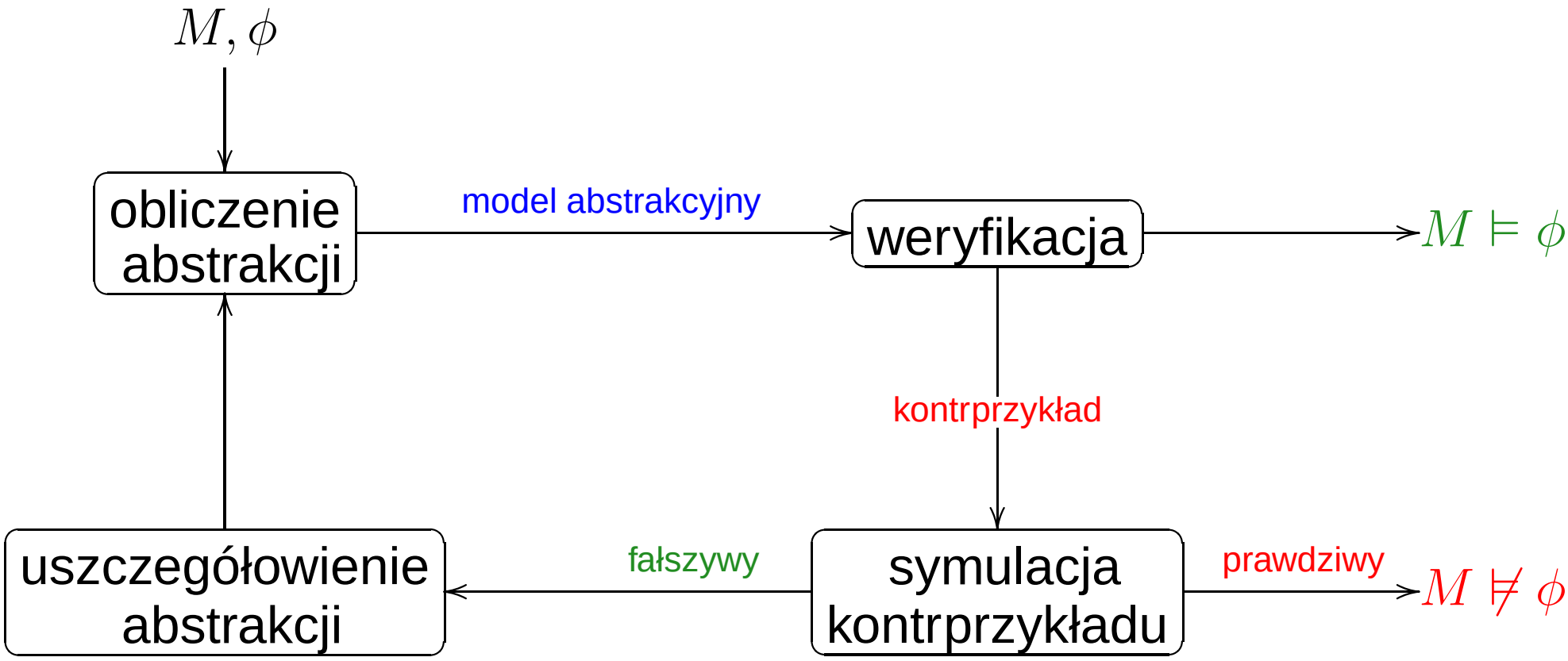
$$(x \geq 5,\ x \leq 5,\ y \neq 5) \ \mapsto\ (x < 5,\ x \leq 5,\ y \neq 5) \vee (x \geq 5,\ x > 5,\ y \neq 5$$

przybliżenie kartezjańskie:

$$2^{A_1 \times A_2} \ \mapsto\ 2^{A_1} \times 2^{A_2}$$

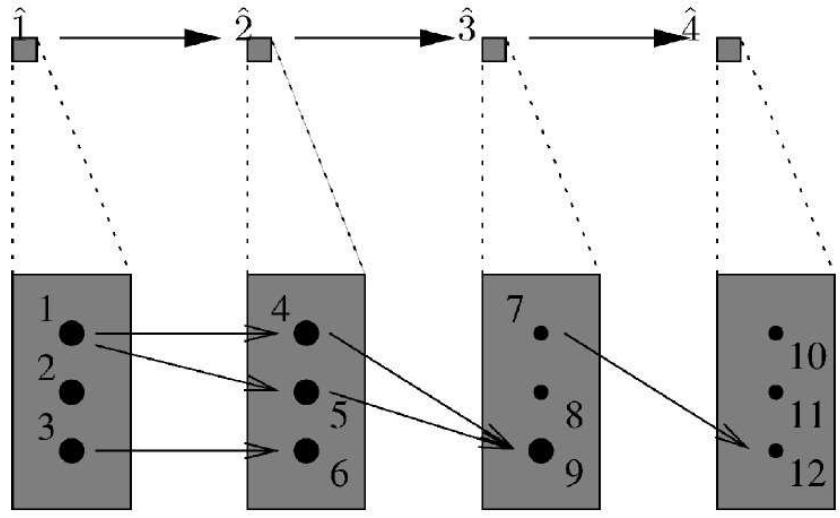
$$(x \geq 5,\ x \leq 5,\ y \neq 5) \ \mapsto\ (x < 5 \vee x \geq 5,\ x \leq 5 \vee x > 5,\ y \neq 5)$$

CEGAR: fałszywy kontrprzykład



CEGAR: fałszywy kontrprzykład

kontrprzykład abstrakcyjny $\widehat{1}\,\widehat{2}\,\widehat{3}\,\widehat{4}$



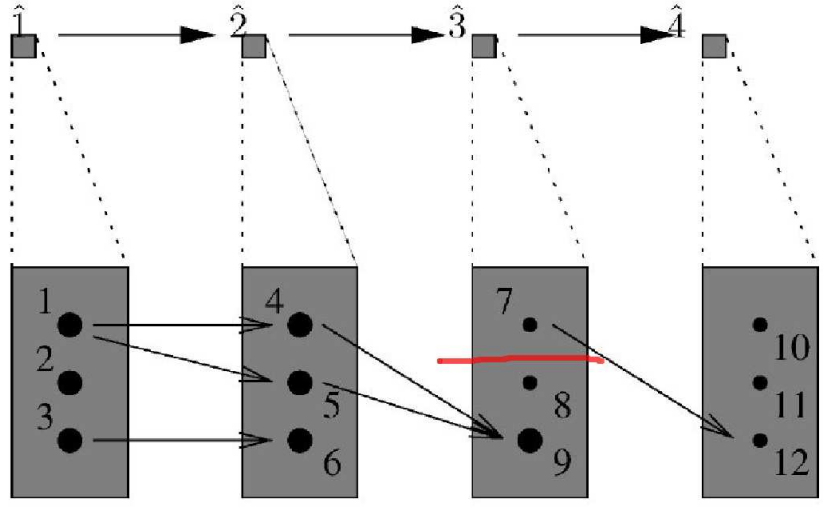
[Clarke, Grumberg, Jha, Lu, Veith 2003]

symulacja kontrprzykładu (najsilniejszy warunek końcowy)

$$S_1 := S_0 \cap h^{-1}(\,\widehat{1}\,) \qquad S_i := \vec{R}(S_{i-1}) \cap h^{-1}(\,\widehat{i}\,)$$

CEGAR: uszczegółowienie abstrakcji

kontrprzykład abstrakcyjny $\widehat{1}\,\widehat{2}\,\widehat{3}\,\widehat{4}$



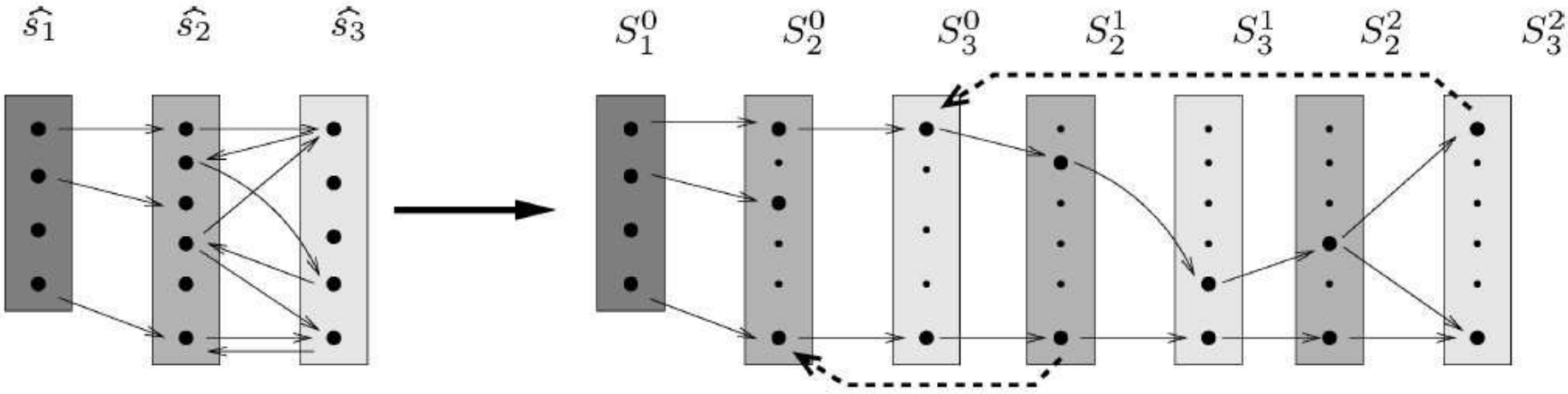
[Clarke, Grumberg, Jha, Lu, Veith 2003]

$$S_3 \cap \vec{R}^{-1}(h^{-1}(\widehat{4})) = \emptyset$$

cel: wyeliminowanie kontrprzykładu

CEGAR: pętla

kontrprzykład abstrakcyjny $\widehat{s_1}(\widehat{s_2}\widehat{s_3})^\omega$



[Clarke, Grumberg, Jha, Lu, Veith 2003]

– rozwijamy pętlę $\min\{\text{rozmiar}(\widehat{s_2}), \text{rozmiar}(\widehat{s_3})\}$ razy

Uszczegółowienie abstr. – nowe predykaty

```
   numUnits: int;
   level: int;
   void getUnit() {
[1]  canEnter: bool := F;
[2]  if (numUnits = 0) {
[3]    if (level > 10){
[4]      NewUnit();
[5]      numUnits := 1;
[6]      canEnter := T;
       }
     } else
[7]    canEnter := T;

[8]  if (canEnter)
[9]    if (numUnits = 0)
[10]     assert(F);
       else
[11]     gotUnit();
   }
```

P

```
   void getUnit() {
[1]  ...;
[2]  if (?) {
[3]    if (?){
[4]      ...;
[5]      ...;
[6]      ...;
       }
     } else
[7]    ...;

[8]  if (?)
[9]    if (?)
[10]     ...;
       else
[11]     ...;
   }
```

B_1

```
   nU0: bool;

   void getUnit() {
[1]  ...;
[2]  if (nU0) {
[3]    if (?){
[4]      ...;
[5]      nU0:=F;
[6]      ...;
       }
     } else
[7]    ...;

[8]  if (?){
[9]    if (nU0)
[10]     ...;
       else
[11]     ...;
   }
```

B_2

```
   nU0: bool;

   void getUnit() {
[1]  cE: bool :=F;
[2]  if (nU0) {
[3]    if (?){
[4]      ...;
[5]      nU0:=F;
[6]      cE:=T;
       }
     } else
[7]    cE:=T;

[8]  if (cE)
[9]    if (nU0)
[10]     ...;
       else
[11]     ...;
   }
```

B_3

[T. Ball, S. K. Rajamani 2000]

– system dowodzący (rozwiązywacz SMT)

– rozwiązywacz SAT

Uszczegółowienie abstr. – nowe predykaty

– interpolacja: jeśli $A \wedge B$ niespełnialna, to istnieje A' t.że:
  – A implikuje A', $A' \wedge B$ niespełnialna
  – zmienne$(A') \subseteq$ zmienne$(A) \cap$ zmienne(B)

– abstrakcja z interpolacji

	kontrprzykład		interpolanty
1:	$x := ctr;$	$\langle x,1\rangle = \langle ctr,0\rangle$	$x = ctr$
2:	$ctr := ctr + 1;$	$\langle ctr,1\rangle = \langle ctr,0\rangle + 1$	$x = ctr - 1$
3:	$y := ctr;$	$\langle y,2\rangle = \langle ctr,1\rangle$	$x = y - 1$
4:	$\mathtt{assume}(x = m);$	$\langle x,1\rangle = \langle m,0\rangle$	$y = m + 1$
5:	$\mathtt{assume}(y \neq m + 1);$	$\langle y,2\rangle = \langle m,0\rangle + 1$	

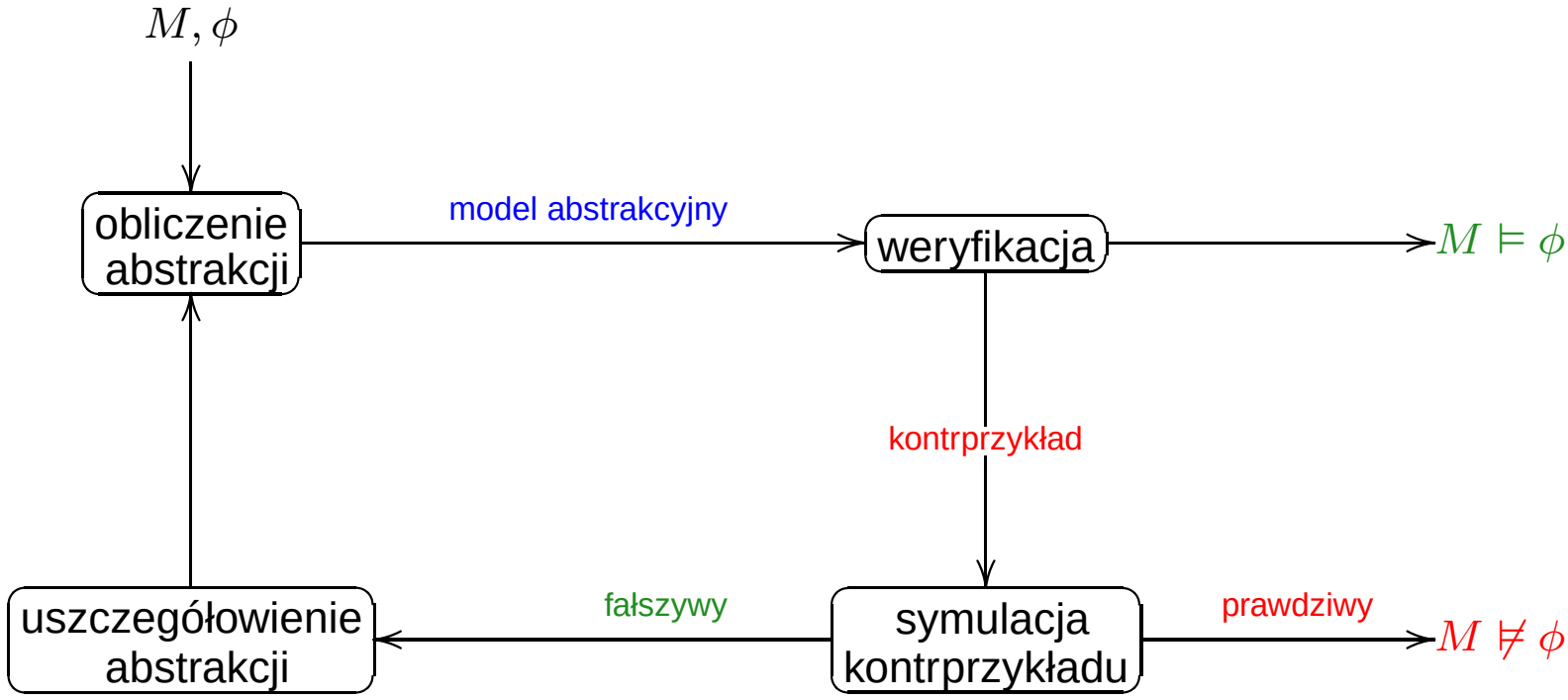
[T. A. Henzinger, R. Jhala, R. Majumdar, K. L. McMillan 2004]

CEGAR: BDD + SAT

– BDD: weryfikacja

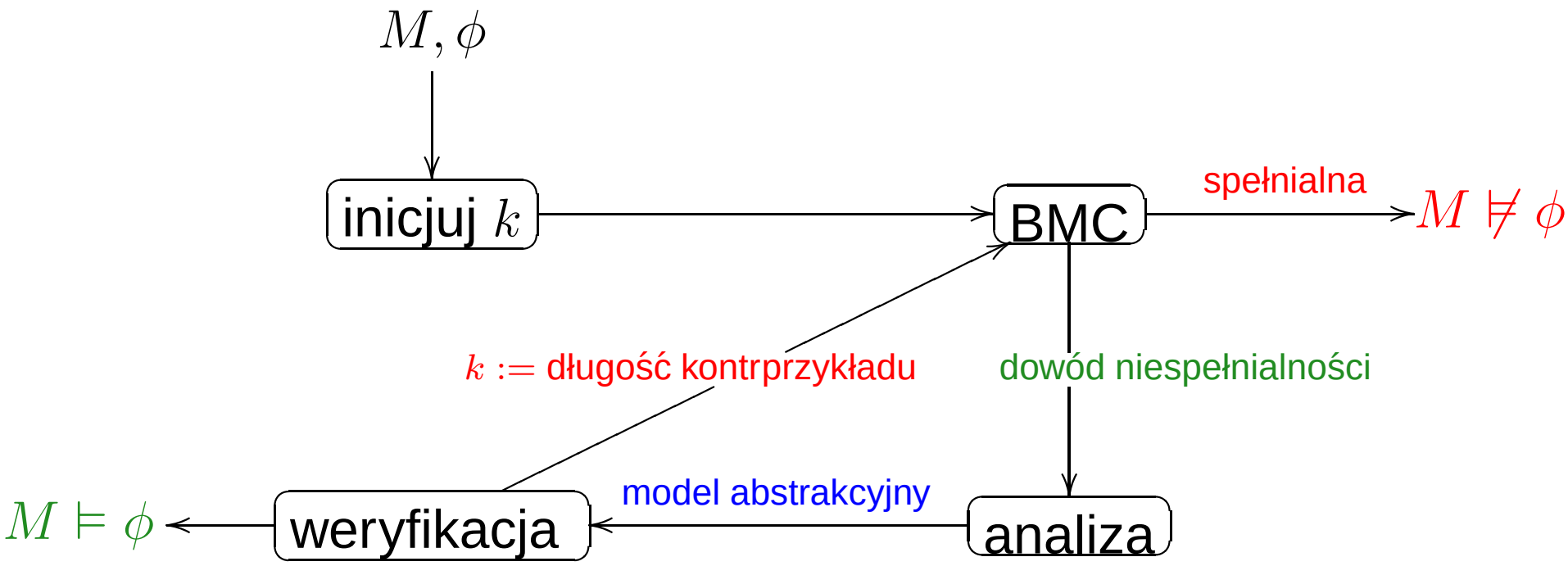
– rozwiązywacz SAT: symulacja kontrprzykładu,

obliczenie/uszczegółowienie abstrakcji



Odmiana: BDD + SAT

– abstrakcja z dowodu niespełnialności



– wyeliminowanie wszystkich kontrprzykładów długości $\leq k$