Modele abstrakcyjne w weryfikacji

Krzysztof Nozderko
kn201076@students.mimuw.edu.pl

16 maj 2006

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Gra w bisymulację

- jest dwóch graczy
- gracz Odróżniający – chce udowodnić różność modeli
- gracz Broniący – chce obronić tezę o równoważności modeli
- jeśli graczowi **Odróżniającemu** uda się odróżnić modele w skończonej liczbie kroków – wygrywa, modele nie są w bisymulacji
- wpp. wygrywa gracz **Broniący** – modele są w bisymulacji

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Gra w bisymulację

- jest dwóch graczy
- gracz Odróżniający – chce udowodnić różność modeli
- gracz Broniący – chce obronić tezę o równoważności modeli
- jeśli graczowi **Odróżniającemu** uda się odróżnić modele w skończonej liczbie kroków – wygrywa, modele nie są w bisymulacji
- wpp. wygrywa gracz **Broniący** – modele są w bisymulacji

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Gra w bisymulację

- jest dwóch graczy
- gracz Odróżniający – chce udowodnić różność modeli
- gracz Broniący – chce obronić tezę o równoważności modeli
- jeśli graczowi **Odróżniającemu** uda się odróżnić modele w skończonej liczbie kroków – wygrywa, modele nie są w bisymulacji
- wpp. wygrywa gracz **Broniący** – modele są w bisymulacji

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Gra w bisymulację

- jest dwóch graczy
- gracz Odróżniający – chce udowodnić różność modeli
- gracz Broniący – chce obronić tezę o równoważności modeli
- jeśli graczowi **Odróżniającemu** uda się odróżnić modele w skończonej liczbie kroków – wygrywa, modele nie są w bisymulacji
- wpp. wygrywa gracz **Broniący** – modele są w bisymulacji

Bisymualcja jako gra

Weźmy dwa modele. Żeby rozstrzygnąć, czy są one z punktu widzenia obserwatora nierozróżnialne, zagramy w pewną grę.

Gra w bisymulację

- jest dwóch graczy
- gracz Odróżniający – chce udowodnić różność modeli
- gracz Broniący – chce obronić tezę o równoważności modeli
- jeśli graczowi **Odróżniającemu** uda się odróżnić modele w skończonej liczbie kroków – wygrywa, modele nie są w bisymulacji
- wpp. wygrywa gracz **Broniący** – modele są w bisymulacji

Gra w bisymulację

Zasady gry

W pętli:

- gracz **Odróżniający** wybiera sobie 1 z modeli i wykonuje w nim jakąś akcję
- gracz **Broniący** w drugim z modeli wykonuje akcję o takiej samej etykiecie – jeśli nie potrafi to przegrywa

Gra w bisymulację

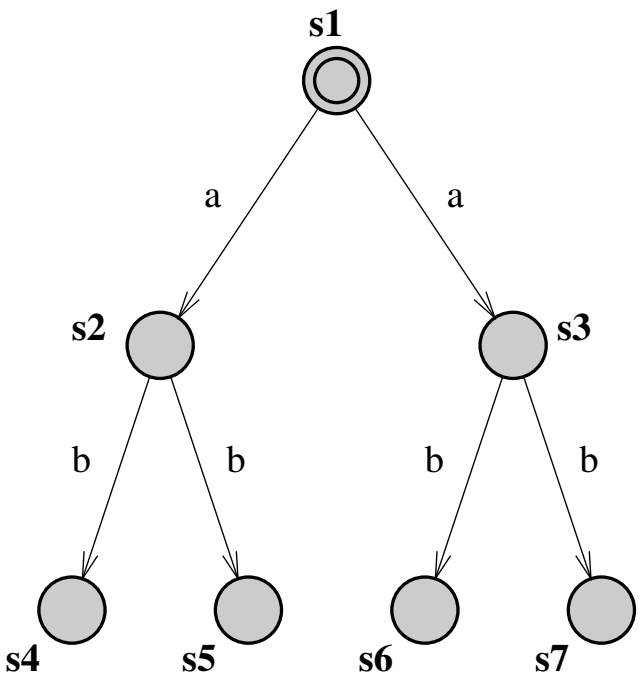
Zasady gry

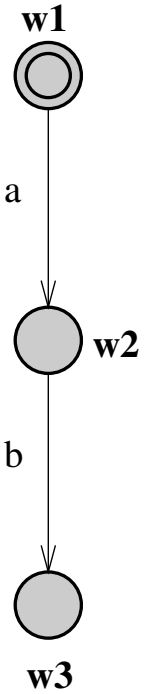
W pętli:

- gracz **Odróżniający** wybiera sobie 1 z modeli i wykonuje w nim jakąś akcję
- gracz **Broniący** w drugim z modeli wykonuje akcję o takiej samej etykiecie – jeśli nie potrafi to przegrywa

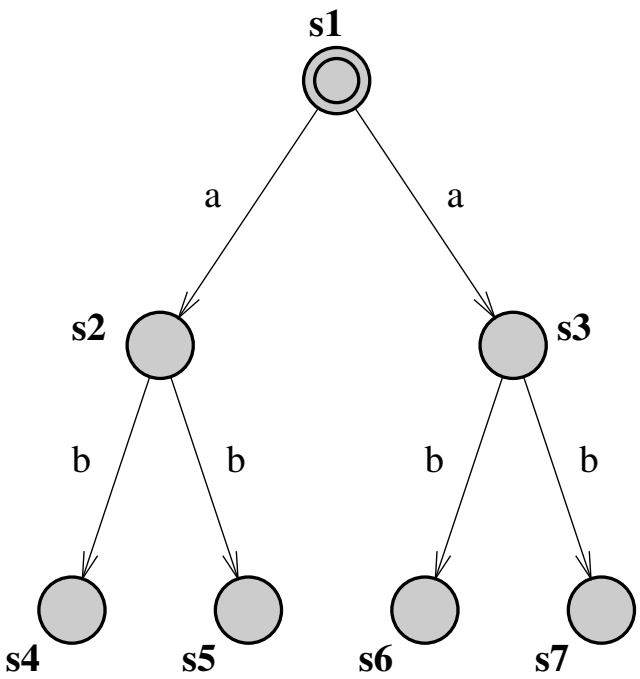
Uwaga: W kolejnych turach gracz **Odróżniający** może wybierać różne modele.

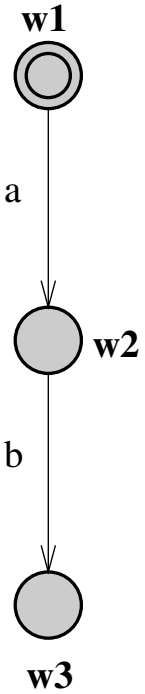
Bisymulacja – przykład





Bisymulacja – przykład

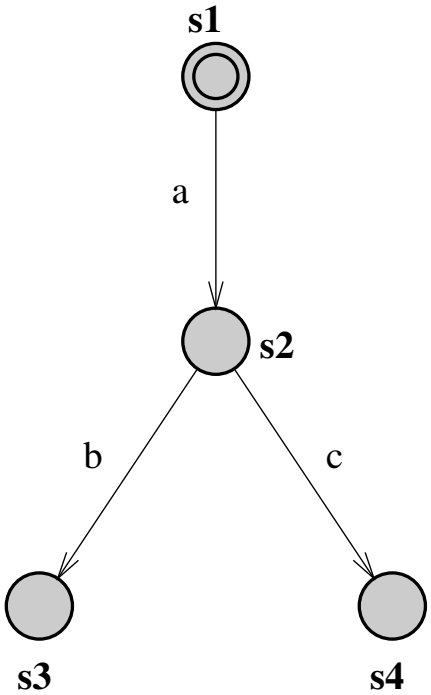


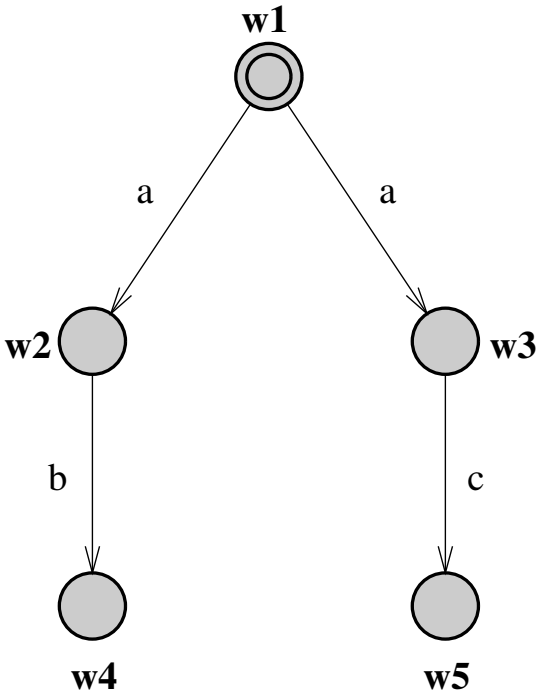


Bisymulacja:

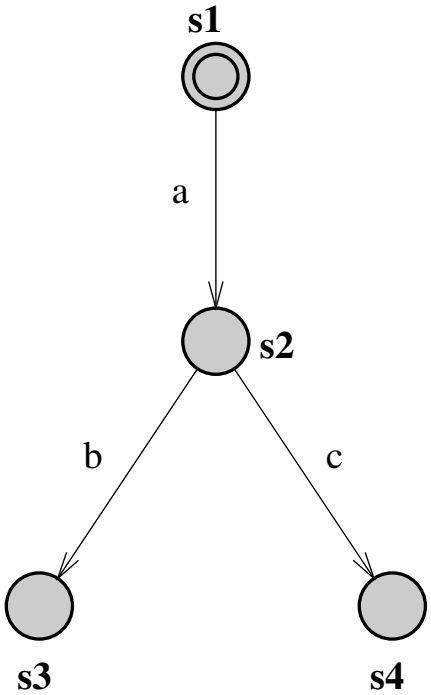
$\{(s_1, w_1),$
$(s_2, w_2), (s_3, w_2),$
$(s_4, w_3), (s_5, w_3), (s_6, w_3), (s_7, w_3)\}$

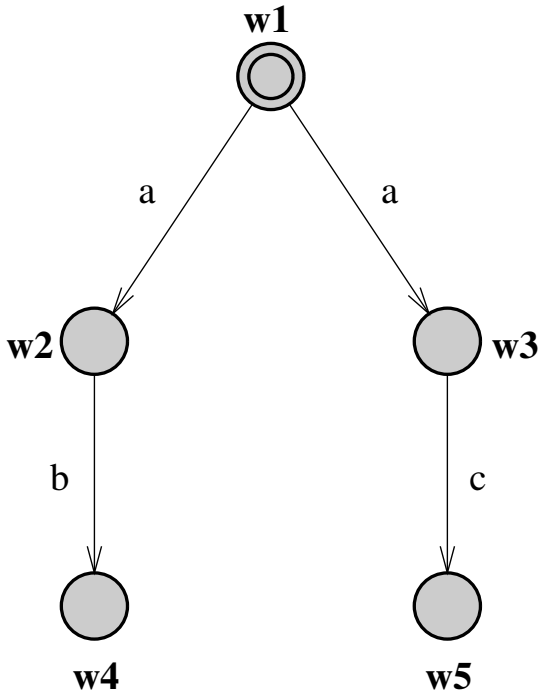
Bisymulacja – przykład





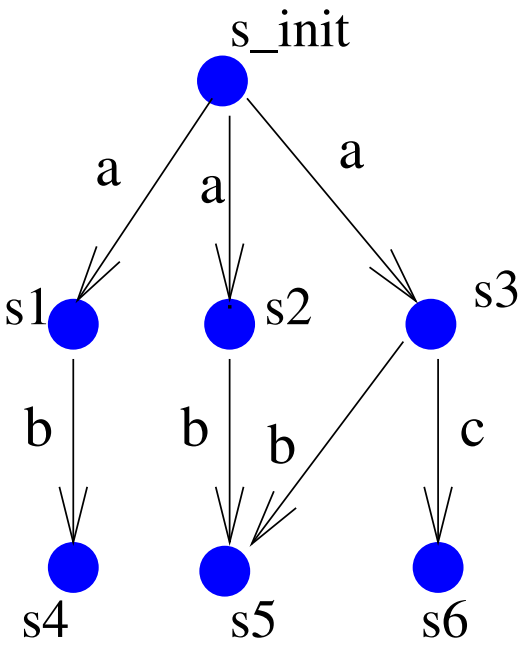
Bisymulacja – przykład

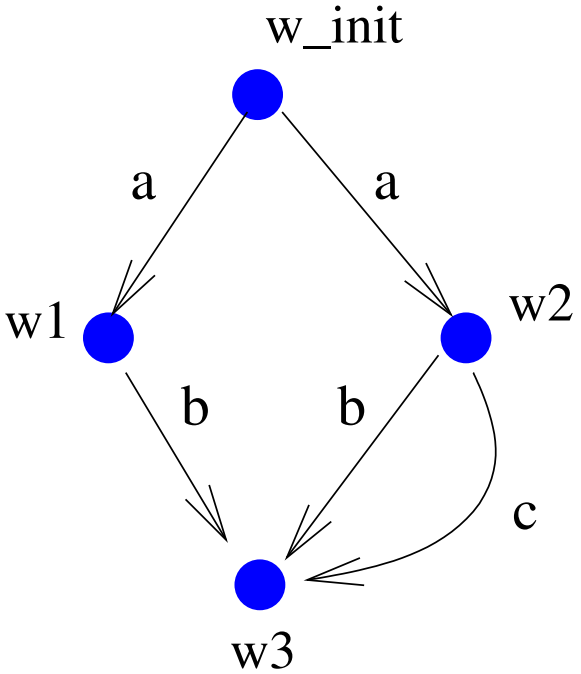




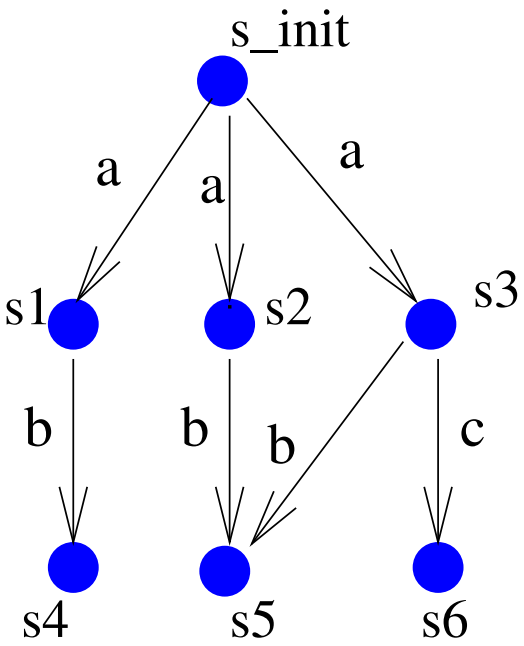
Nie ma bisymulacji.

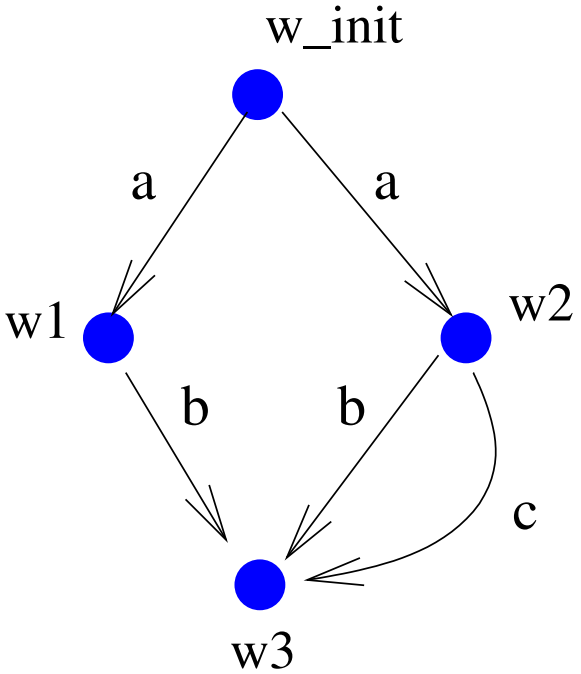
Bisymulacja – przykład





Bisymulacja – przykład





Bisymulacja:

$\{(s_{init}, w_{init}),$
$(s_1, w_1), (s_2, w_1), (s_3, w_2)$
$(s_4, w_3), (s_5, w_3), (s_6, w_3)\}$

Bisymulacja – trochę głębiej

- strategia wygrywająca gracza **Broniącego** to relacja bisymulacji
- jeśli by zabronić graczowi **Odróżniającemu** zmienianie stron, mielibyśmy symulację
- jeśli system **A** symuluje **B** oraz **B** symuluje **A**, to systemy są symulacyjnie równoważne
- jeśli
  - **A** i **B** są symulacyjnie równoważne
  - **A** symuluje **B** przy pomocy relacji R
  - **B** symuluje **A** przy pomocy relacji R^{-1},

  to systemy są w bisymulacji

Bisymulacja – trochę głębiej

- strategia wygrywająca gracza **Broniącego** to relacja bisymulacji
- jeśli by zabronić graczowi **Odróżniającemu** zmienianie stron, mielibyśmy symulację
- jeśli system **A** symuluje **B** oraz **B** symuluje **A**, to systemy są symulacyjnie równoważne
- jeśli
  - **A** i **B** są symulacyjnie równoważne
  - **A** symuluje **B** przy pomocy relacji R
  - **B** symuluje **A** przy pomocy relacji R^{-1},

  to systemy są w bisymulacji

Bisymulacja – trochę głębiej

- strategia wygrywająca gracza **Broniącego** to relacja bisymulacji
- jeśli by zabronić graczowi **Odróżniającemu** zmienianie stron, mielibyśmy symulację
- jeśli system **A** symuluje **B** oraz **B** symuluje **A**, to systemy są symulacyjnie równoważne
- jeśli
  - **A** i **B** są symulacyjnie równoważne
  - **A** symuluje **B** przy pomocy relacji R
  - **B** symuluje **A** przy pomocy relacji R^{-1},

  to systemy są w bisymulacji

Bisymulacja – trochę głębiej

- strategia wygrywająca gracza **Broniącego** to relacja bisymulacji
- jeśli by zabronić graczowi **Odróżniającemu** zmienianie stron, mielibyśmy symulację
- jeśli system **A** symuluje **B** oraz **B** symuluje **A**, to systemy są symulacyjnie równoważne
- jeśli
  - **A** i **B** są symulacyjnie równoważne
  - **A** symuluje **B** przy pomocy relacji R
  - **B** symuluje **A** przy pomocy relacji R^{-1},

  to systemy są w bisymulacji

Weryfikacja

- chcemy weryfikować różne własności systemów, np.
  - gdy szlaban jest podniesiony, na torach nie ma pociągu
  - w sekcji krytycznej przebywa co najwyżej 1 proces
  - zawsze któryś z procesów ma aktualne dane
- budujemy jakiś model systemu (automat)
- często system modelujemy jako sieć automatów
- weryfikowaną formułę rozbijamy na własności atomowe
- stany wzbogacamy o wartościowanie własności atomowych

Weryfikacja

- chcemy weryfikować różne własności systemów, np.
  - gdy szlaban jest podniesiony, na torach nie ma pociągu
  - w sekcji krytycznej przebywa co najwyżej 1 proces
  - zawsze któryś z procesów ma aktualne dane
- budujemy jakiś model systemu (automat)
- często system modelujemy jako sieć automatów
- weryfikowaną formułę rozbijamy na własności atomowe
- stany wzbogacamy o wartościowanie własności atomowych

Weryfikacja

- chcemy weryfikować różne własności systemów, np.
  - gdy szlaban jest podniesiony, na torach nie ma pociągu
  - w sekcji krytycznej przebywa co najwyżej 1 proces
  - zawsze któryś z procesów ma aktualne dane
- budujemy jakiś model systemu (automat)
- często system modelujemy jako sieć automatów
- weryfikowaną formułę rozbijamy na własności atomowe
- stany wzbogacamy o wartościowanie własności atomowych

Weryfikacja

- chcemy weryfikować różne własności systemów, np.
  - gdy szlaban jest podniesiony, na torach nie ma pociągu
  - w sekcji krytycznej przebywa co najwyżej 1 proces
  - zawsze któryś z procesów ma aktualne dane
- budujemy jakiś model systemu (automat)
- często system modelujemy jako sieć automatów
- weryfikowaną formułę rozbijamy na własności atomowe
- stany wzbogacamy o wartościowanie własności atomowych

Weryfikacja

- chcemy weryfikować różne własności systemów, np.
  - gdy szlaban jest podniesiony, na torach nie ma pociągu
  - w sekcji krytycznej przebywa co najwyżej 1 proces
  - zawsze któryś z procesów ma aktualne dane
- budujemy jakiś model systemu (automat)
- często system modelujemy jako sieć automatów
- weryfikowaną formułę rozbijamy na własności atomowe
- stany wzbogacamy o wartościowanie własności atomowych

Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

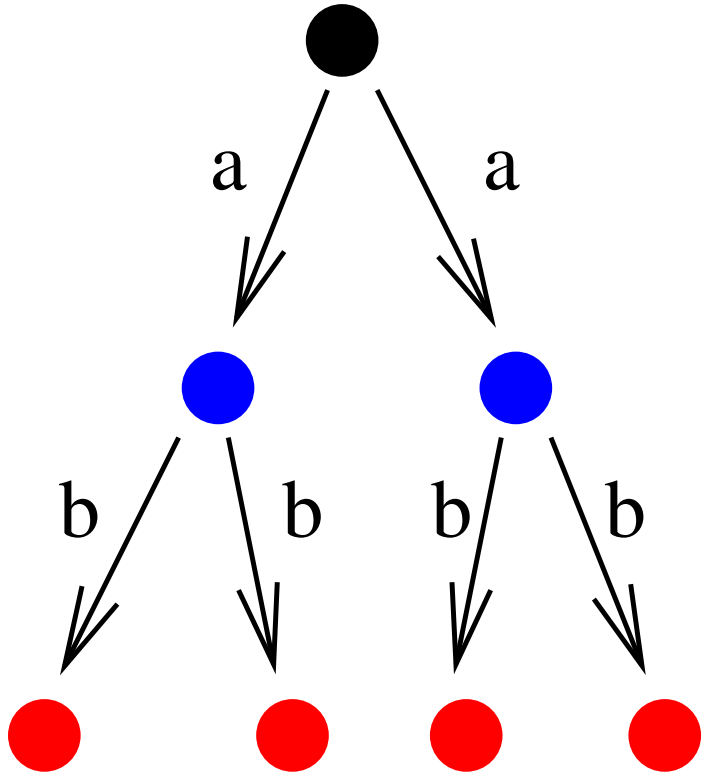
Weryfikacja

- modele mogą być duże (nawet nieskończone)
- konstruuje się więc mniejsze modele abstrakcyjne
- chcemy, by abstrakcyjny model był w pewnej relacji z modelem oryginalnym (np. bisymulacyjnie równoważny)
- relacja między modelami (np. bisymulacja) determinuje, że są one nierozróżnialne przez odpowiednią logikę
- wybieramy taki typ modelu, by własność, którą chcemy zweryfikować była wyrażalna w tej logice
- np. do weryfikowania osiągalności stanów możemy użyć np. bisymulacji, symulacji, pseudosymulacji, pseudobisymulacji

Modele konkretne i abstrakcyjne

Model konkretny

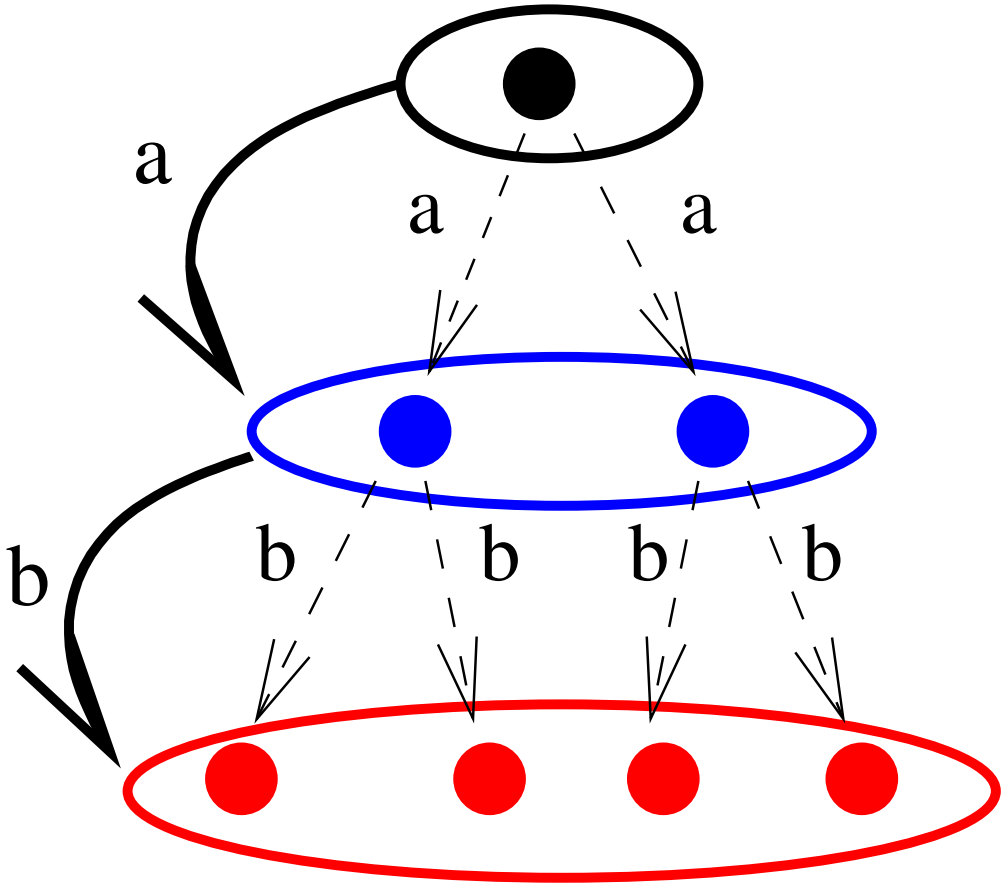
- struktura Kripke-go
- jego stany odpowiadają stanom systemu
- jego tranzycje odpowiadają zmianom w systemie



Modele konkretne i abstrakcyjne

Model abstrakcyjny

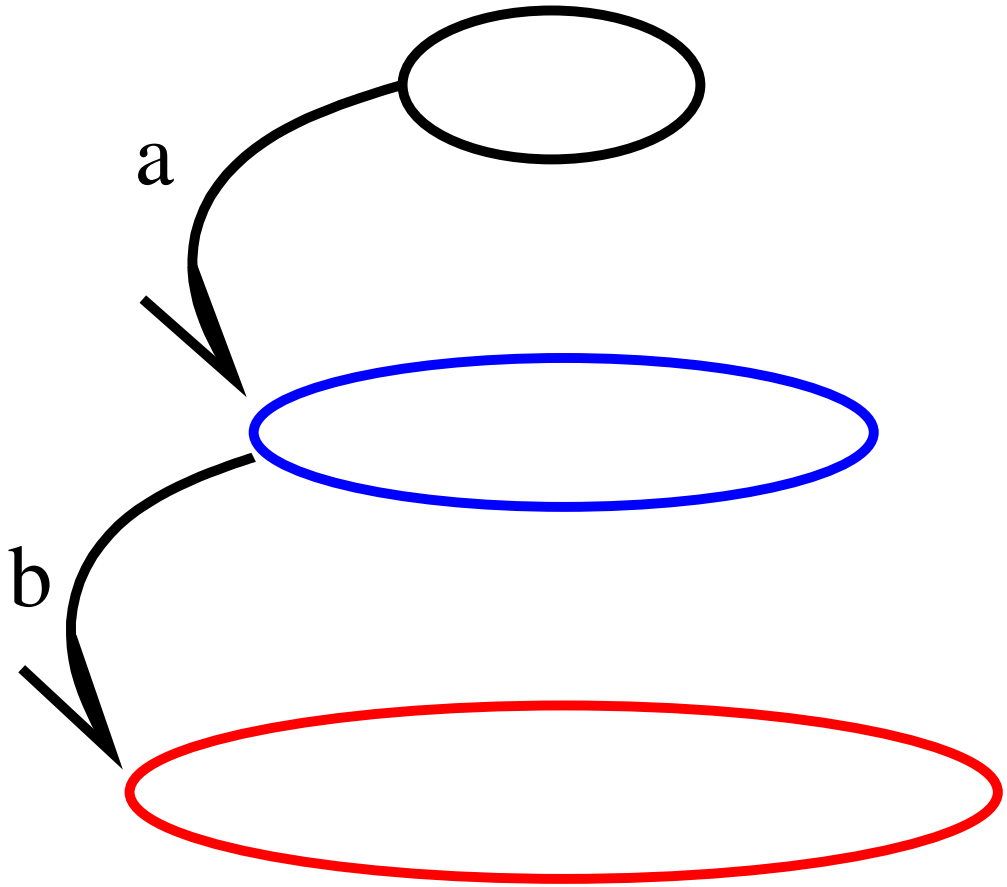
- abstahujemy od rzeczy nieistotnych
- mniejszy stopień szczegółowości
- stan abstrakcyjny to zbiór stanów konkretnych
- modele abstrakcyjne są mniejsze i efektywniejsze



Modele konkretne i abstrakcyjne

Model abstrakcyjny

- abstahujemy od rzeczy nieistotnych
- mniejszy stopień szczegółowości
- stan abstrakcyjny to zbiór stanów konkretnych
- modele abstrakcyjne są mniejsze i efektywniejsze



Model konkretny – definicja

Definicja

Model $\mathbf{M}_c$ to para (K, V_c), gdzie

- K to struktura Kripke-go (S, s_{init}, $\rightarrow$)
- V_c to funkcja wartościowująca $V_c : S \rightarrow 2^{PV}$ (PV to zbiór atomowych własności)

Model konkretny – definicja

Definicja

Model $\mathbf{M}_c$ to para (K, V_c), gdzie

- K to struktura Kripke-go (S, s_{init}, $\rightarrow$)
- V_c to funkcja wartościowująca $V_c : S \rightarrow 2^{PV}$
  (PV to zbiór atomowych własności)

Inaczej

Dla każdego stanu modelu pamiętamy, które własności są w nim spełnione.

Model konkretny – definicja

Definicja

Model $\mathbf{M}_c$ to para (K, V_c), gdzie

- K to struktura Kripke-go (S, s_{init}, $\rightarrow$)
- V_c to funkcja wartościowująca $V_c : S \rightarrow 2^{PV}$
  (PV to zbiór atomowych własności)

Inaczej

Dla każdego stanu modelu pamiętamy, które własności są w nim spełnione.

Rozmiar modelu

Rozmiarem modelu jest ($|S|$, $|\rightarrow|$).

Model abstrakcyjny definicja

Definicja (1/3)

Model abstrakcyjny **M** to para (G, V), gdzie

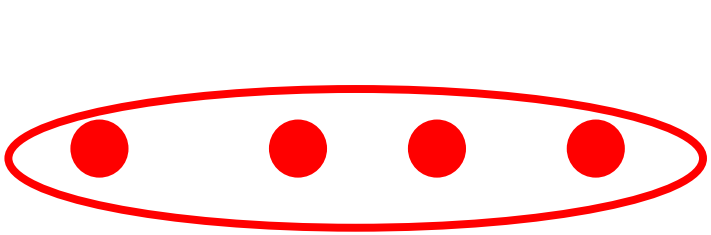
- G to struktura Kripke-go (W, w_{init}, $\rightarrow$)
- stany struktury G to węzły
- każdy węzeł w $\in$ W jest zbiorem stanów S oraz $s_{init} \in w_{init}$
- relacja przejścia etykietowana tym samym alfabetem E, co w modelu konkretnym
- V to funkcja wartościująca $V : W \rightarrow 2^{PV}$
- oraz ...

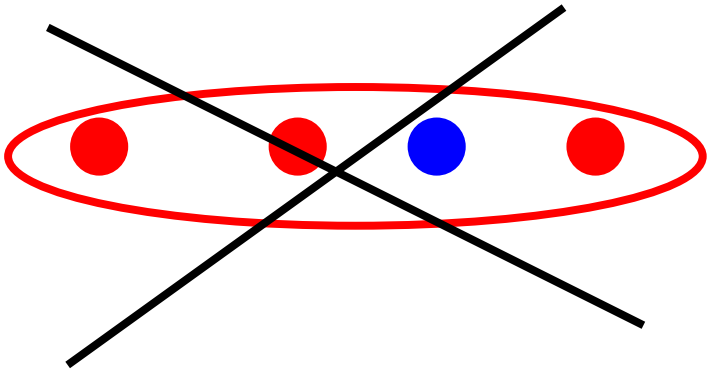
Model abstrakcyjny definicja

Definicja (2/3)

Model abstrakcyjny **M** to para (G, V), gdzie

- ...
- $\forall$ w $\in$ W i s $\in$ w zachodzi $V_c(s) = V(w)$





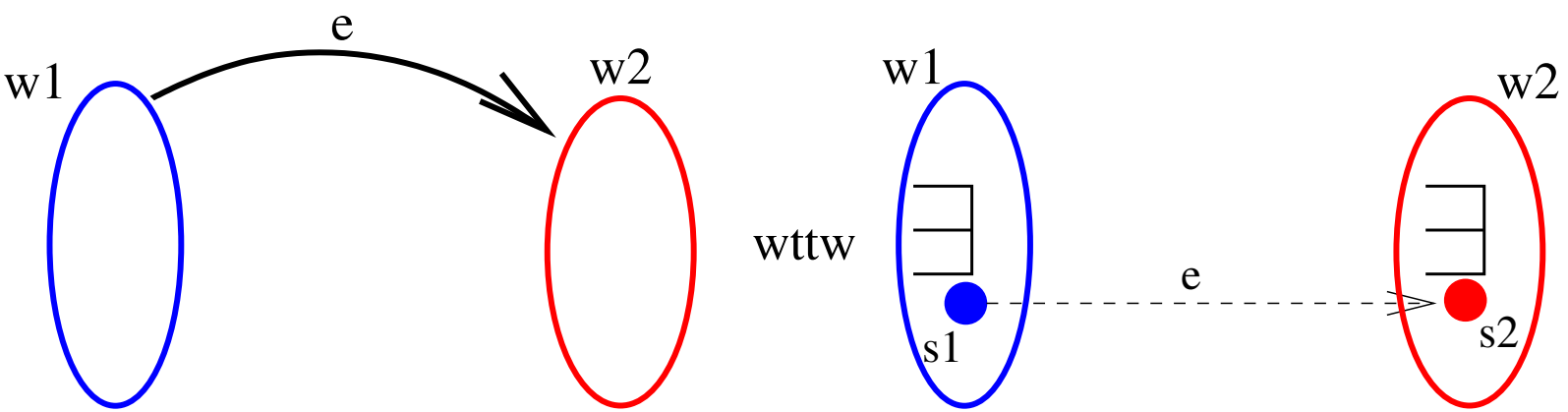
- oraz ...

Model abstrakcyjny definicja

Definicja (3/3)

Model abstrakcyjny **M** to para (G, V), gdzie

- ...
- $\forall\, w_1, w_2 \in \text{Reach}(W), \forall\, e \in E, w_1 \rightarrow^e w_2$
  wttw
  $\exists\, s_1 \in w_1, s_2 \in w_2, s_1 \rightarrow^e s_2$



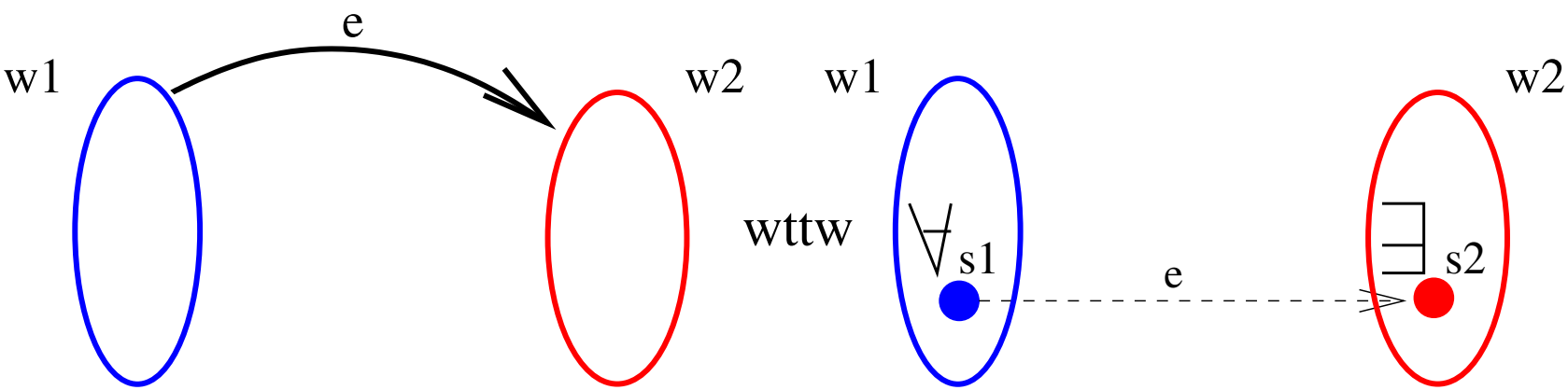
Przegląd modeli

Wzmacniając niektóre warunki w definicji modelu abstrakcyjnego możemy wyprowadzić różne modele, np:

- **model bisymulacyjny**
  (model abstrakcyjny bisymulacyjnie równoważny odpowiedniemu modelowi konkretnemu)
- **model symulacyjny**
  (model abstrakcyjny symulacyjnie równoważny odpowiedniemu modelowi konkretnemu)
- ...
- i wiele innych

Model bisymulacyjny

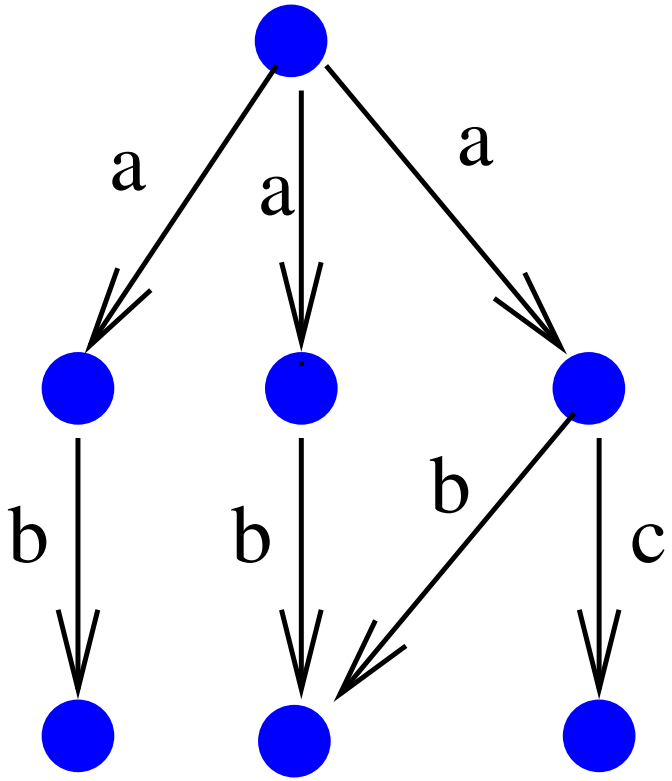
Model abstrakcyjny M = (G, V) dla modelu konkretnego M_c = (K, V_c) nazywany jest **modelem bisymulacyjnym** wttw ...



... dla każdej pary osiągalnych węzłów, jest ona połączona tranzycją wttw zachodzi **warunek bisymulacyjny**

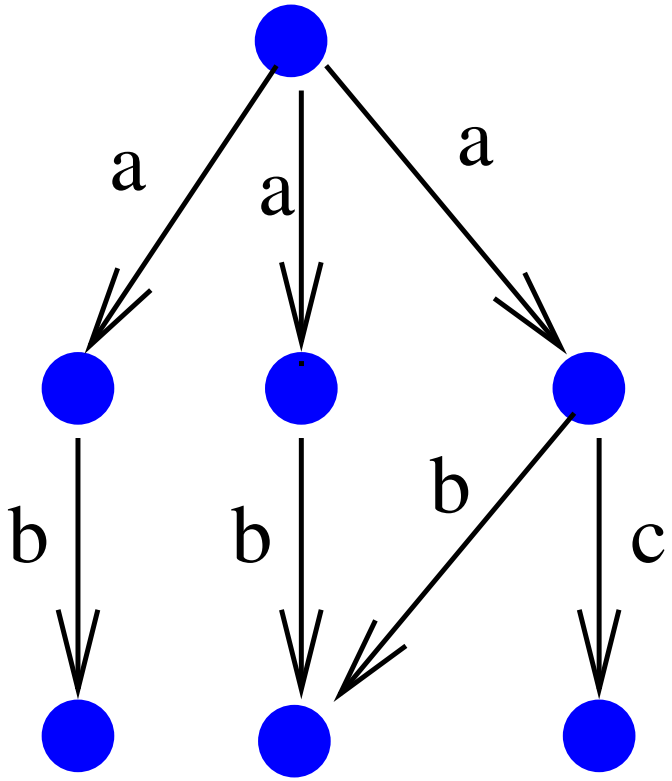
Model konkretny i bisymulacyjny

Model konkretny

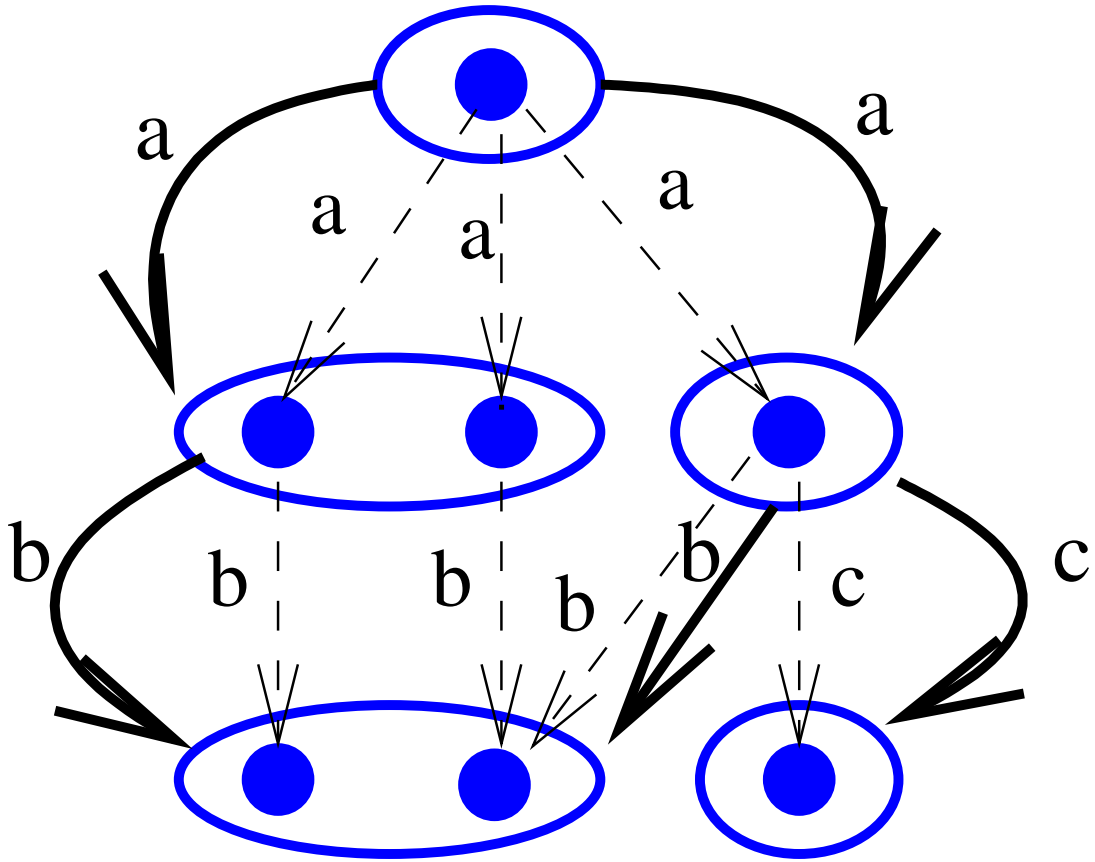


Model konkretny i bisymulacyjny

Model konkretny



Model bisymulacyjny



Własności modelu bisymulacyjnego

Lemat

Model konkretny $M_c = (K, V_c)$ oraz model bisymulacyjny $M = (G, V)$ są bisymulacyjnie równoważne.

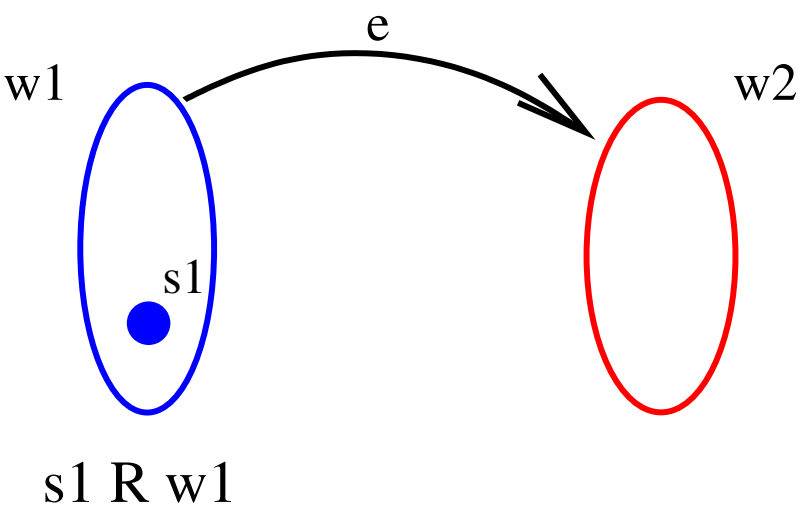
Dowód

Pokażemy, że relacja $\mathcal{R} \subseteq S \times W$,
zdefiniowana jako $\mathcal{R} = \{(s, w) \mid s \in w\}$ jest bisymulacją.

Z definicji modelu bisymulacyjnego $s_{init} \mathcal{R} w_{init}$ i $w_{init} \mathcal{R}^{-1} s_{init}$.

Dowód lematu (2/3)

jesli

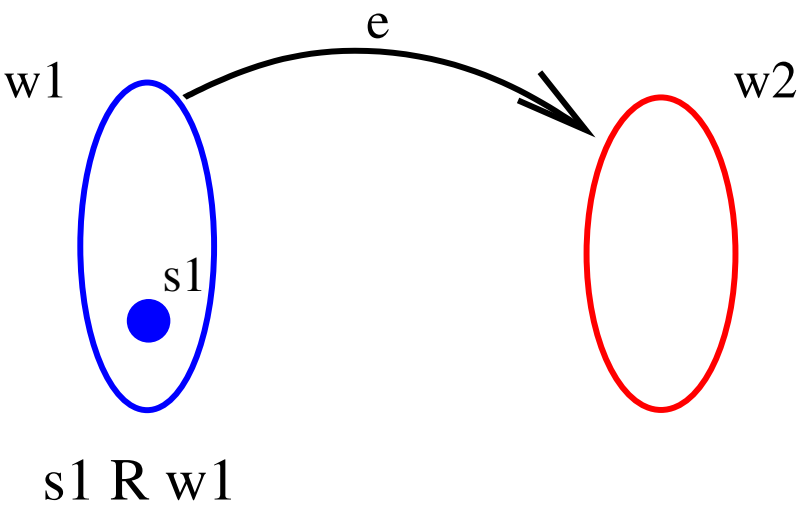


s1 R w1

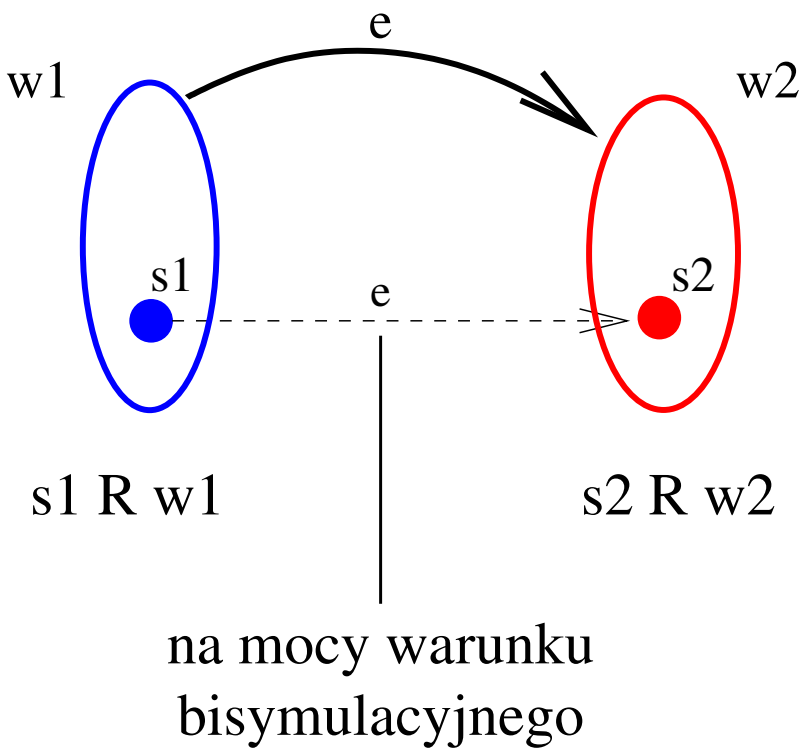
Własności modelu bisymulacyjnego

Dowód lematu (2/3)

jesli

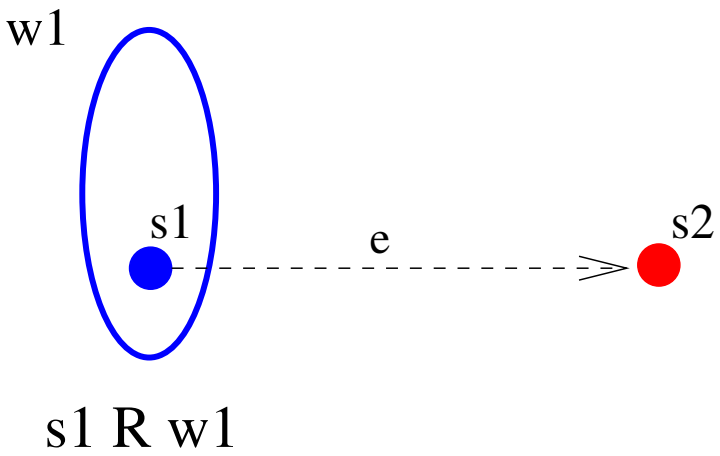


to



Dowód lematu (3/3)

jesli

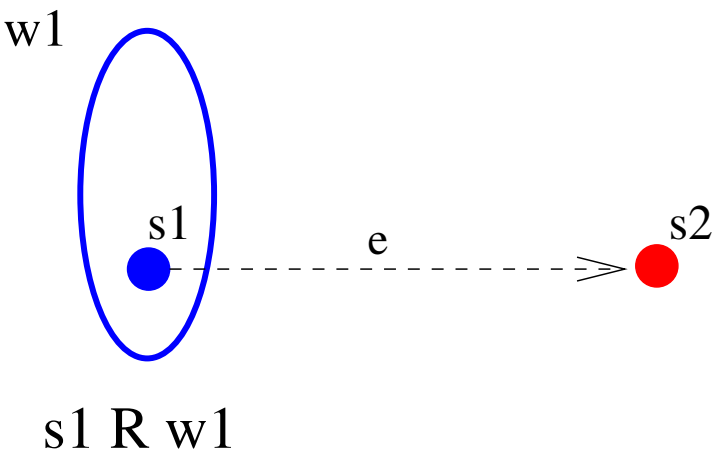


s1 R w1

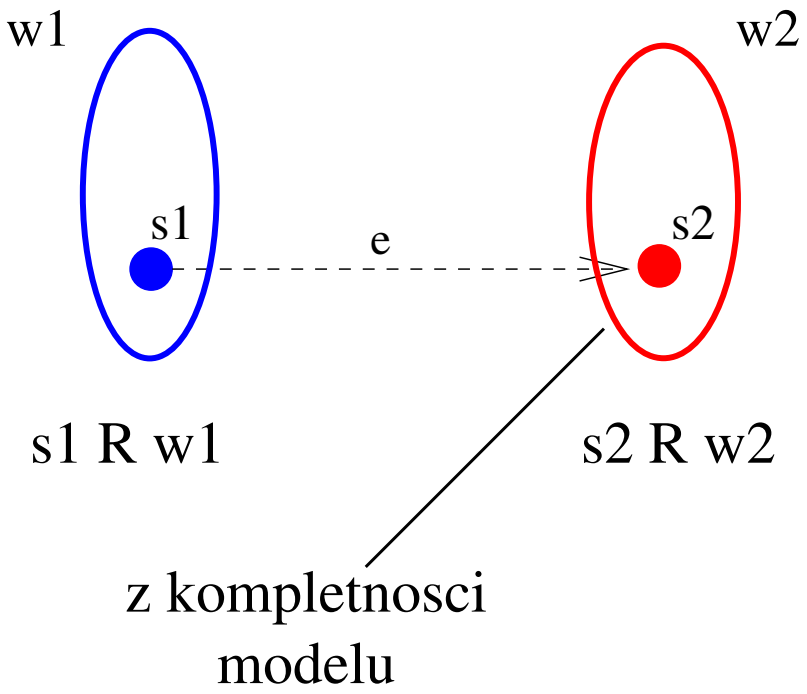
Własności modelu bisymulacyjnego

Dowód lematu (3/3)

jesli

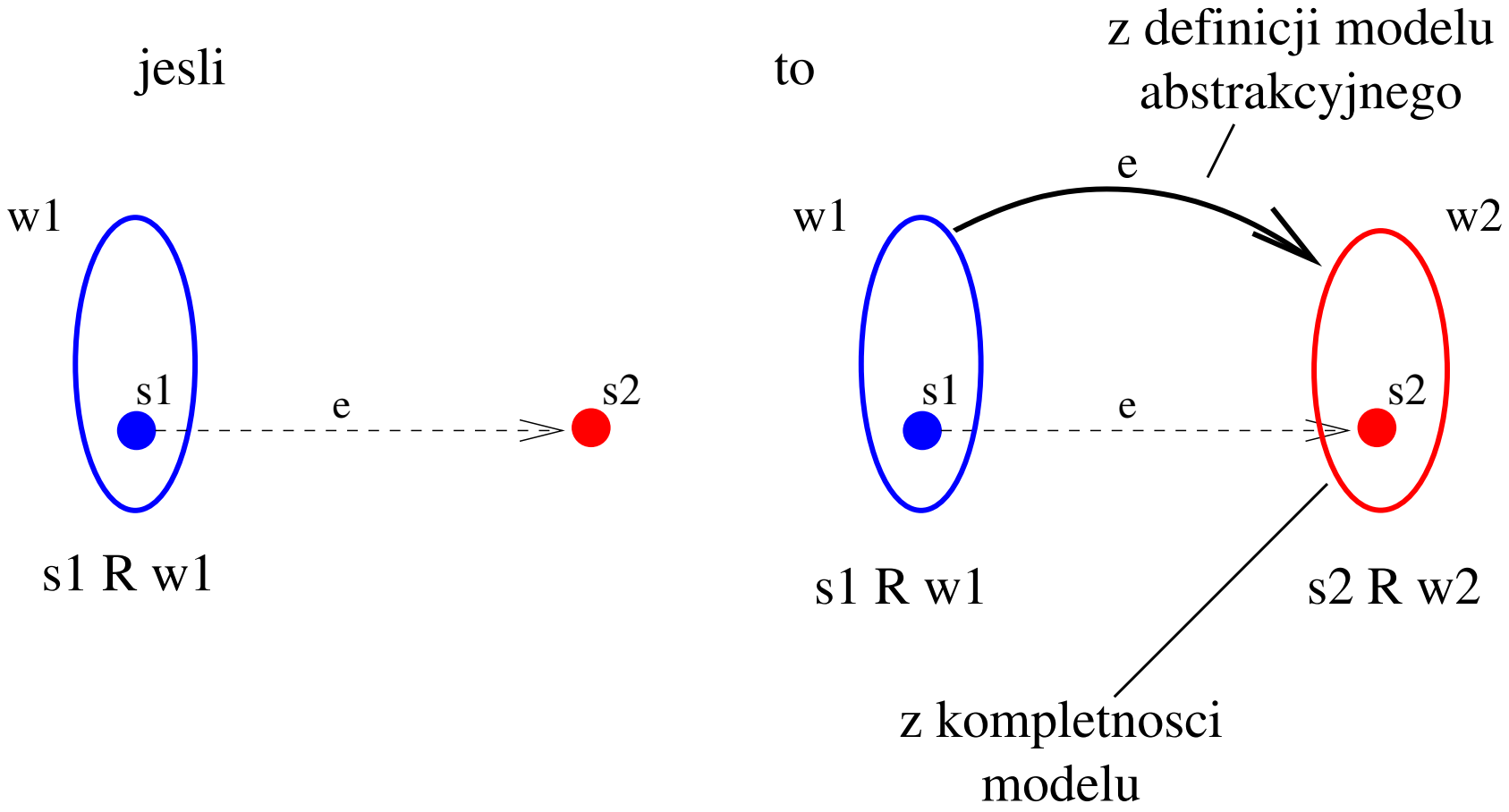


to



Własności modelu bisymulacyjnego

Dowód lematu (3/3)



Algorytm minimalizacji

Algorym minimalizacji (uszczegółowianie podziału)

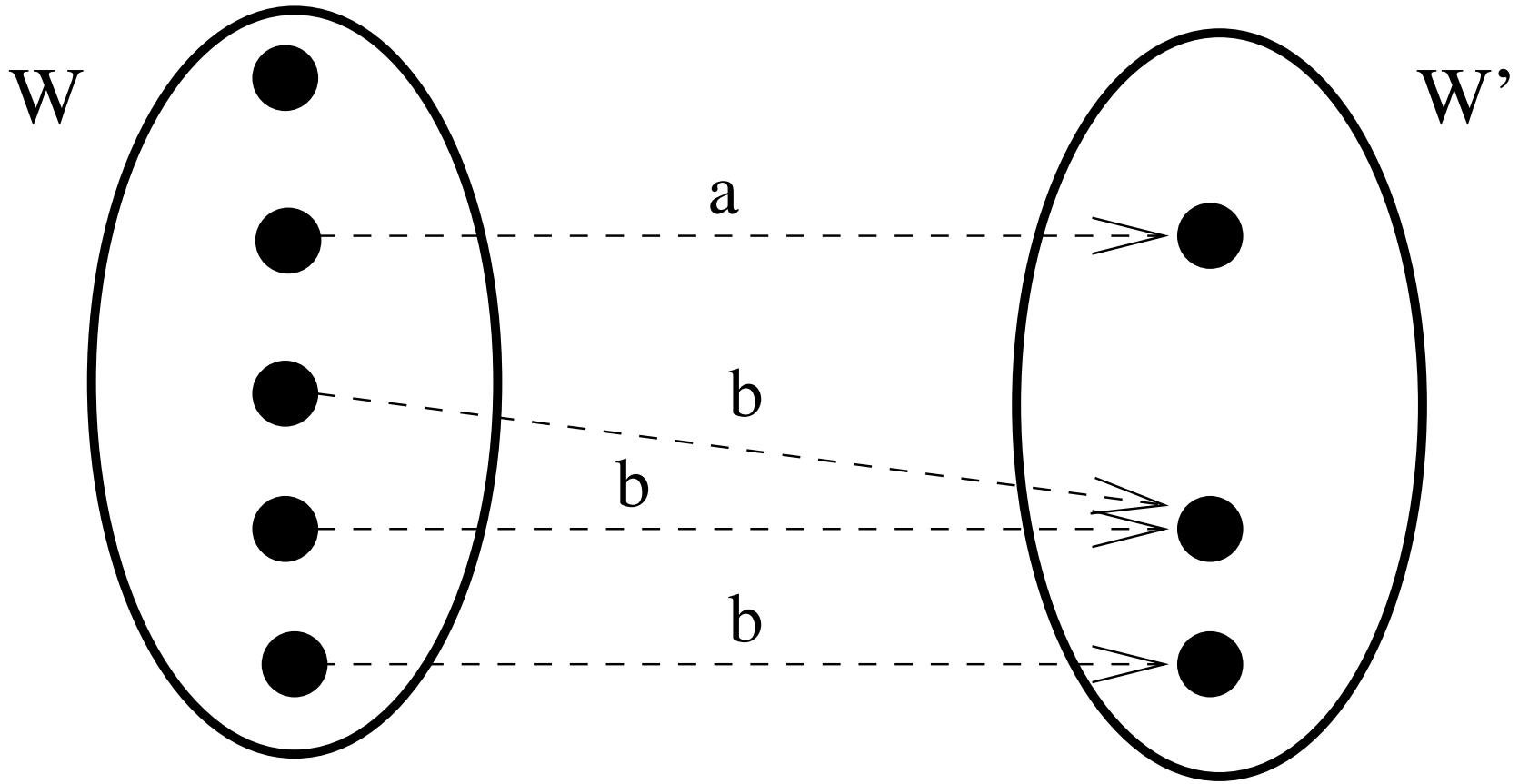
Idea

- zaczynamy od pewnego dość ogólnego początkowego podziału stanów konkretnych (bądź ich osiągalnej części)
- stopniowo uszczegółowiamy podział
- kończymy gdy wszystkie klasy podziału będą spełniać określone warunki (stabilność podziału)

Opracje na klasach podziału

$pre_b(W, W')$

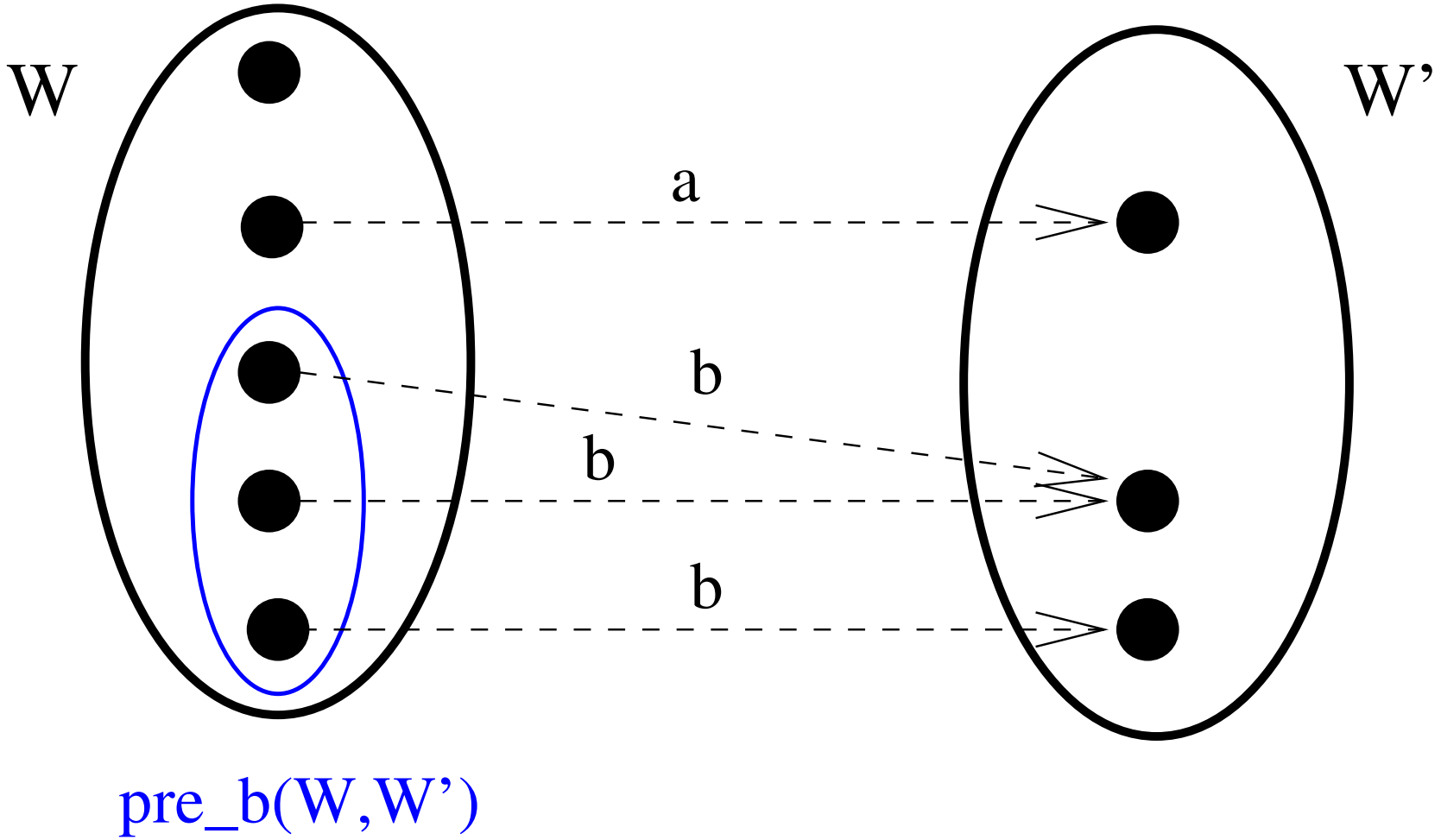
$pre_b(W, W') = \{ s \in W \mid \exists s' \in W' : s \rightarrow^b s' \}$



Opracje na klasach podziału

$pre_b(W, W')$

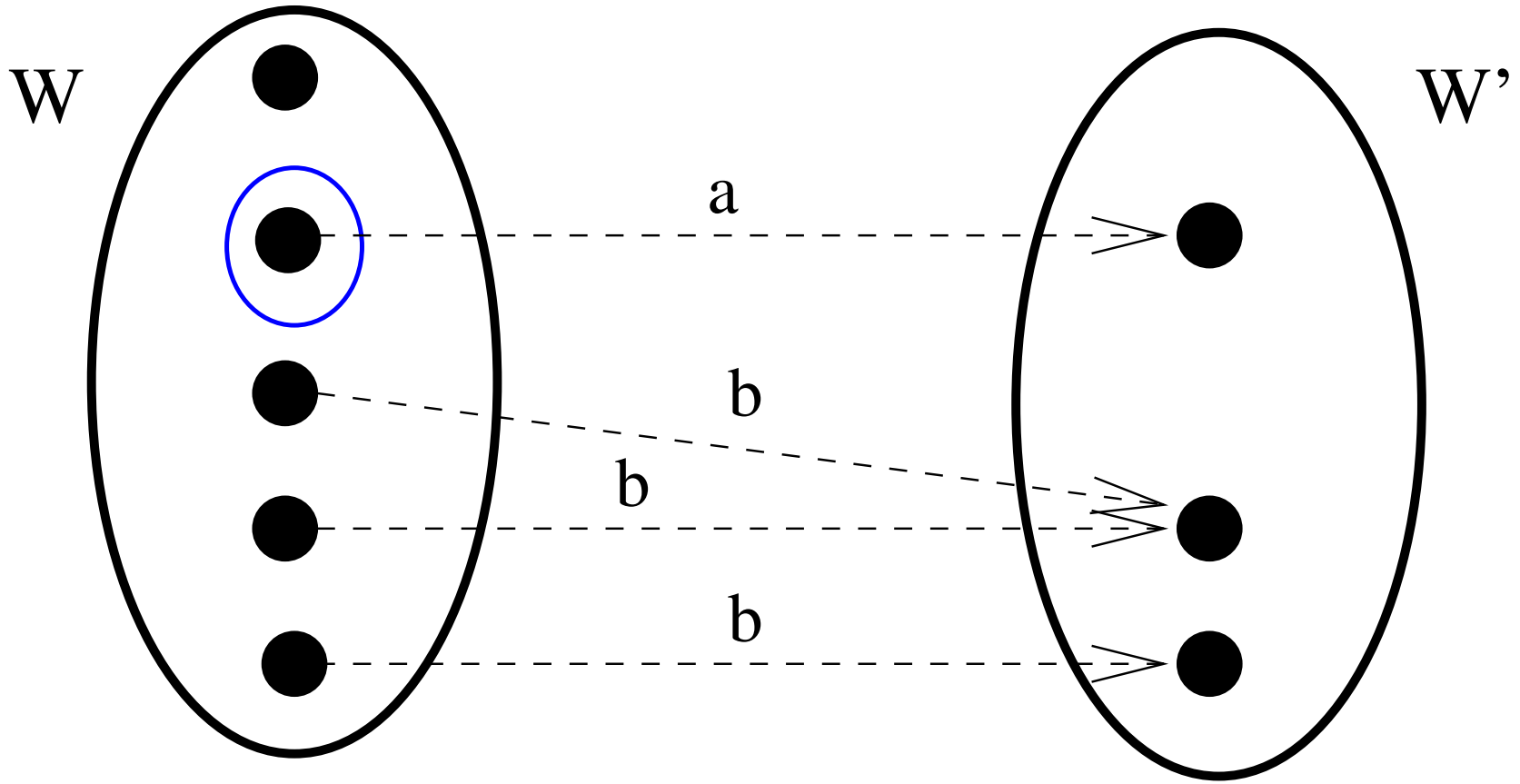
$pre_b(W, W') = \{ s \in W \mid \exists s' \in W' : s \rightarrow^b s' \}$



Opracje na klasach podziału

$pre_b(W, W')$

$pre_b(W, W') = \{ s \in W \mid \exists s' \in W' : s \rightarrow^b s' \}$

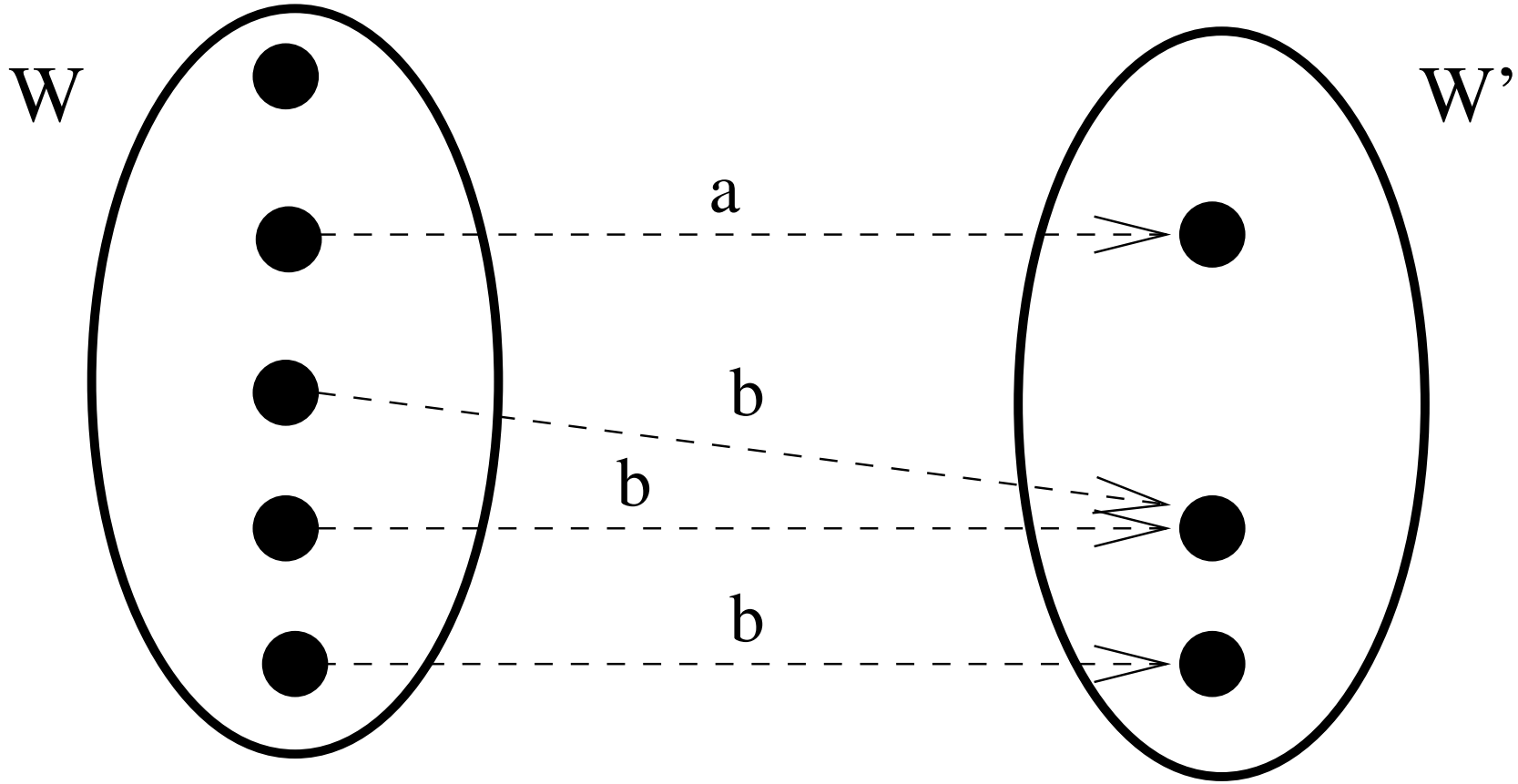


pre_a(W,W')

Opracje na klasach podziału

$post_b(W, W')$

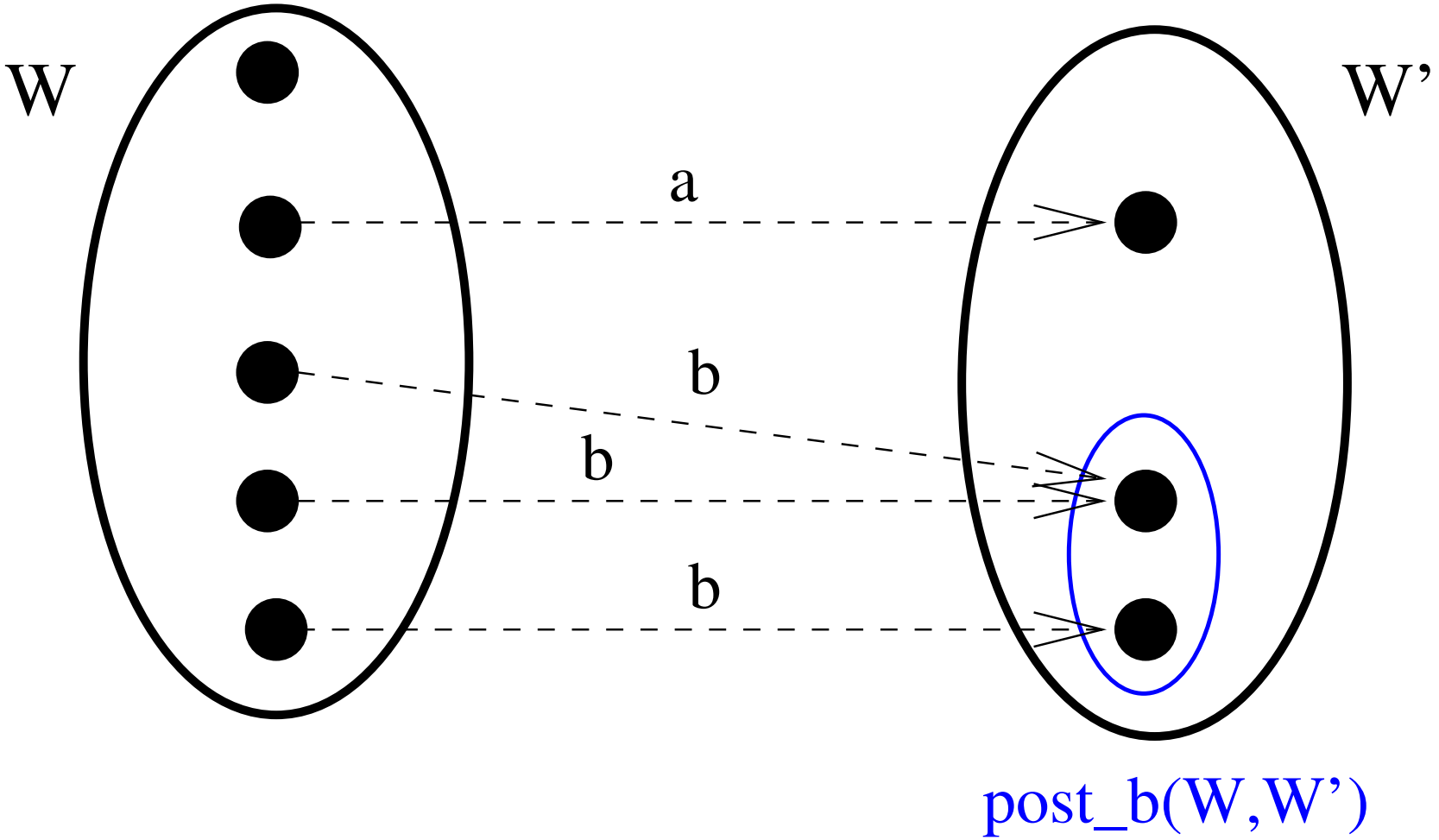
$post_b(W, W') = \{ s' \in W' \mid \exists\, s \in W : s \rightarrow^b s' \}$



Opracje na klasach podziału

$post_b(W, W')$

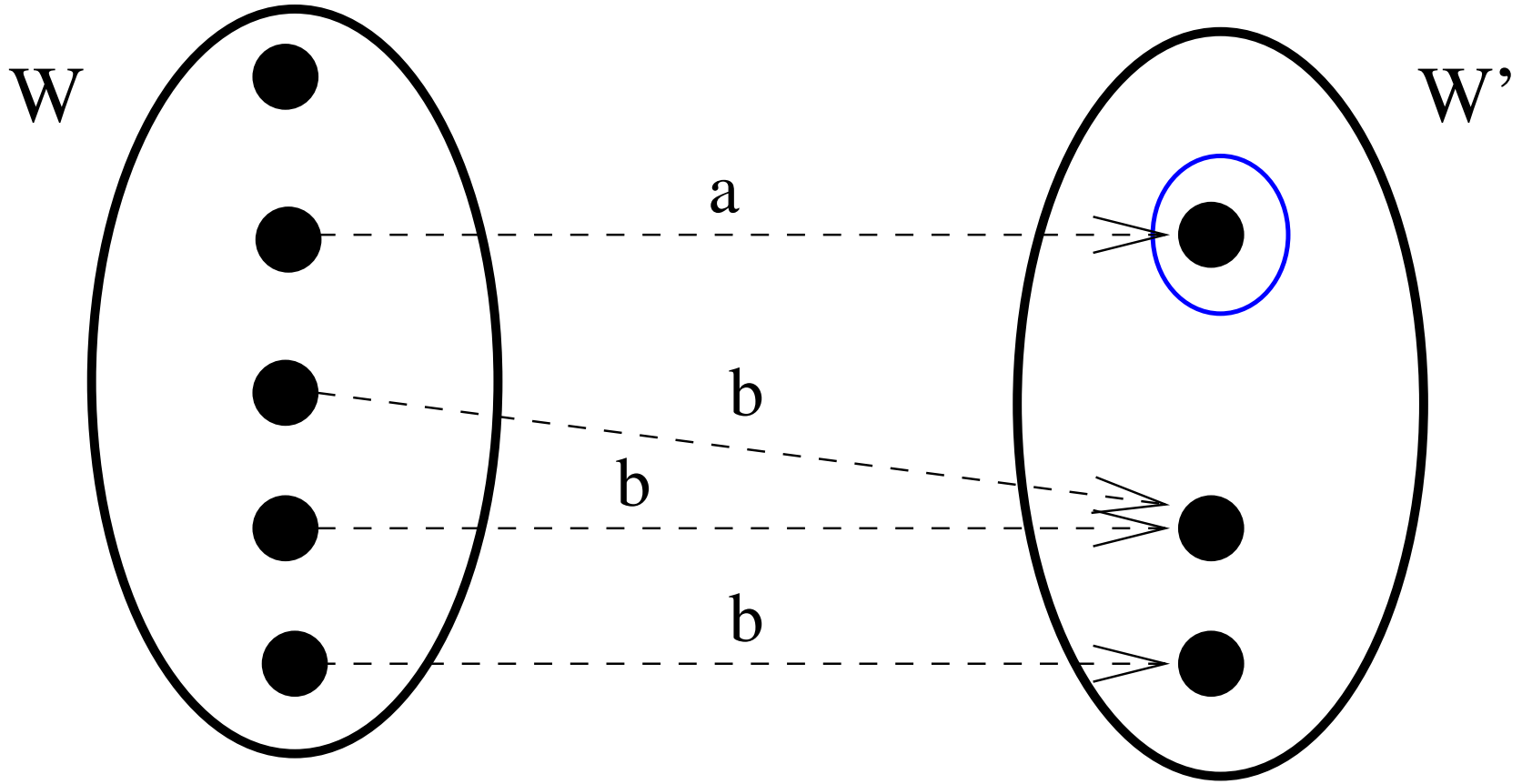
$post_b(W, W') = \{ s' \in W' \mid \exists\, s \in W : s \rightarrow^b s' \}$



Opracje na klasach podziału

$post_b(W, W')$

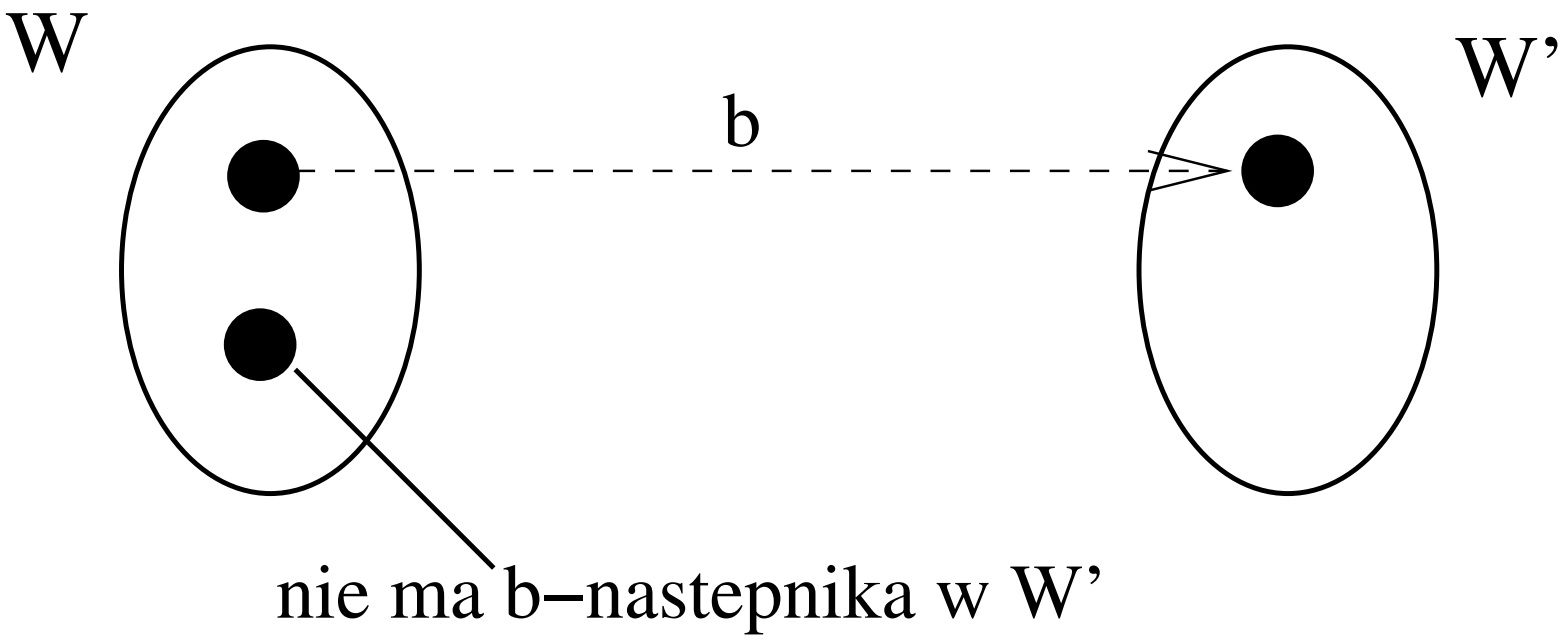
$post_b(W, W') = \{ s' \in W' \mid \exists\, s \in W : s \rightarrow^b s' \}$



bi-niestabliność

Klasa W wymaga modyfikacji, gdy jest bi-niestabilna ze względu na jakąś klasę W'.

- mówimy, że W jest bi-niestabilna ze względu na W' wttw $\exists$ b $\in$ E pre_b(W, W') $\notin$ { W, ϕ }



Algorytm minimalizacji

- rozpoczyna działanie na Π_0 – jakimś początkowym podziale zbioru zawierającego wszystkie konkretne stany osiągalne
- buduje minimalny model
  - stany w modelu to klasy podziału tego zbioru
  - stanem początkowym jest klasa zawierająca konkretny stan początkowy [s_{init}]

Split_{bi}

Algorytm parametryzowany jest pewną niedeterministyczną funkcją $\text{Split}_{bi}(W, \Pi)$.

$\text{Split}_{bi}(W, \Pi)$

- funkcja dokonuje uszczegółowienia podziału klasy W
- wybierana jest klasa W', względem której klasa W jest bi-niestabilna
- dzieli W tak, by otrzymane klasy były bi-stabilne względem W'
- zwracany jest podział klasy W

Split_{bi}

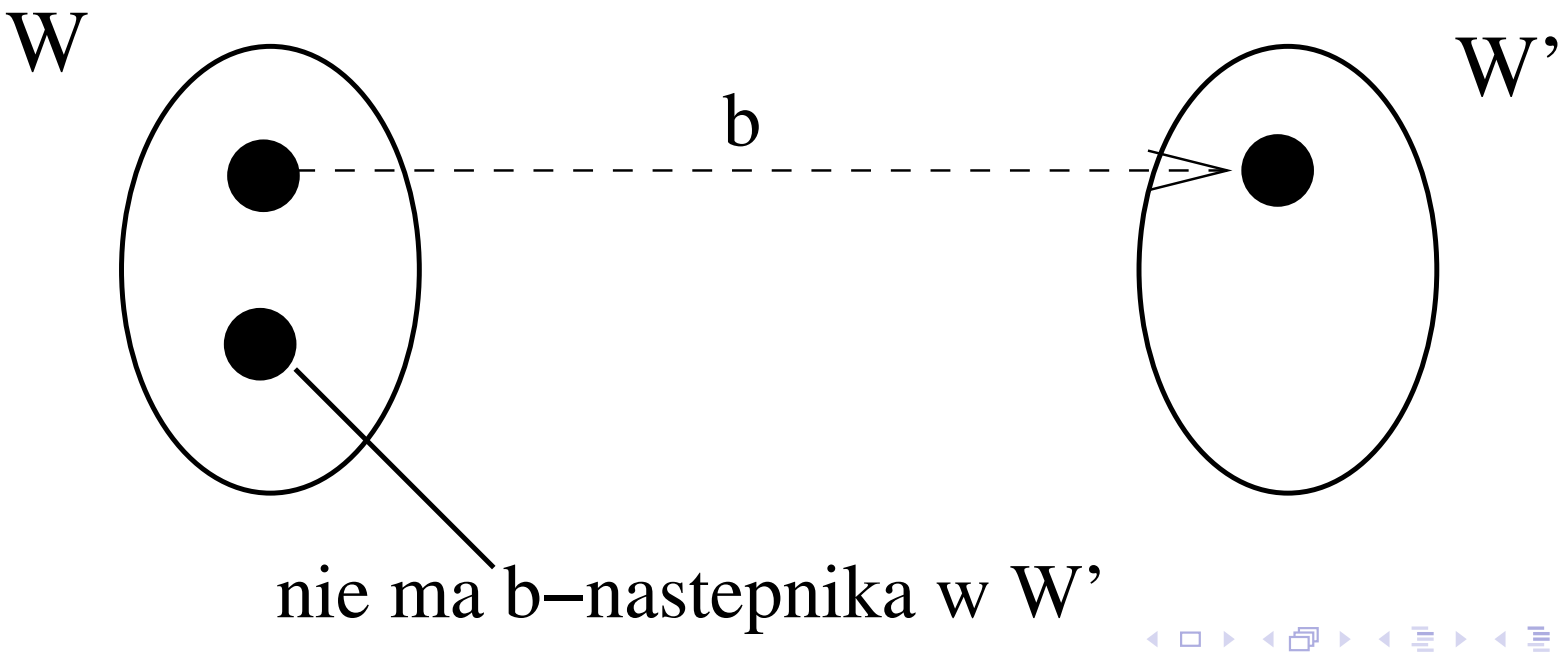
$\text{Split}_{bi}(W, \Pi)$

- $\text{Split}_{bi}(W, \Pi) = \{ W \}$
  jeśli W jest bi-stabilne względem wszystkich $W' \in \Pi$
- $\text{Split}_{bi}(W, \Pi) = \{ \text{pre}_b(W,W'), Y \setminus \text{pre}_b(W,W') \}$
  jeśli W jest bi-niestabilne względem pewnego W', dla pewnego b

Split_{bi}

$\text{Split}_{bi}(W, \Pi)$

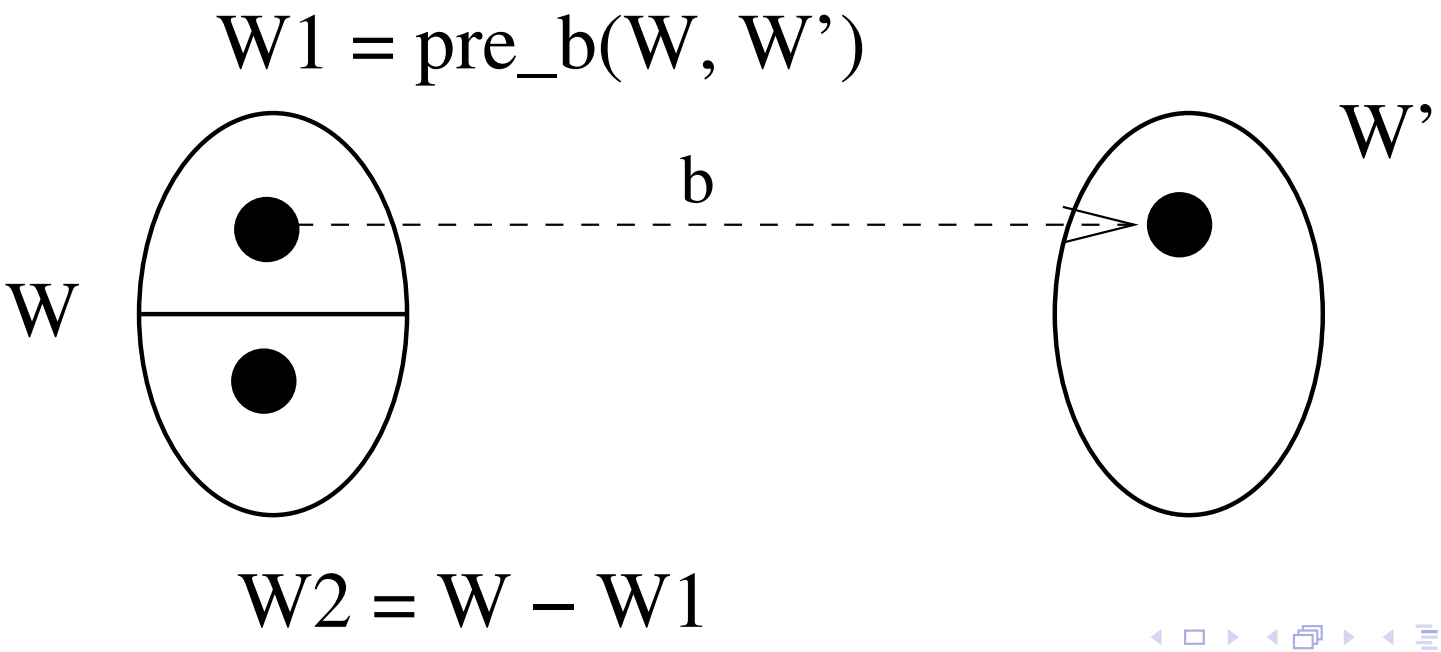
- $\text{Split}_{bi}(W, \Pi) = \{ W \}$
  jeśli W jest bi-stabilne względem wszystkich W' $\in \Pi$
- $\text{Split}_{bi}(W, \Pi) = \{ \text{pre}_b(W,W'), Y \setminus \text{pre}_b(W,W') \}$
  jeśli W jest bi-niestabilne względem pewnego W', dla pewnego b



Split_{bi}

$\text{Split}_{bi}(W, \Pi)$

- $\text{Split}_{bi}(W, \Pi) = \{ W \}$
  jeśli W jest bi-stabilne względem wszystkich $W' \in \Pi$
- $\text{Split}_{bi}(W, \Pi) = \{ \text{pre}_b(W,W'), Y \setminus \text{pre}_b(W,W') \}$
  jeśli W jest bi-niestabilne względem pewnego W', dla pewnego b



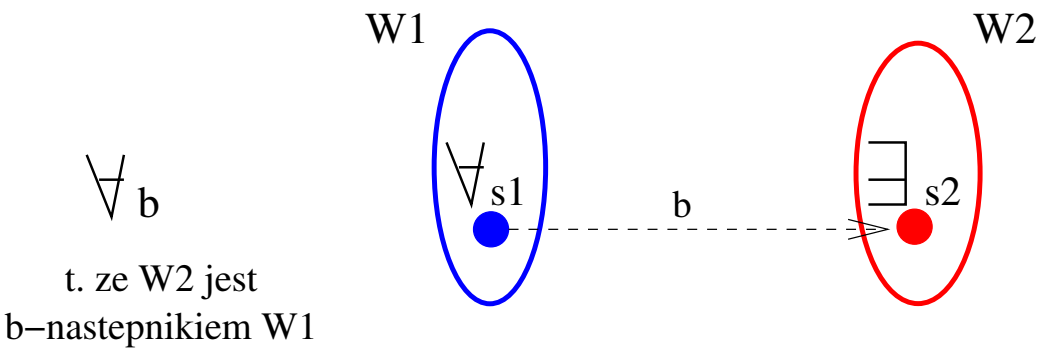
Algorytm

Zarys algorytmu

- **reachable** – zbiór klas osiągalnych
- **stable** – zbiór klas stabilnych
- początkowo **reachable** := { [s_{init}] }, **stable** := ϕ
- w kolejnych krokach testujemy stabilność klas ze zbioru **reachable** $\setminus$ **stable**
- jeśli dana klasa jest bi-stabilna względem wszystkich swoich następników – jest dodawana do zbioru **stable** a jej następniki do zbioru **reachable**
- wpp. klasa dzielona jest tak, by zapewnić jej bi-stabilność względem wybranego następnika (który powodował jej bi-niestabilność)
- koniec, gdy wszystkie klasy osiągalne są stabilne

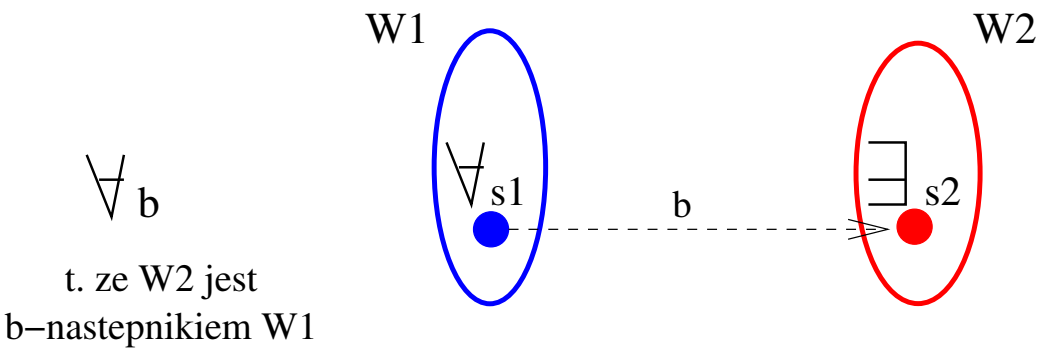
Stabilność

Niech W1 będzie bi-stabilne względem W2.

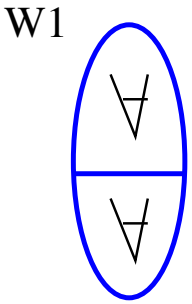


Stabilność

Niech W1 będzie bi-stabilne względem W2.

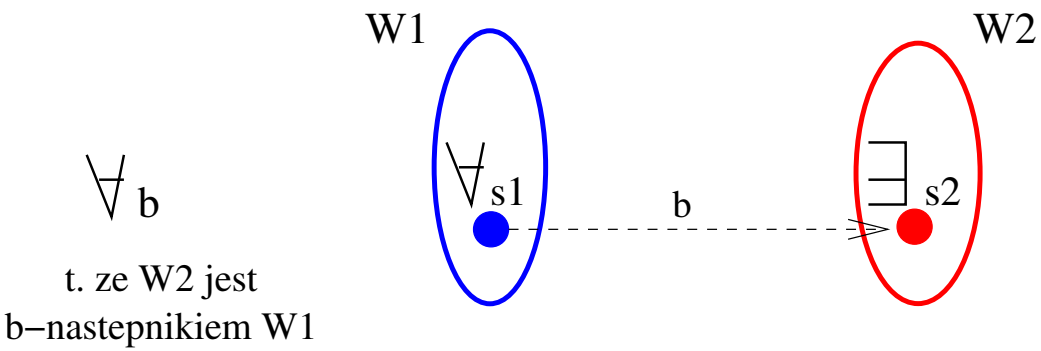


- podział W1 nie powoduje utraty bi-stabilności

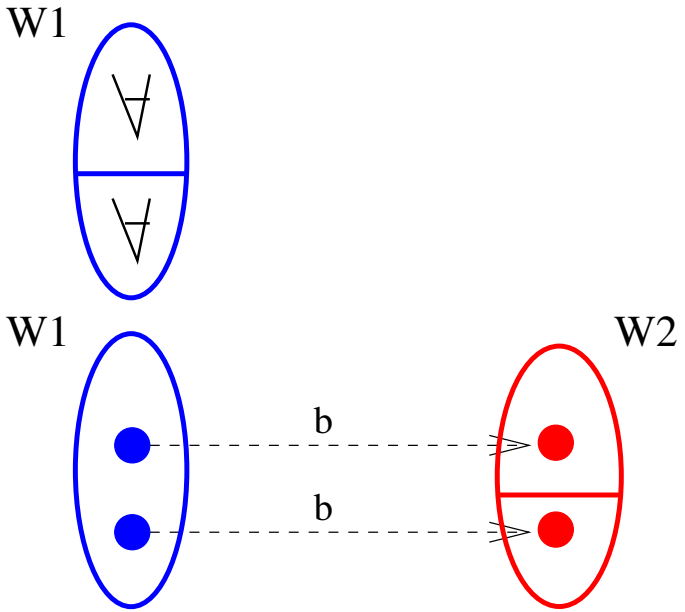


Stabilność

Niech W1 będzie bi-stabilne względem W2.



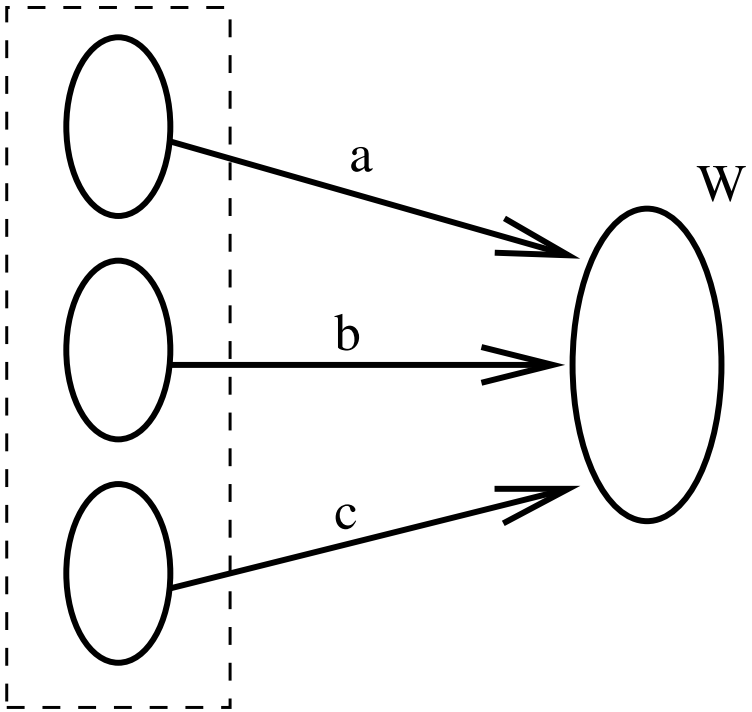
- podział W1 nie powoduje utraty bi-stabilności



- podział W2 może spowodować utratę bi-stabilności

Konsekwencja podziału klasy W

Podział klasy $W \in \Pi$ pociąga za sobą ...



... usunięcie jej poprzedników (czyli $Pre_{\Pi}(W)$) ze zbioru klas stabilnych.

Konsekwencja podziału klasy W

Podział klasy $W \in \Pi$ pociąga za sobą usunięcie W ze zbioru klas **osiągalnych**, ale ...

Konsekwencja podziału klasy W

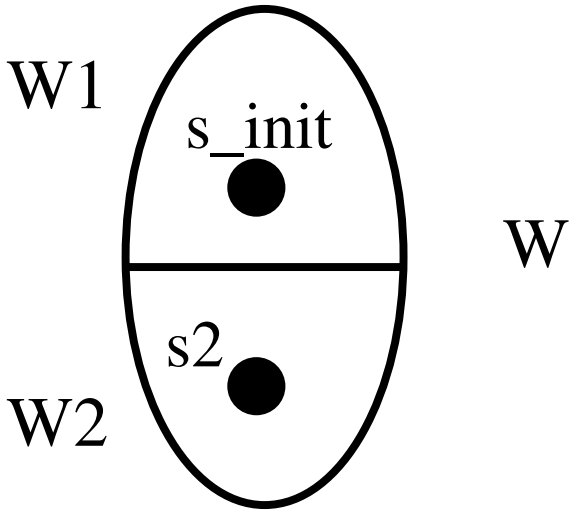
Podział klasy $W \in \Pi$ pociąga za sobą usunięcie W ze zbioru klas **osiągalnych**, ale ...

- chcemy utrzymywać **niezmiennik**:
  klasa zawierająca stan początkowy zawsze należy do zbioru klas osiągalnych

Konsekwencja podziału klasy W

Podział klasy $W \in \Pi$ pociąga za sobą usunięcie W ze zbioru klas **osiągalnych**, ale ...

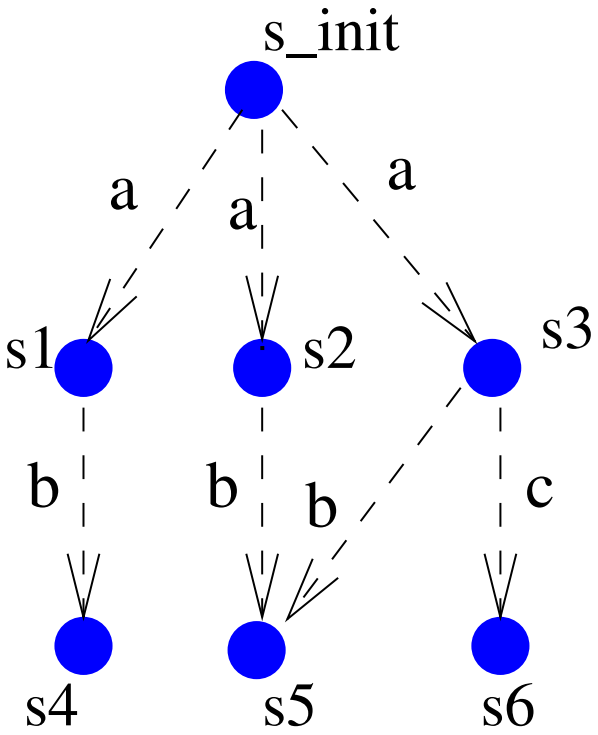
- chcemy utrzymywać **niezmiennik**:
  klasa zawierająca stan początkowy zawsze należy do zbioru klas osiągalnych
- klasę W1 trzeba dodać do zbioru klas osiągalnych



Algorytm – pseudokod

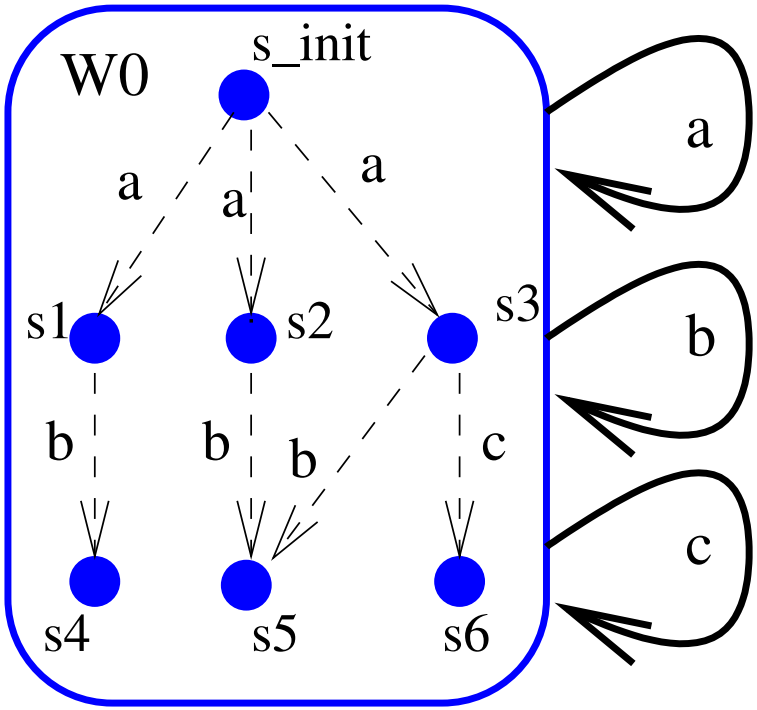
1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | s_{init} $\in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład



1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

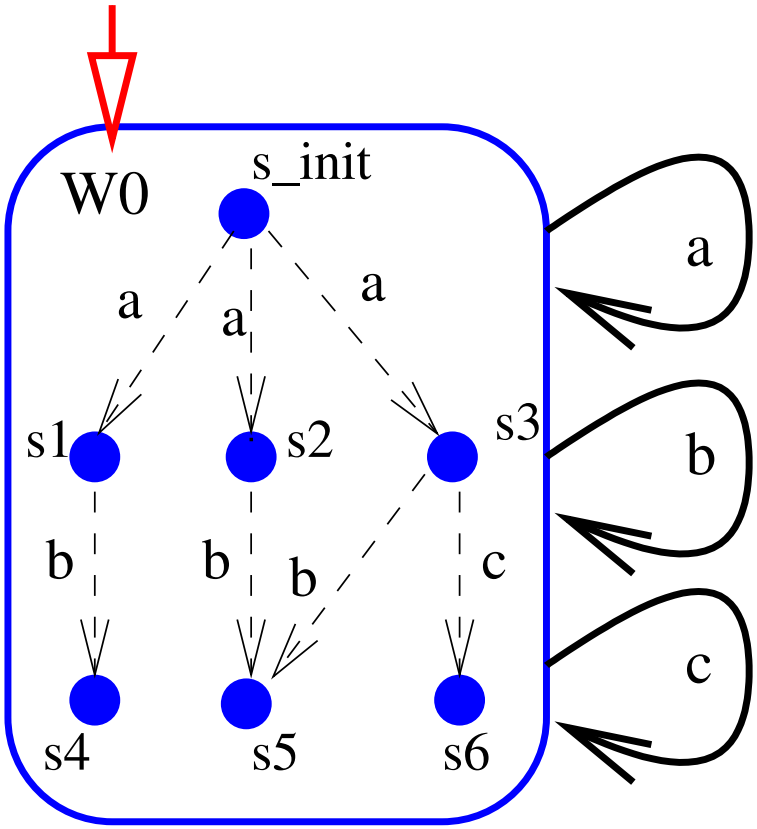
Algorytm – przykład



reachable=$\{W_0\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

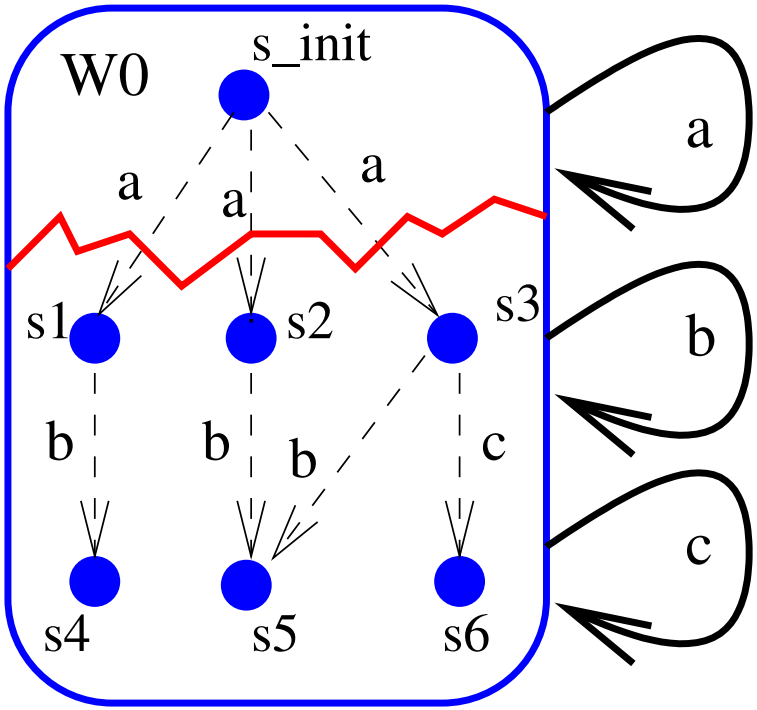
Algorytm – przykład



reachable=$\{W_0\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

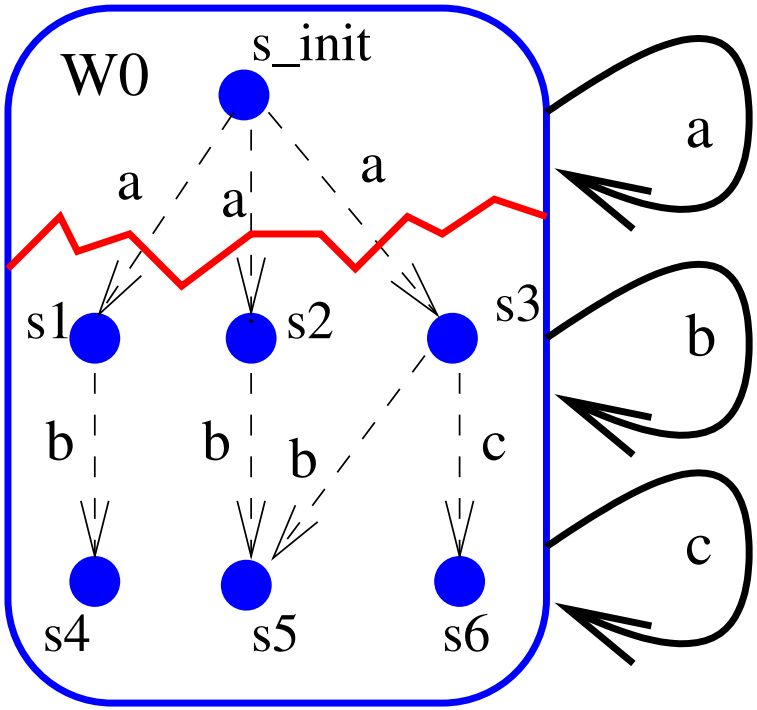
Algorytm – przykład



reachable=$\{W_0\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

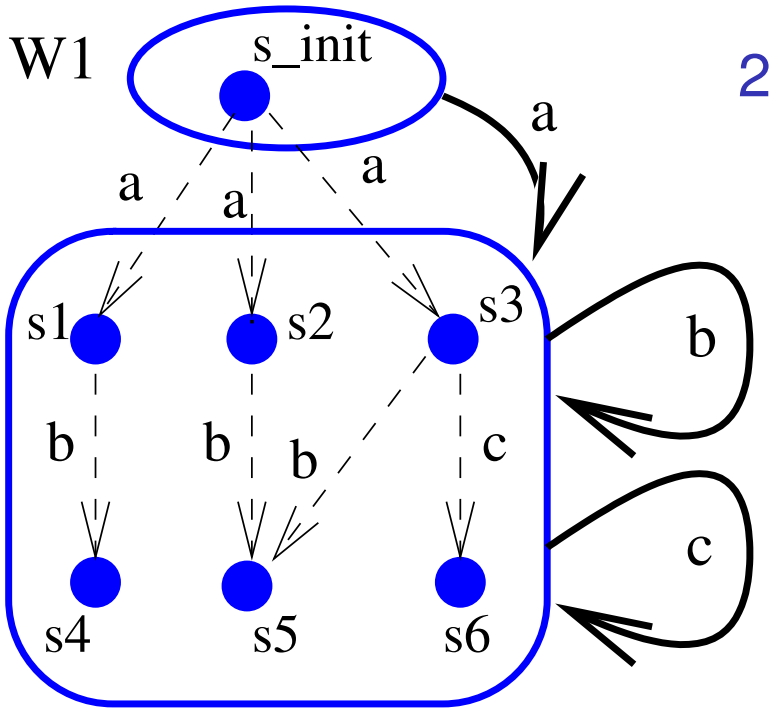
Algorytm – przykład



reachable=$\{W_0\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

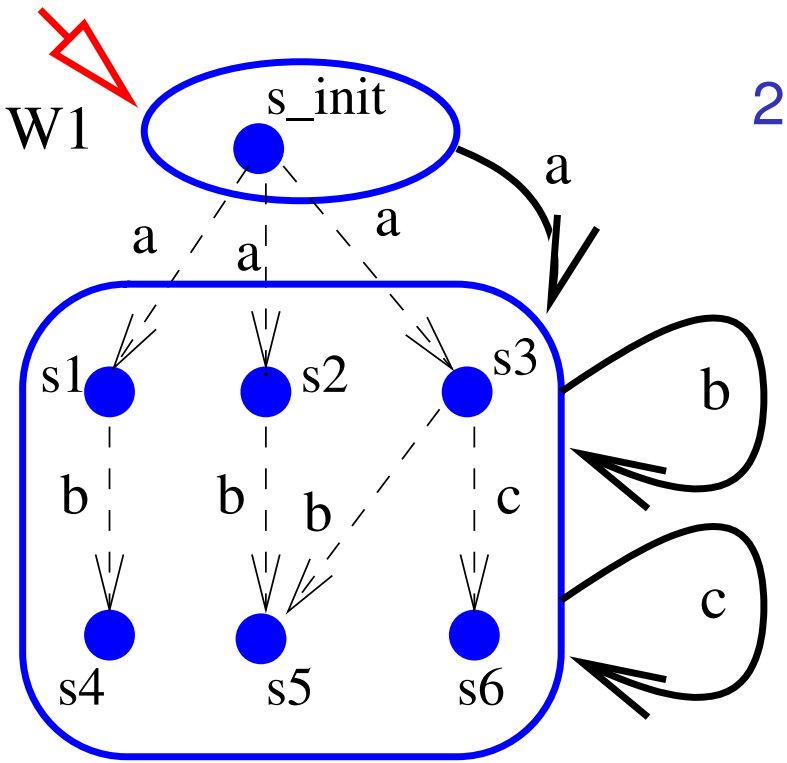
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

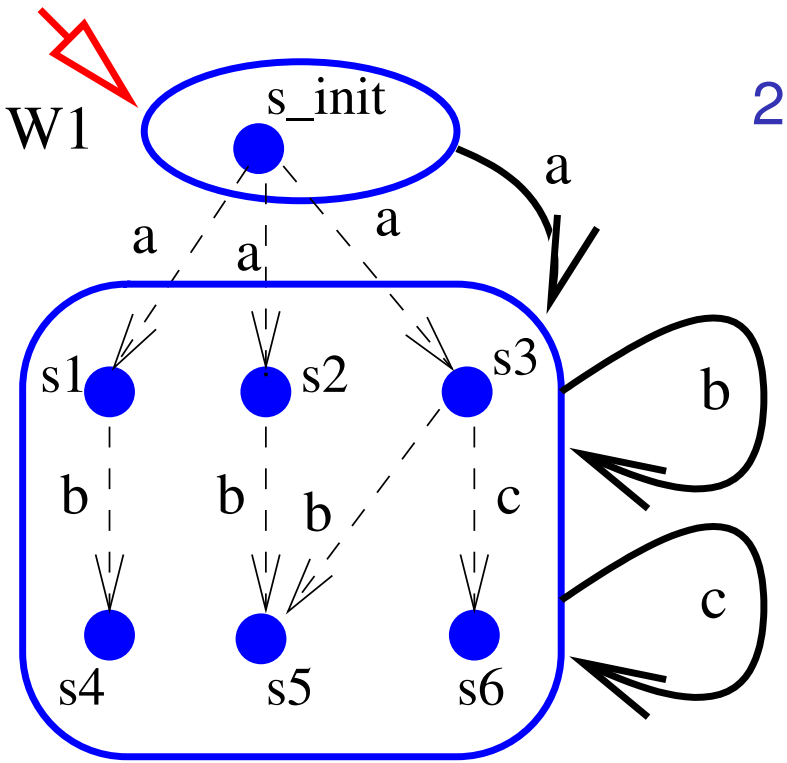
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

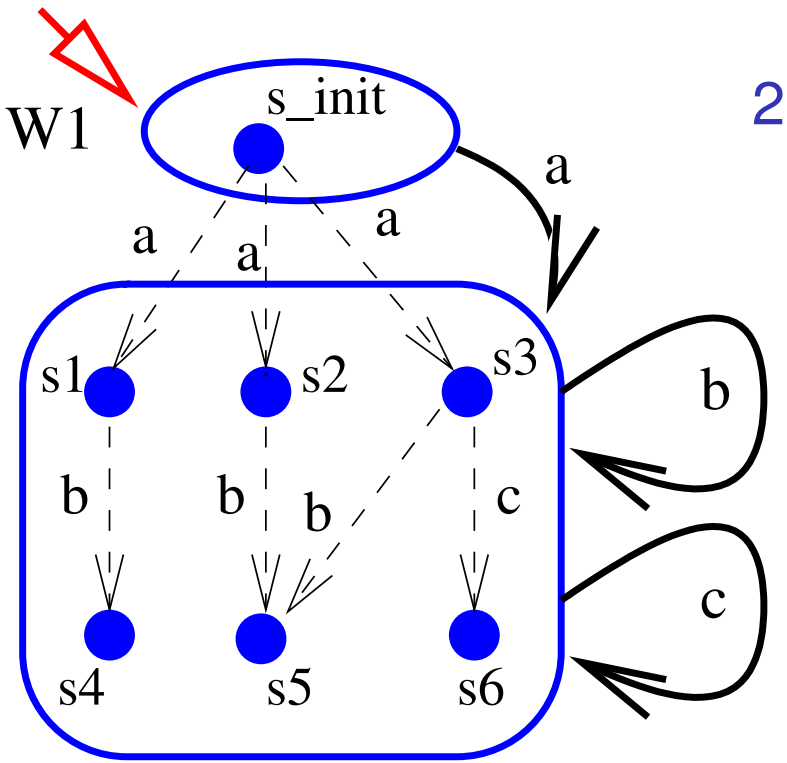
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

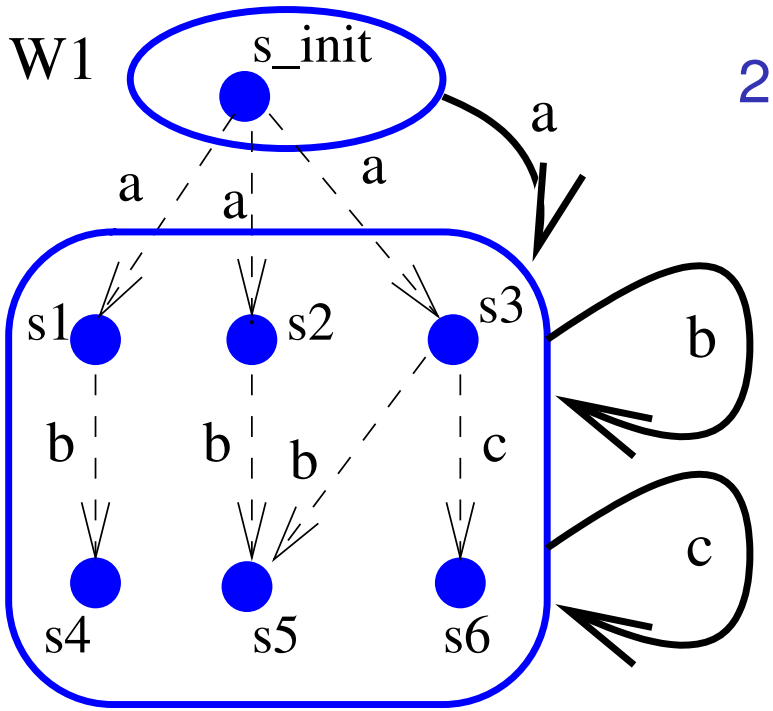
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład



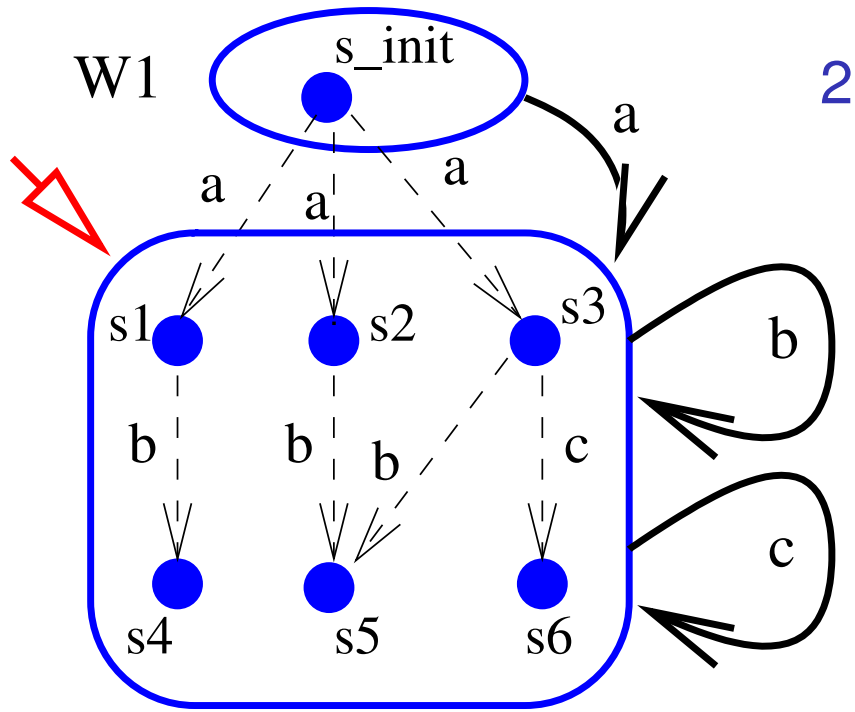
W2

reachable=$\{W_1, W_2\}$,

stable=$\{W_1\}$

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład

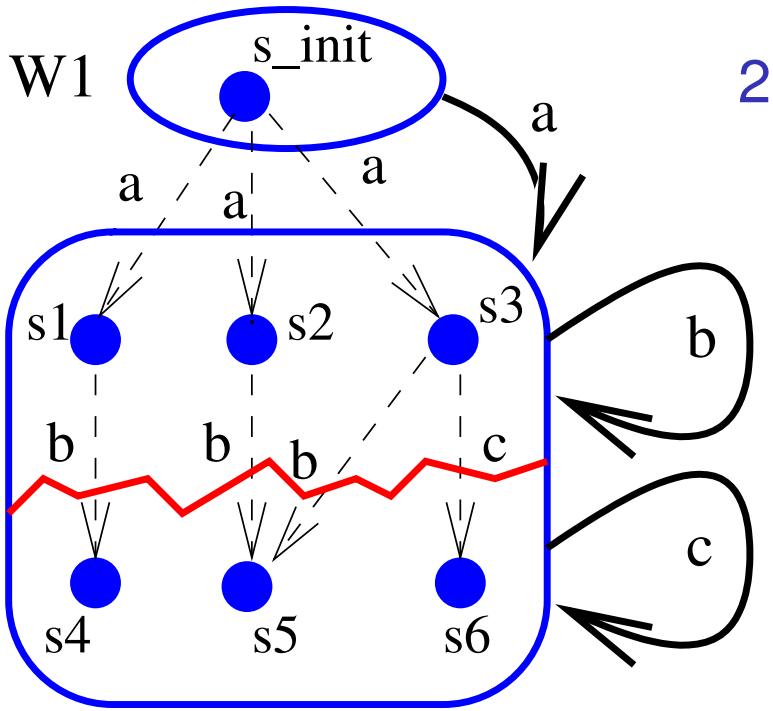


W2

reachable={W_1, W_2},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład



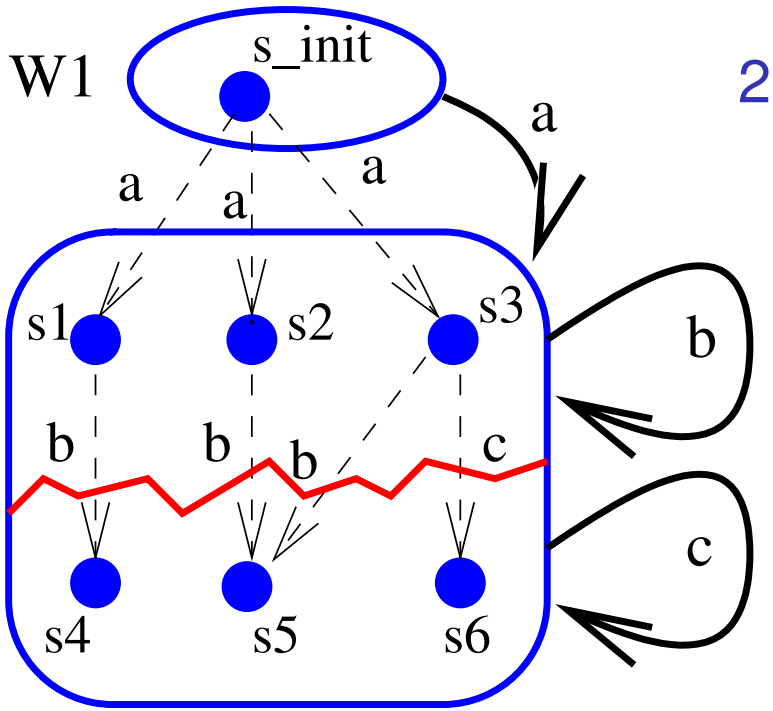
W2

reachable={W_1, W_2},

stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := $Split_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         $Post_\Pi$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_Π(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład



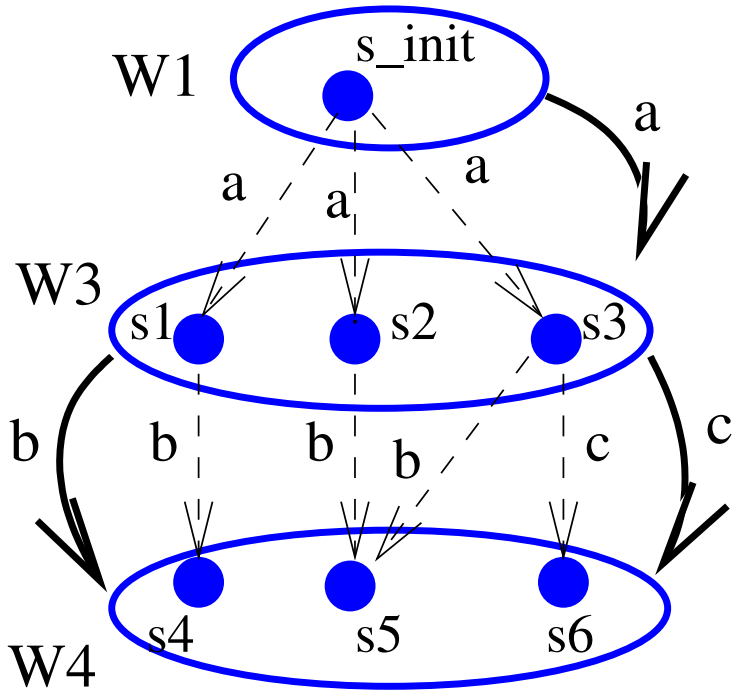
W2

reachable={W_1, W_2},

stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]}; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

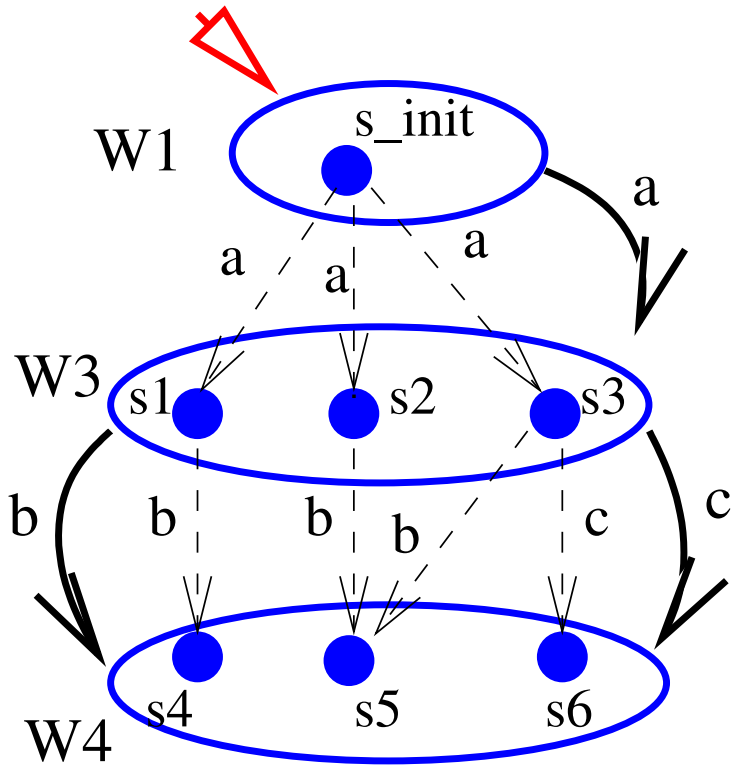
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

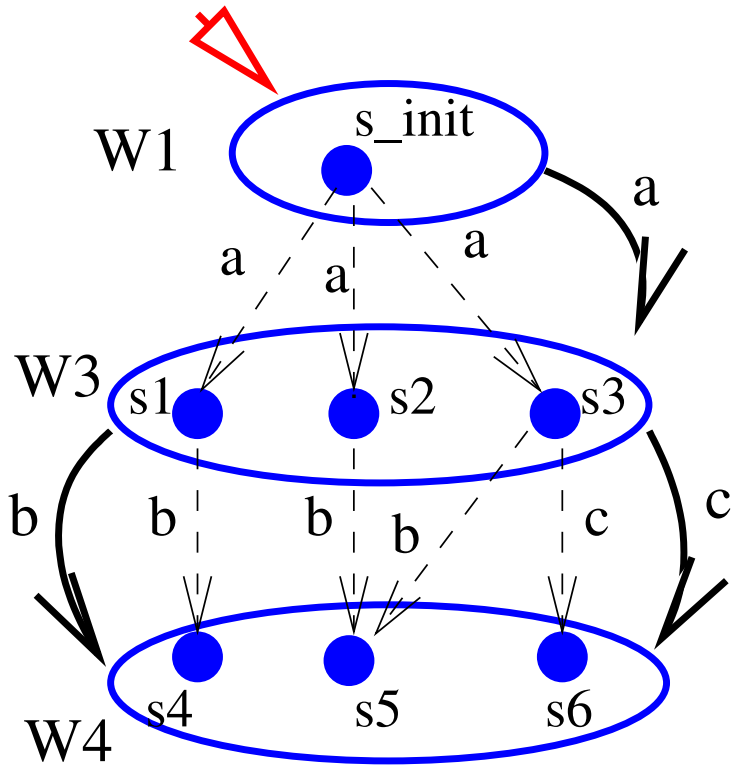
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

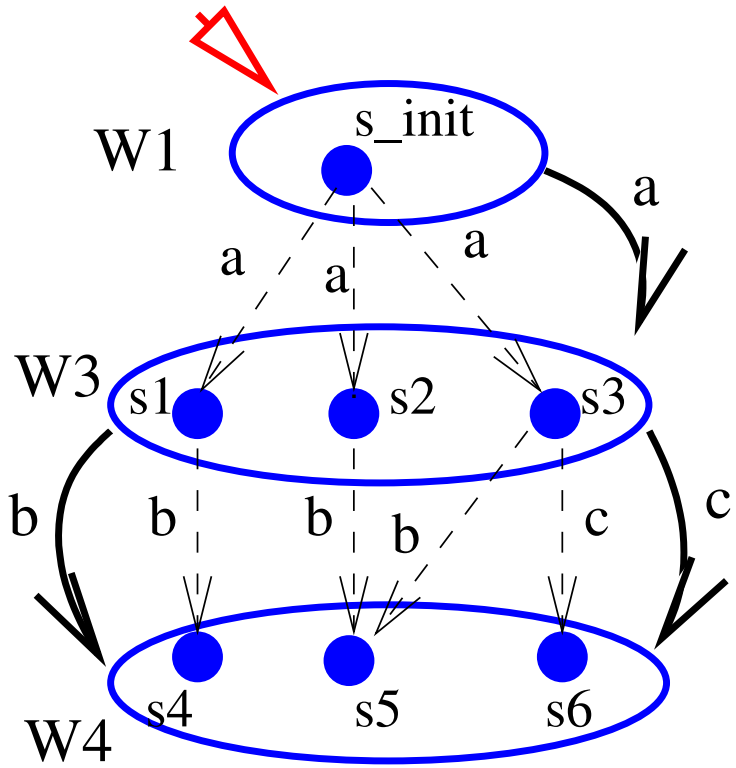
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

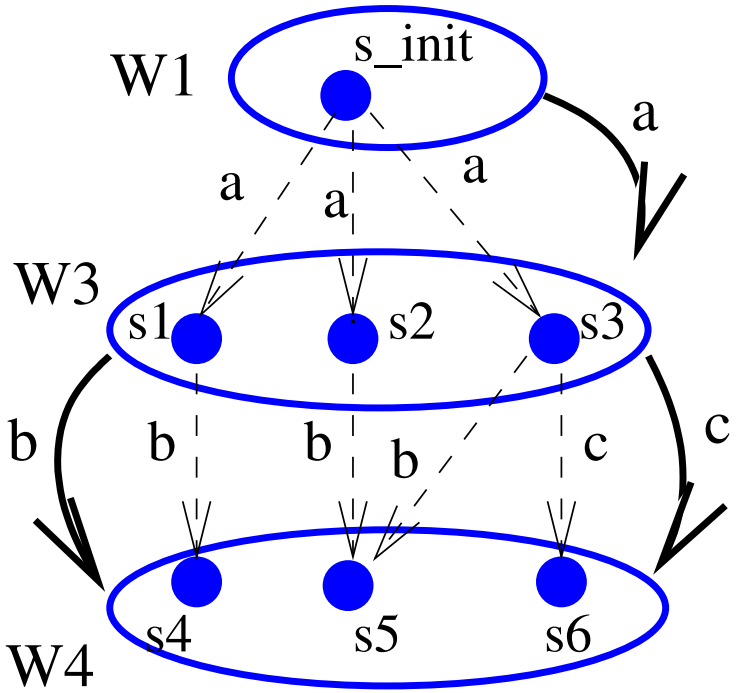
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

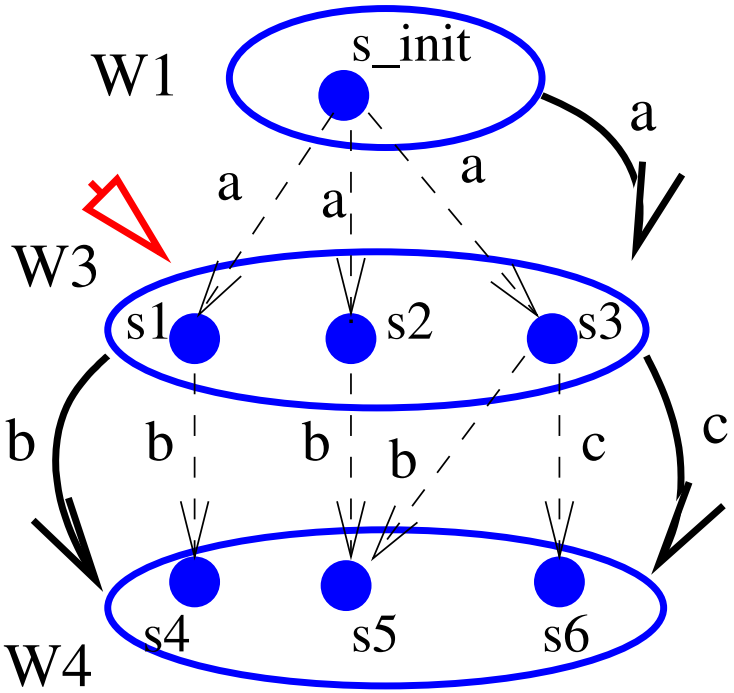
Algorytm – przykład



reachable=$\{W_1, W_3\}$,
stable=$\{W_1\}$

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

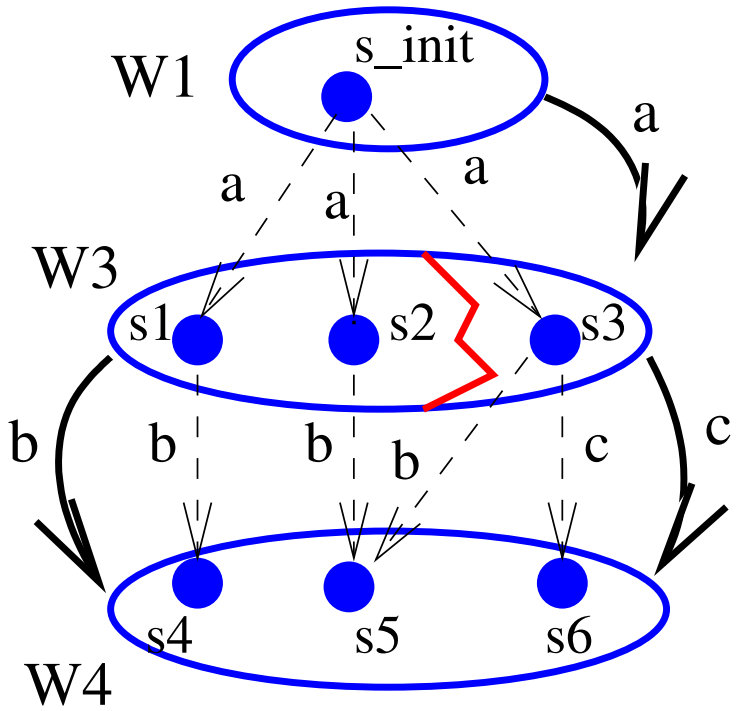
Algorytm – przykład



reachable={W_1, W_3},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
      6. **else**
         7. P := {W};
         8. reachable := reachable $\setminus$ P
            $\cup$ {W' $\in$ C | s_{init} $\in$ W'};
         9. stable := stable $\setminus$ Pre_{Π}(P);
         10. Π := (Π $\setminus$ P) $\cup$ C;
   11. **end if**;
12. **end do**;

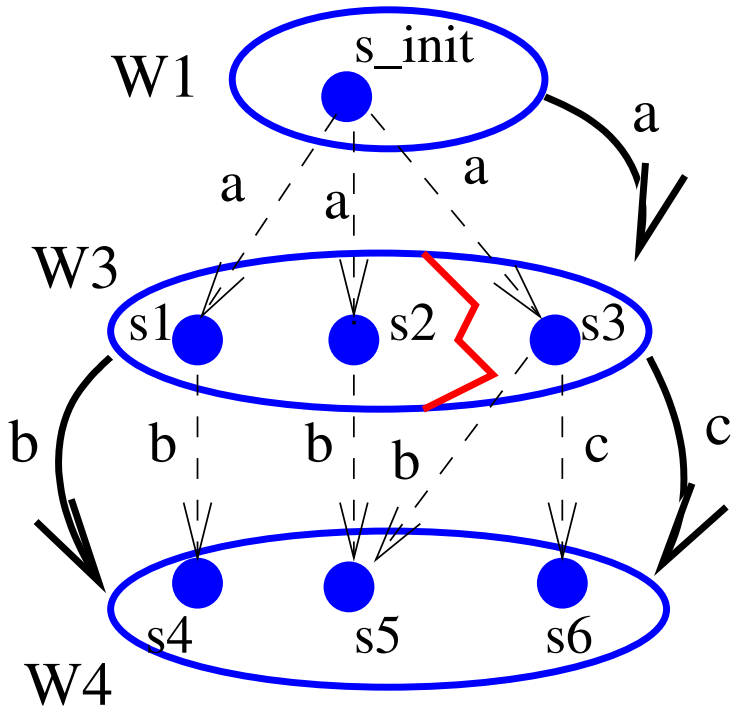
Algorytm – przykład



reachable={W_1, W_3},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := $Split_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         $Post_\Pi$(W);
      6. **else**
         7. P := {W};
         8. reachable := reachable $\setminus$ P
            $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
         9. stable := stable $\setminus$ Pre_Π(P);
         10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

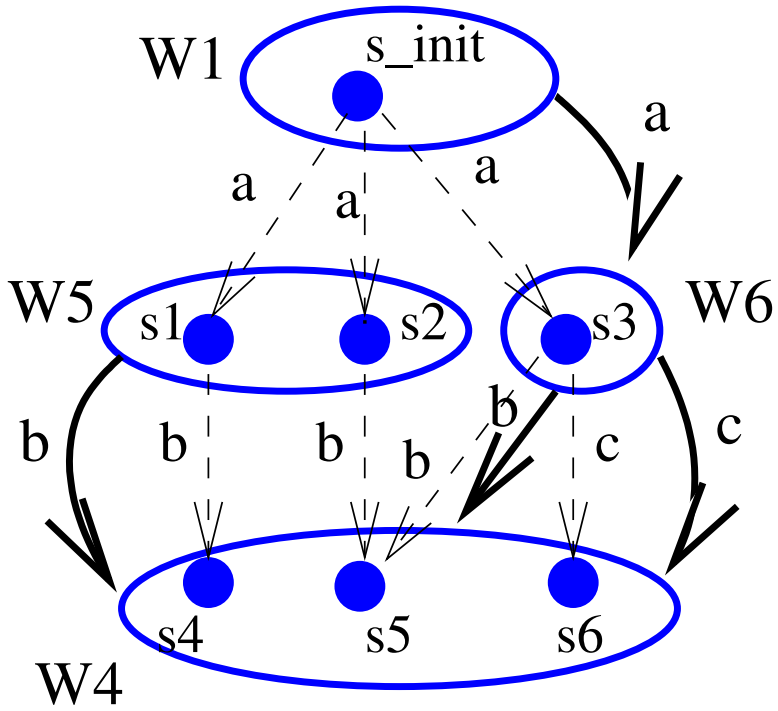
Algorytm – przykład



reachable={W_1, W_3},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

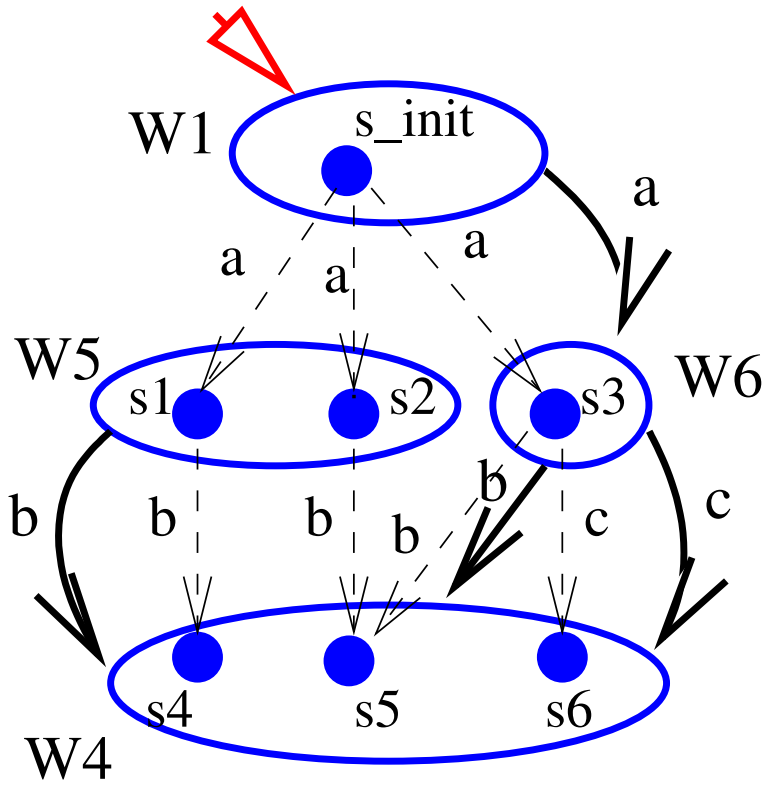
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

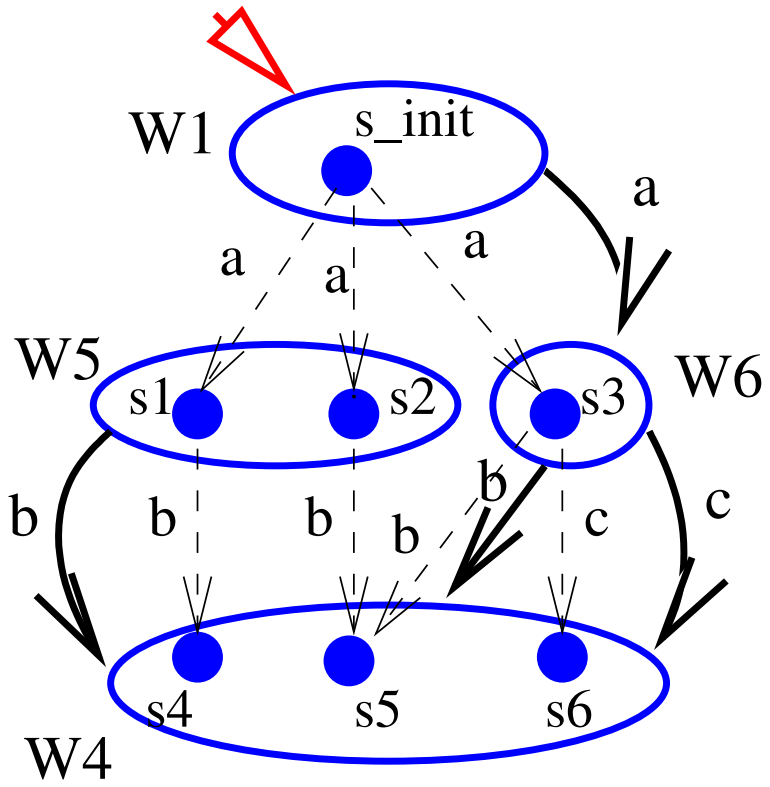
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

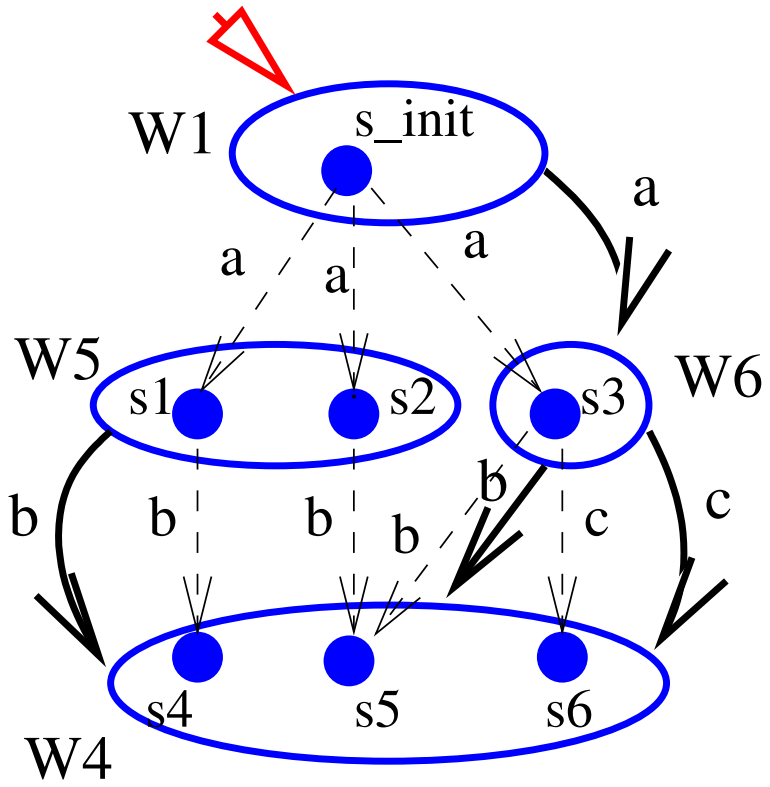
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_Π(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_Π(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

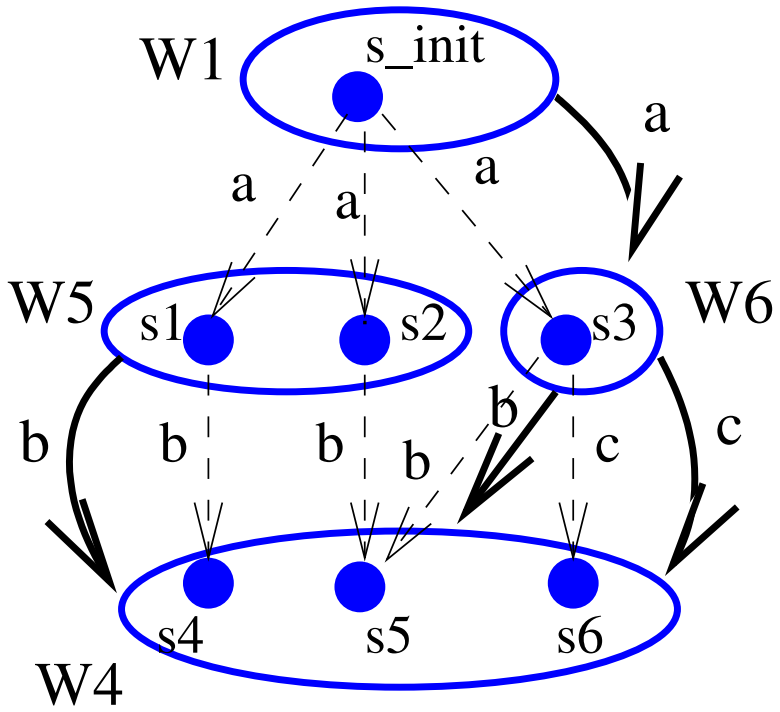
Algorytm – przykład



reachable=$\{W_1\}$, stable=ϕ

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

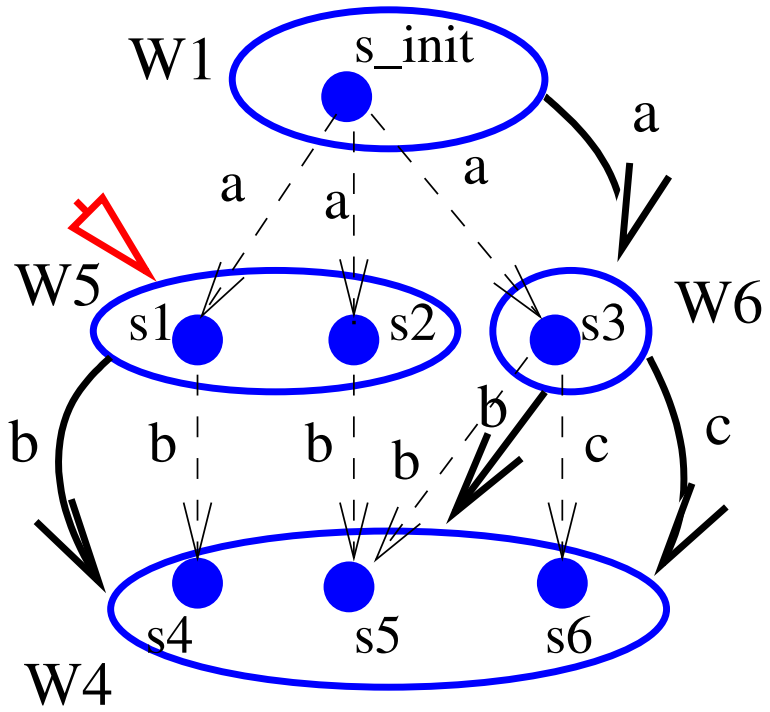
Algorytm – przykład



reachable={W_1, W_5, W_6},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

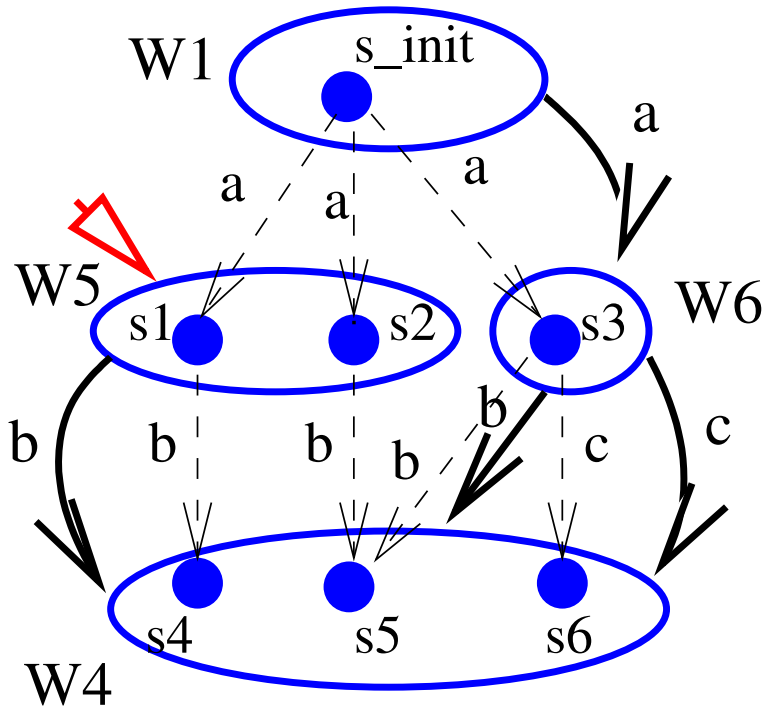
Algorytm – przykład



reachable={W_1, W_5, W_6},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

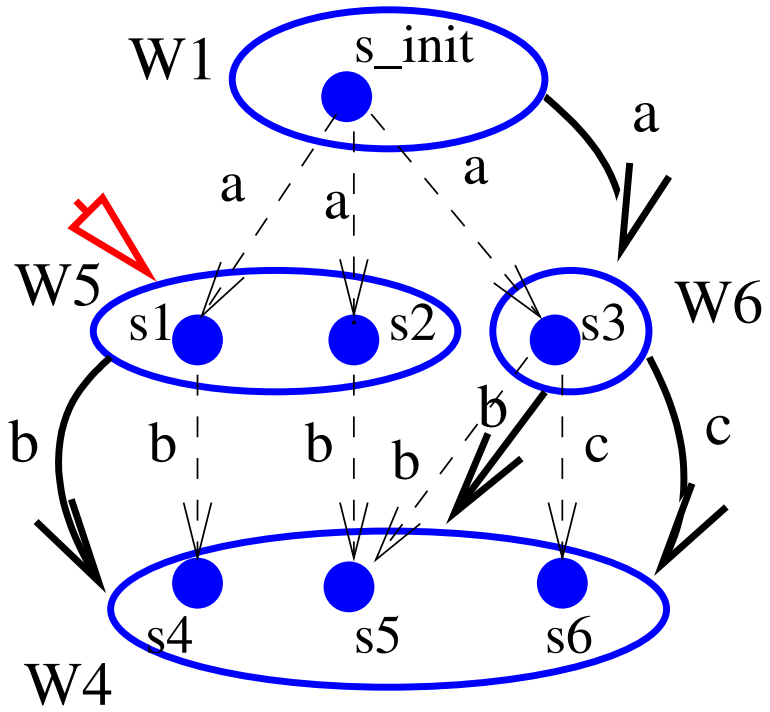
Algorytm – przykład



reachable={W_1, W_5, W_6},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := $Split_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         $Post_{\Pi}$(W);
      6. **else**
         7. P := {W};
         8. reachable := reachable $\setminus$ P
            $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
         9. stable := stable $\setminus$ Pre_{Π}(P);
         10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

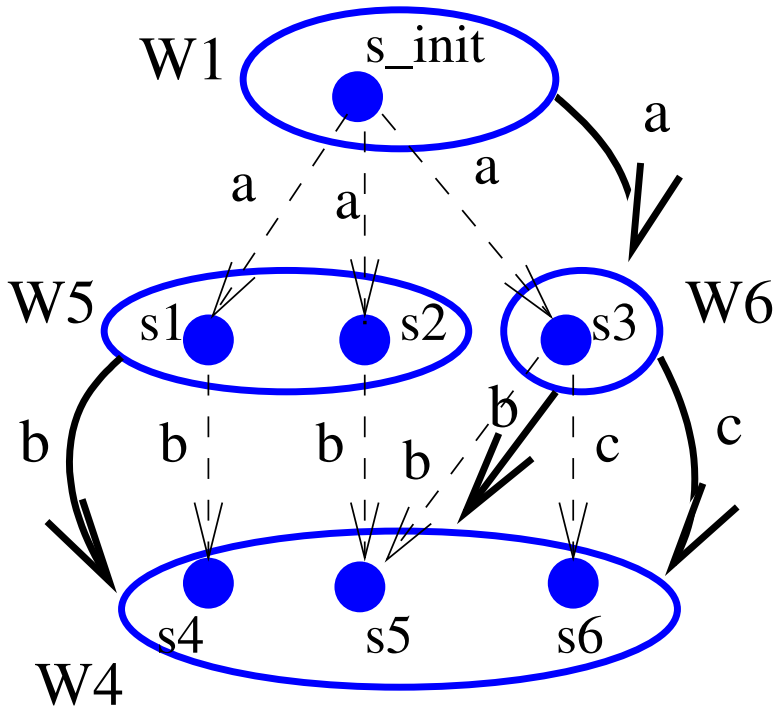
Algorytm – przykład



reachable={W_1, W_5, W_6},
stable={W_1}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split$_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post$_\Pi$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre$_\Pi$(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

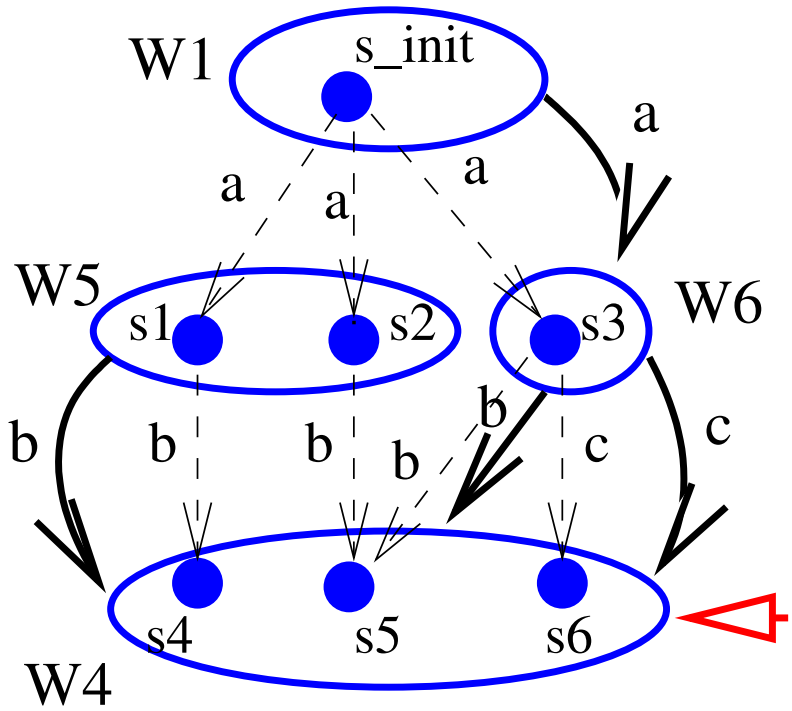
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_5}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split$_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post$_\Pi$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre$_\Pi$(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

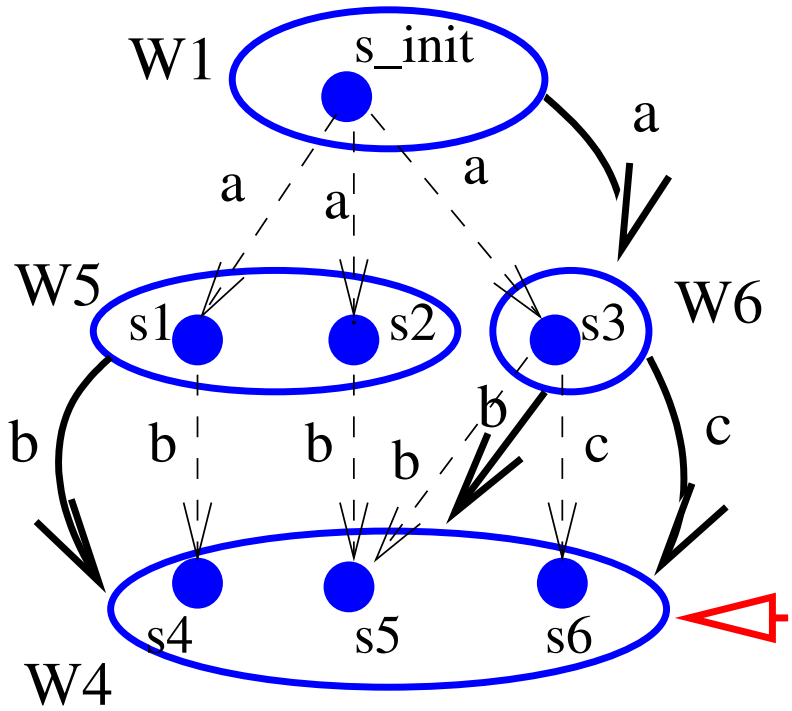
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_5}

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

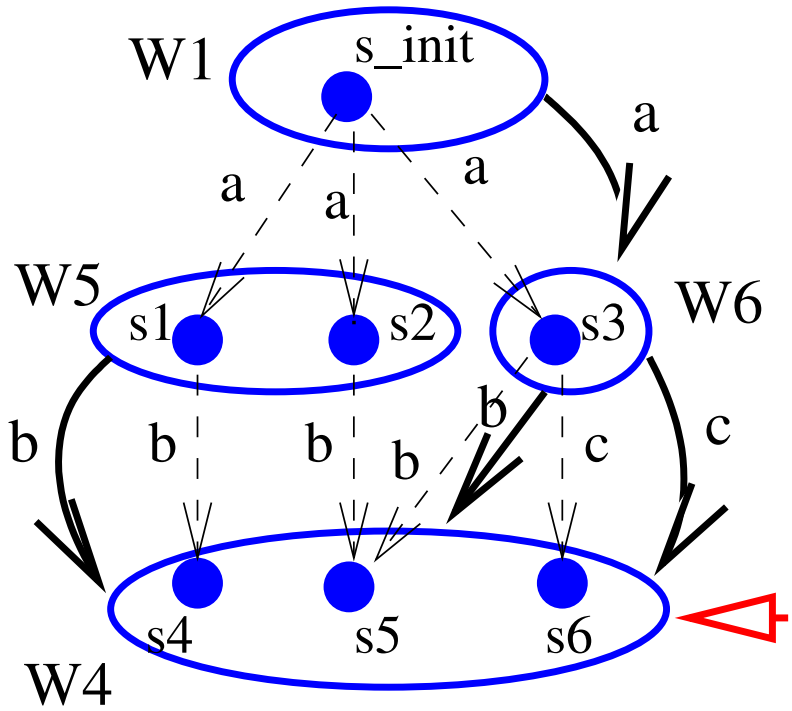
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_5}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := $Split_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         $Post_{\Pi}$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

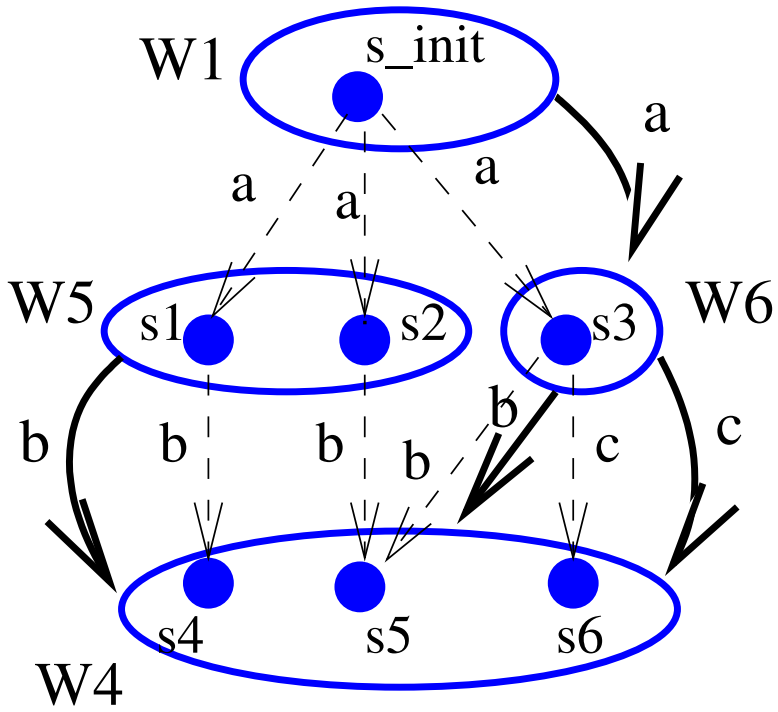
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_5}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split$_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post$_\Pi$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre$_\Pi$(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

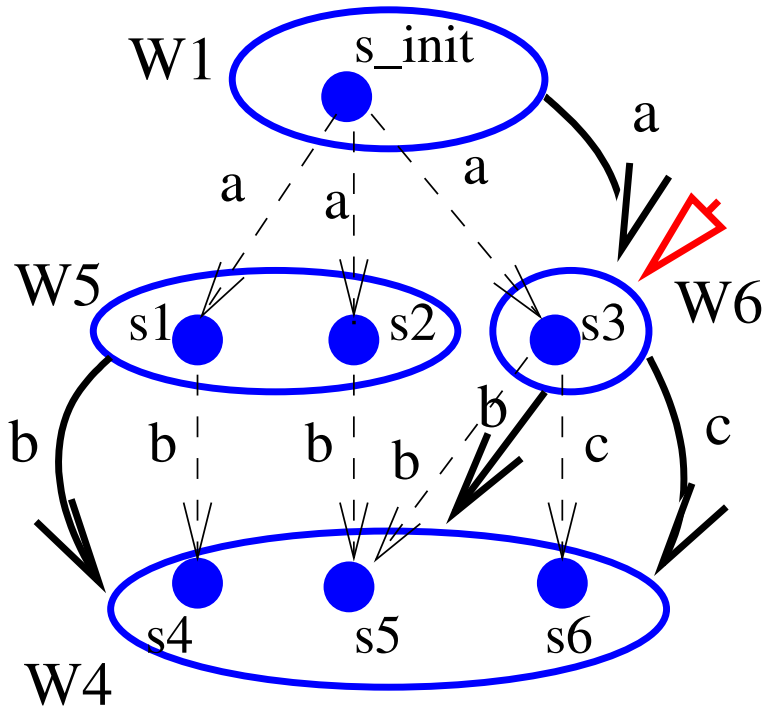
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

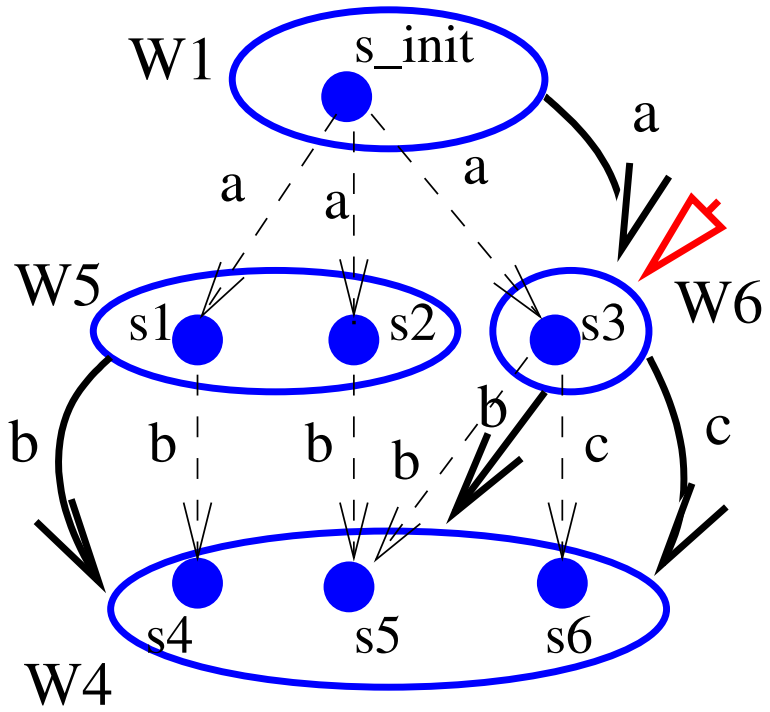
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5}

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

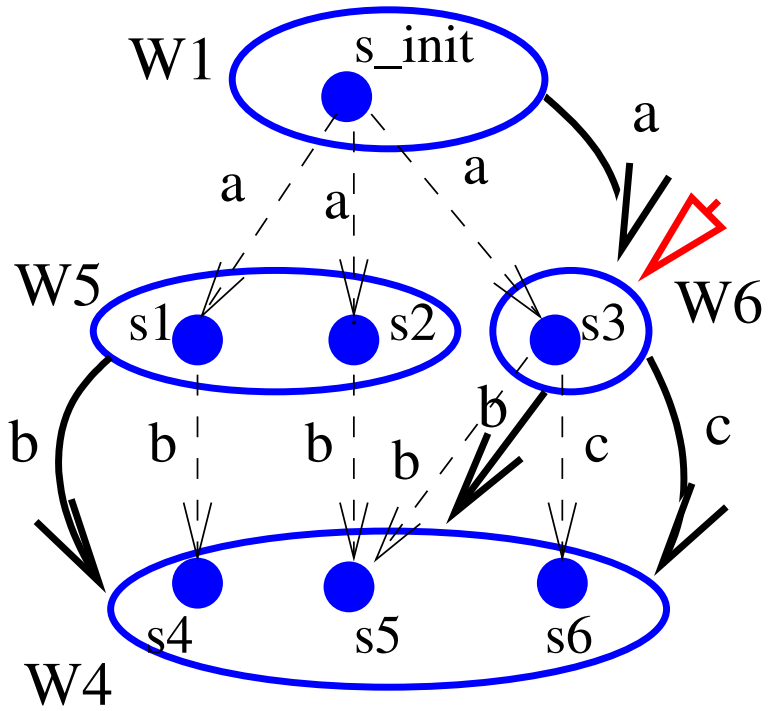
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5}

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := $Split_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ $Post_{\Pi}$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

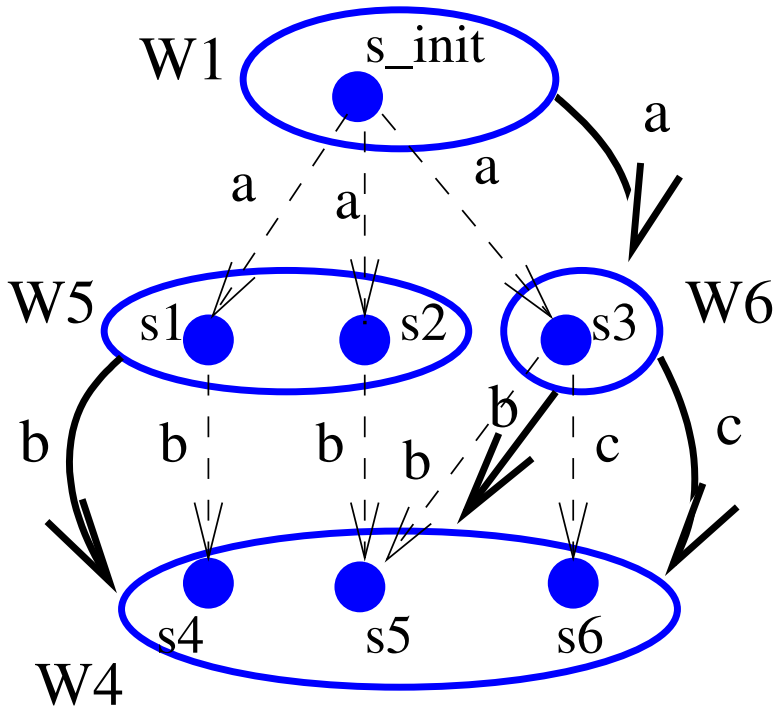
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5}

1. $\Pi := \Pi_0$; reachable := $\{[s_{init}]\}$; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_Π(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_Π(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

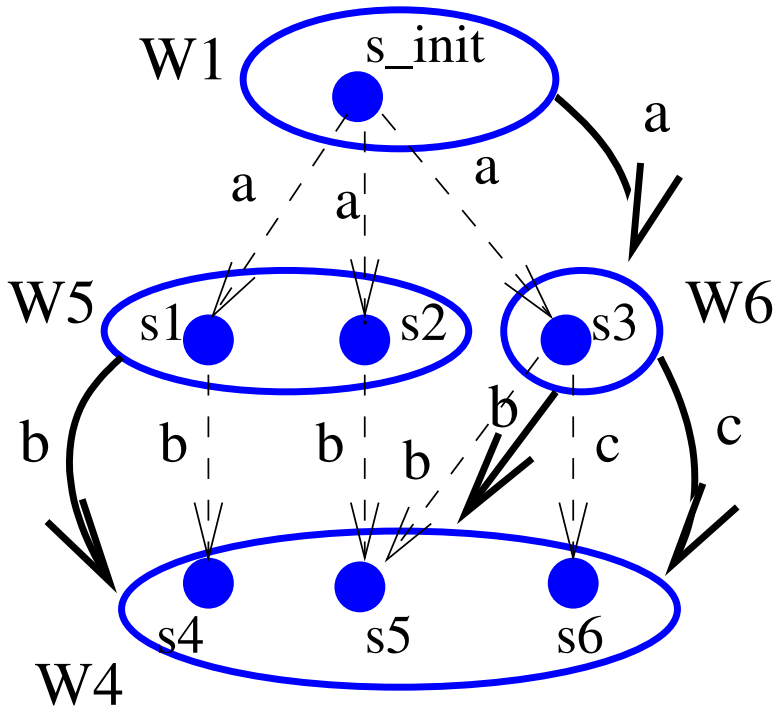
Algorytm – przykład



reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5, W_6}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]};
   stable := ϕ;
2. **while**
   ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split$_{bi}$(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W};
         reachable := reachable $\cup$
         Post$_\Pi$(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P
         $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre$_\Pi$(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Algorytm – przykład

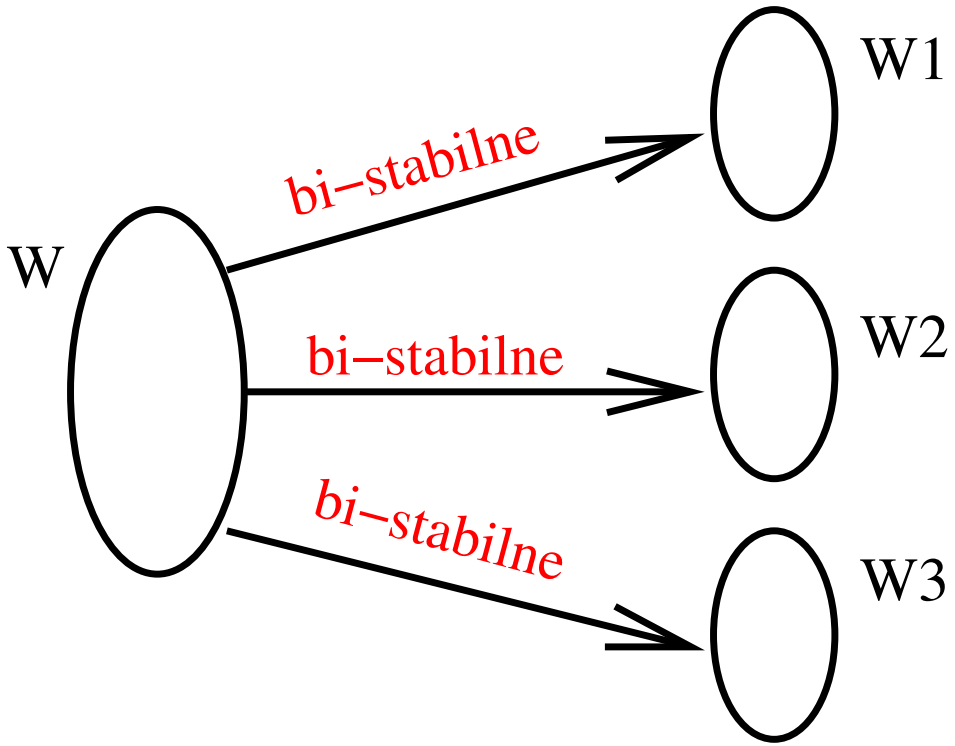


reachable={W_1,W_4,W_5,W_6},
stable={W_1, W_4, W_5, W_6}

1. $\Pi := \Pi_0$; reachable := {[s_{init}]}; stable := ϕ;
2. **while** ($\exists$ W $\in$ reachable $\setminus$ stable) **do**
   3. C := Split_{bi}(W, Π);
   4. **if** (C = {W}) **then**
      5. stable := stable $\cup$ {W}; reachable := reachable $\cup$ Post_{Π}(W);
   6. **else**
      7. P := {W};
      8. reachable := reachable $\setminus$ P $\cup$ {W' $\in$ C | $s_{init} \in$ W'};
      9. stable := stable $\setminus$ Pre_{Π}(P);
      10. $\Pi := (\Pi \setminus P) \cup C$;
   11. **end if**;
12. **end do**;

Usprawnienia algorytmu

Znakowanie bi-stabilności



- dla klasy W pamiętamy względem których W' jest bi-stabilne
- wykorzystujemy to w kolejnych przebiegach
- informacja musi być aktualizowana w przypadku podziału któregoś następnika

Usprawnienia algorytmu

Podział w locie

- odkładamy generowanie klas, aż będą one nam rzeczywiście potrzebne
- unikamy generowania klas nieosiągalnych

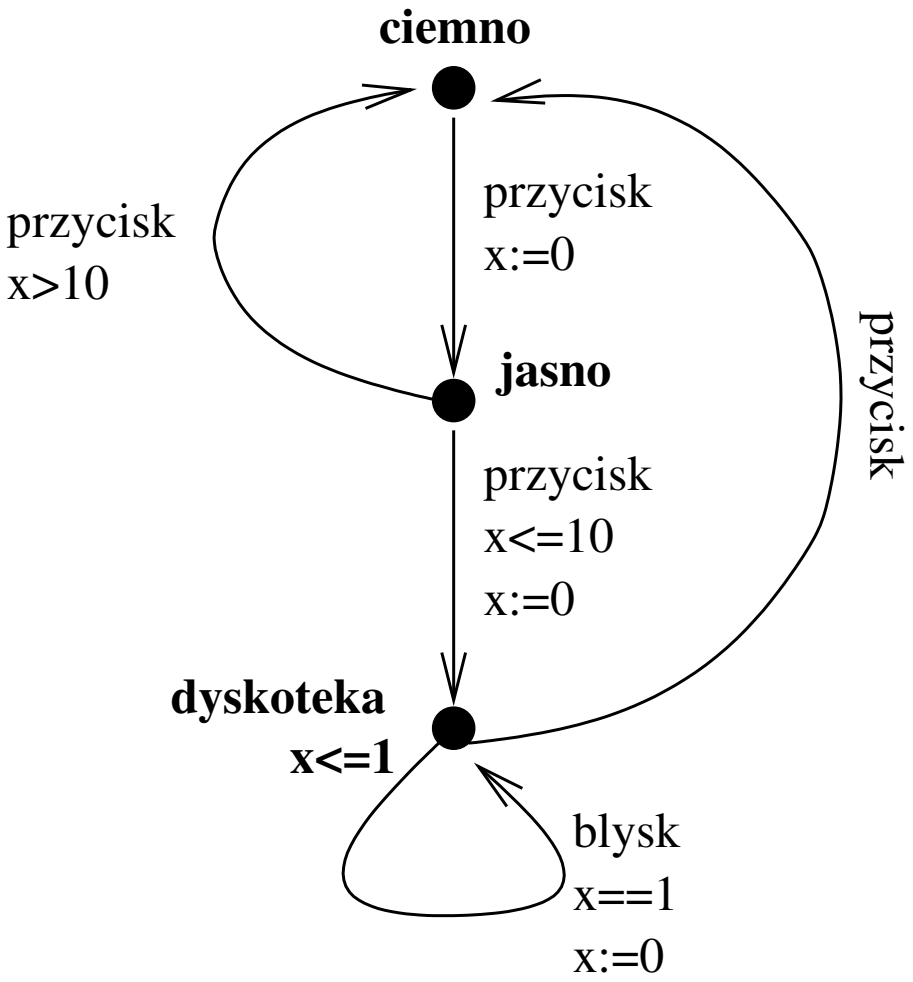
Algorytm dla automatów czasowych

Trzeba zdefiniować:

- stany konkretne i abstrakcyjne
- podział początkowy Π_0
- funkcje pre i post

Automat czasowy

Lampa



- **lokacje** – np. ciemno, jasno, dyskoteka
- **akcje** – np. przycisk, blysk
- **warunki umożliwienia** – np. x<=10, x>10, x<=1, x==1
- **resetowanie zegarów** – np. x:=0
- **niezmienniki lokacji** – np. x<=1

Automat konkretny

System tranzycyjny

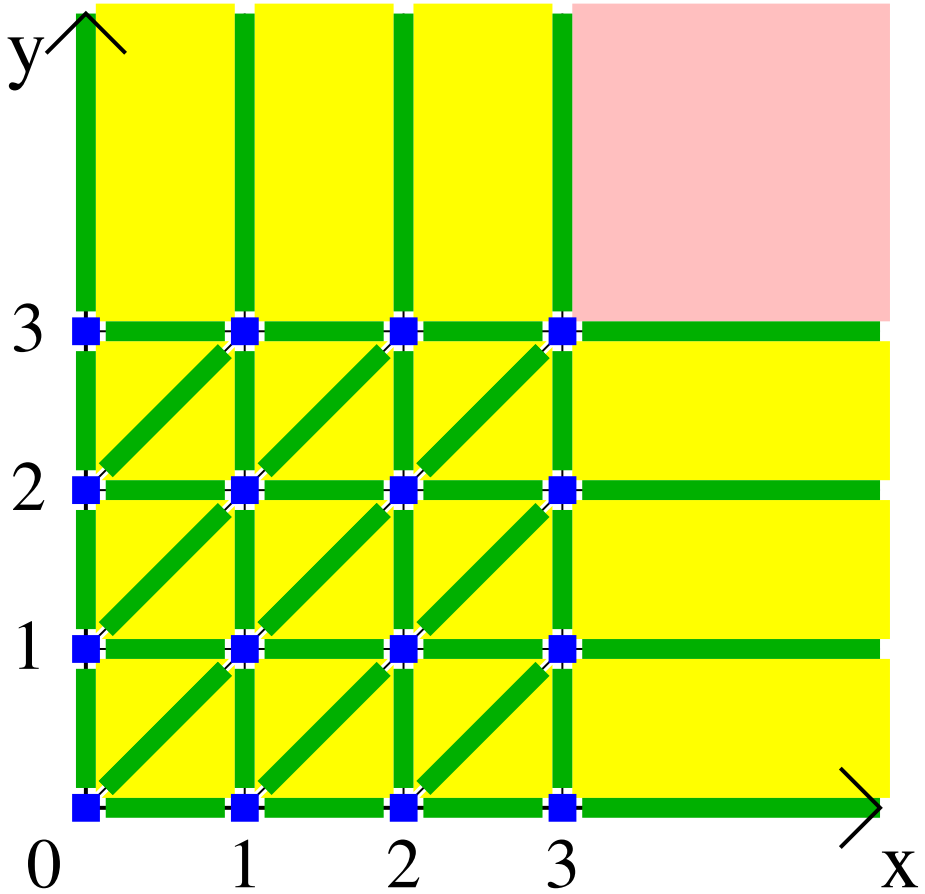
- **stany** to pary (lokacja, wartościowanie zagarów) jest ich continuum
- **przejścia czasowe** – upływ czasu zwiększa wartościowania wszystkich zegarów o daną liczbę rzeczywistą, lokacja pozostaje bez zmian
- **przejścia zwykłe** – możliwe o ile prawdziwy jest warunek umożliwienia, można podać zbiór zegarów, który będzie resetowany przez akcję
- **niezmiennik** – to warunek który możemy przyporządkować lokacji – jeśli automat jest w danej lokacji, (z definicji) jest spełniony odpowiedni niezmiennik

Automat abstrakcyjny

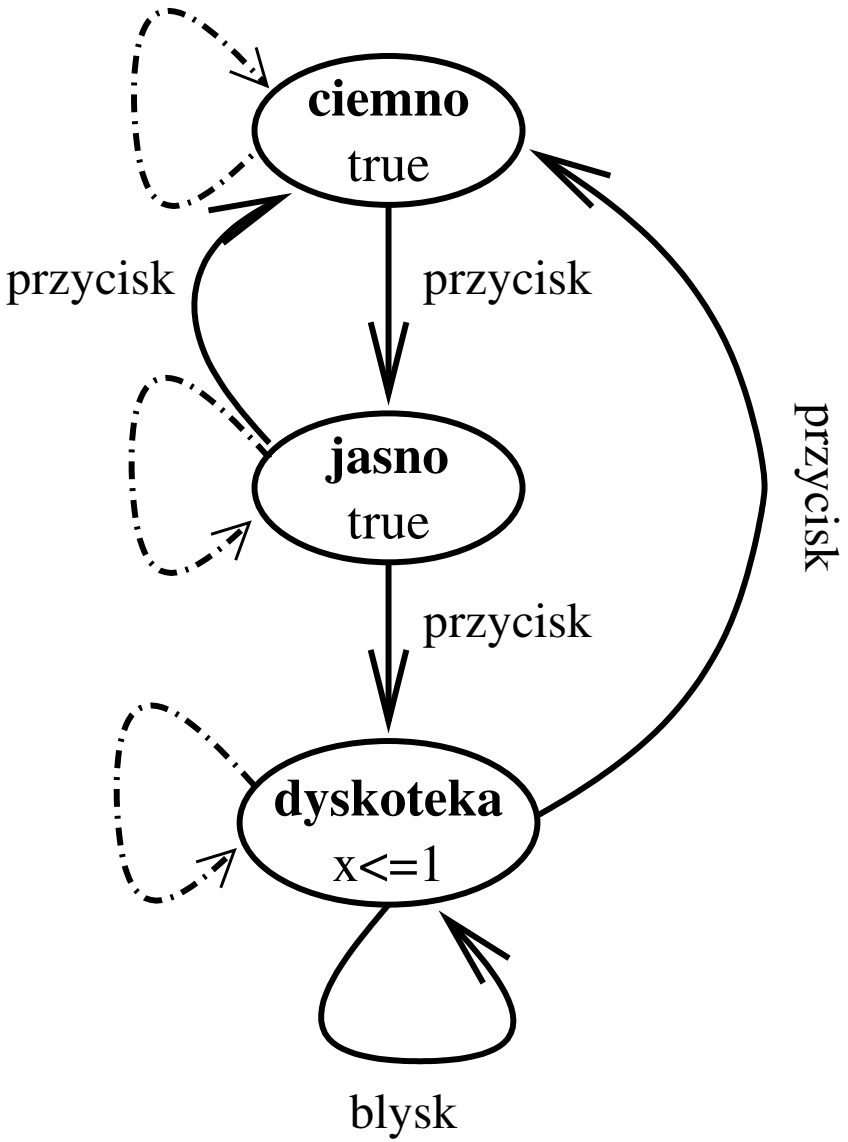
- **stany** – to pary (lokacja, zbiór wartościowań zegarów)
- **relacja przejścia** indukowana z przejść w systemie tranzycyjnym

Automat regionów

- utożsamiamy wartościowania nierozróżnialne ze względu na wykonalność dalszych przejść
- skończona ilość klas abstrakcji
- podział zależny od maksymalnej stałej występującej w definicji automatu



Algorytm dla automatów czasowych



Podział początkowy

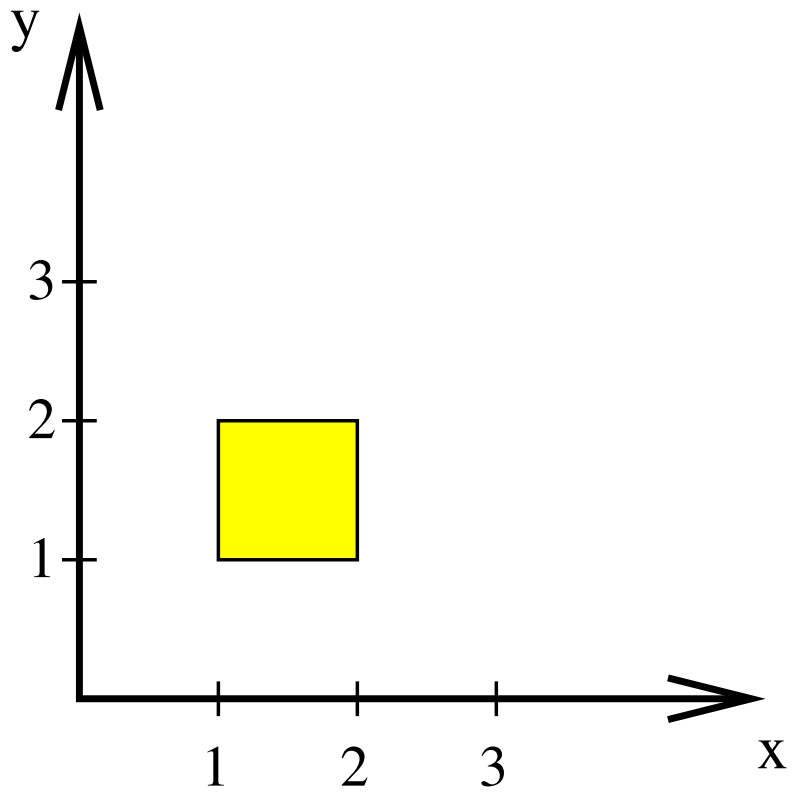
$\Pi_0 = \{(l, [[\mathcal{I}(l)]]) \mid l \in \text{Lokacje}\}$

- Π_0 jest podziałem jedynie pewnej części przestrzeni stanów, zawierającej stany osiągalne

Strefy czasowe

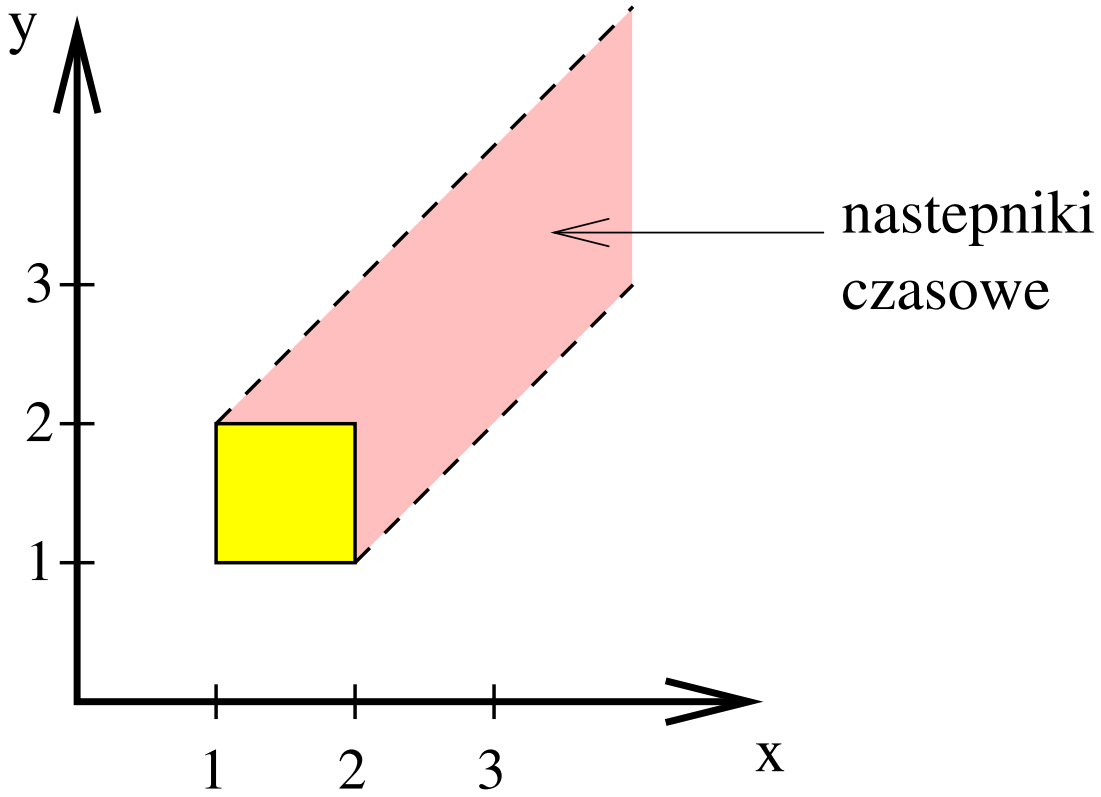
Strefa

- zbiór wartościowań zegarów
- zbiór zadany w postaci koniunkcji warunków
- każdy warunek ogranicza
  - z góry lub z dołu
  - ostro lub nieostro
  - ogranicza zegar lub róznicę dwóch zegarów
- strefa jest zbiorem wypukłym



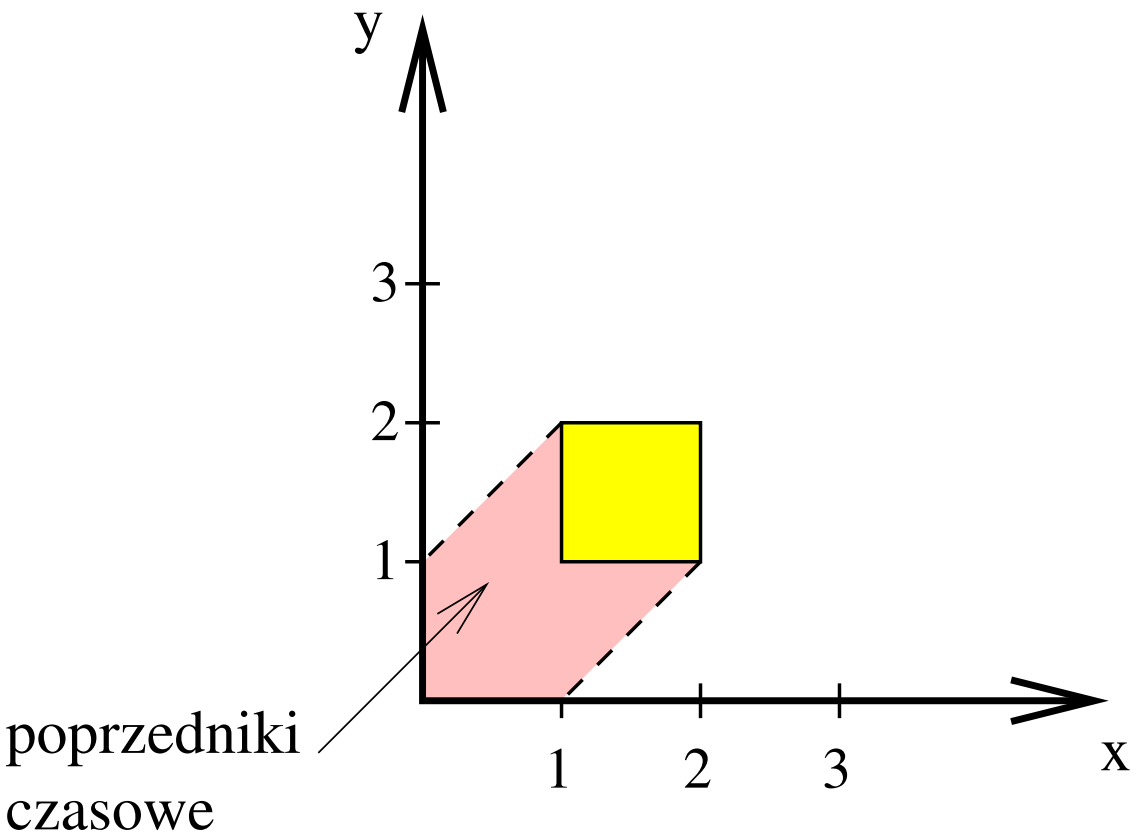
Następstwo czasowe

Następniki czasowe strefy

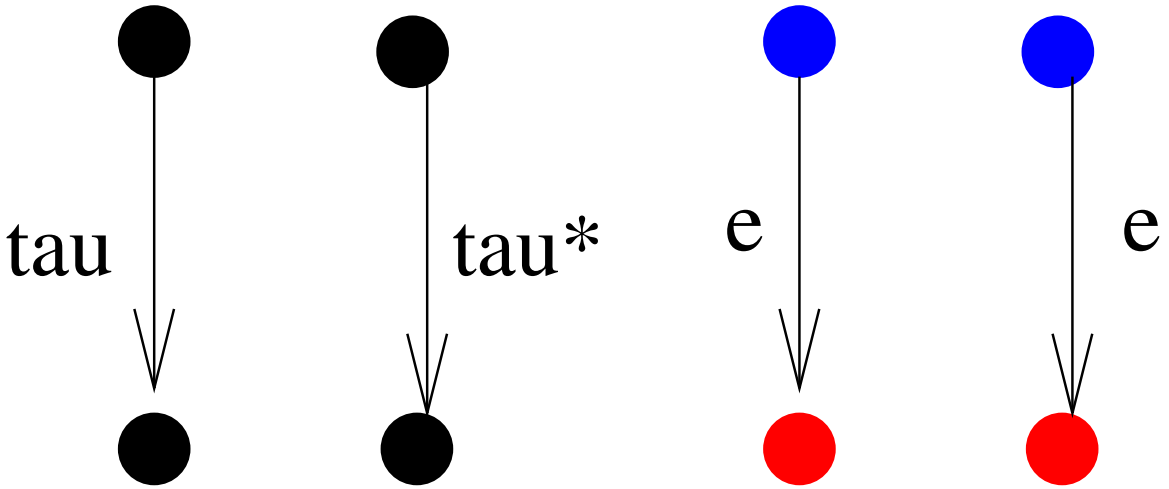


Poprzedniki czasowe

Poprzedniki czasowe strefy



Bisymulacja widoczna



Poprzedniki, następniki

Następstwo akcyjne e = l $\longrightarrow^{a,cc,X}$ l'

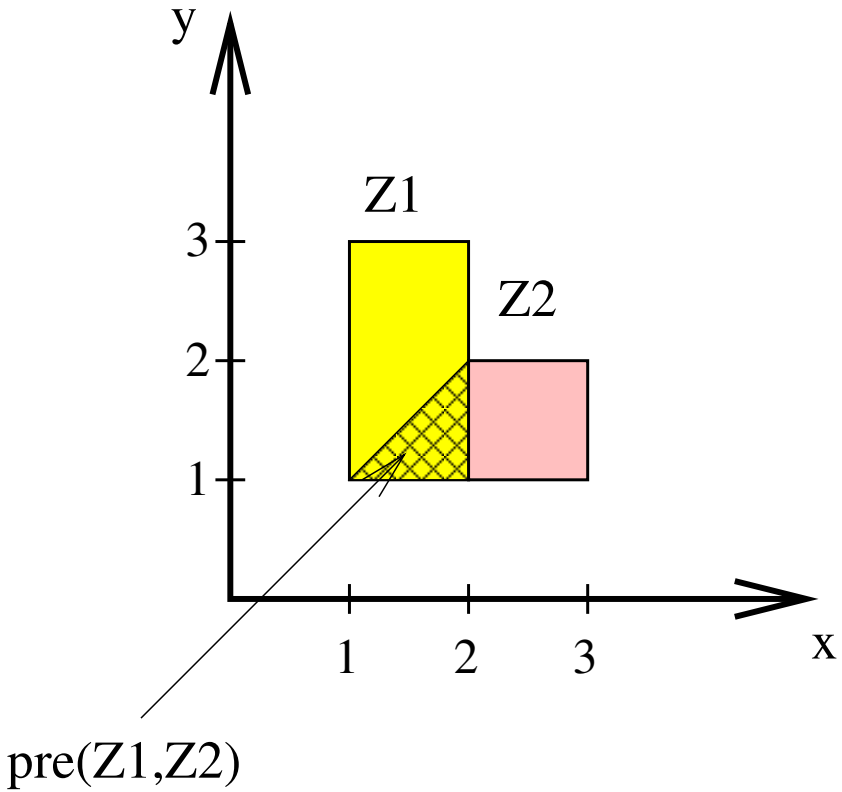
- $pre_e((l,Z), (l',Z')) = (l, Z \cap (([X:=0]Z') \cap [[cc]]))$
- $post_e((l,Z), (l',Z')) = (l', (Z \cap [[cc]])[X:=0] \cap Z')$

Następstwo czasowe

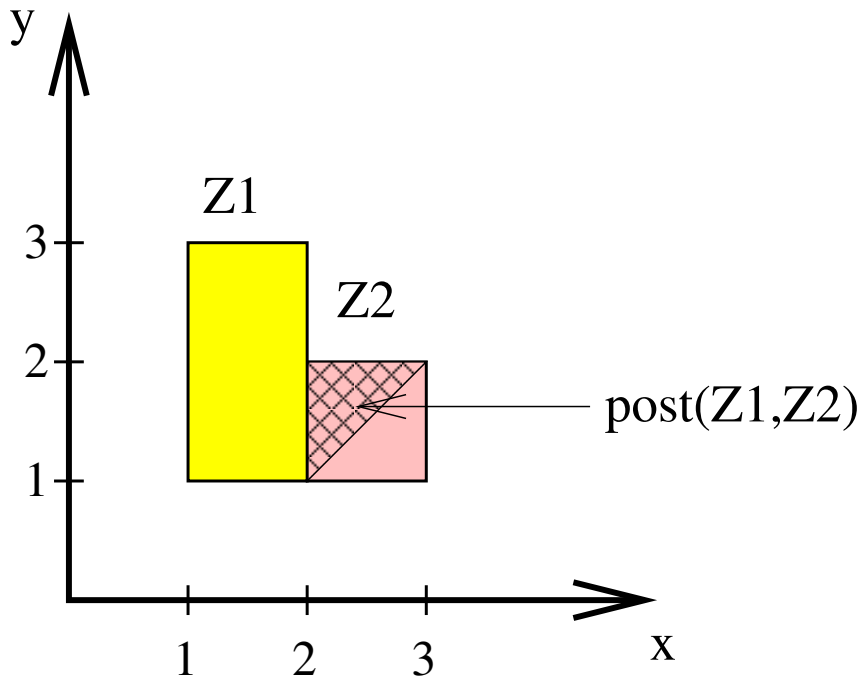
- $pre_\tau((l,Z), (l,Z')) = (l, Z \Uparrow Z')$
- $post_\tau((l,Z), (l,Z')) = (l, (Z \Uparrow Z')^{\nearrow} \cap Z')$

Następstwo czasowe

$pre_\tau((l,Z),(l,Z')) = (l, Z \Uparrow Z')$

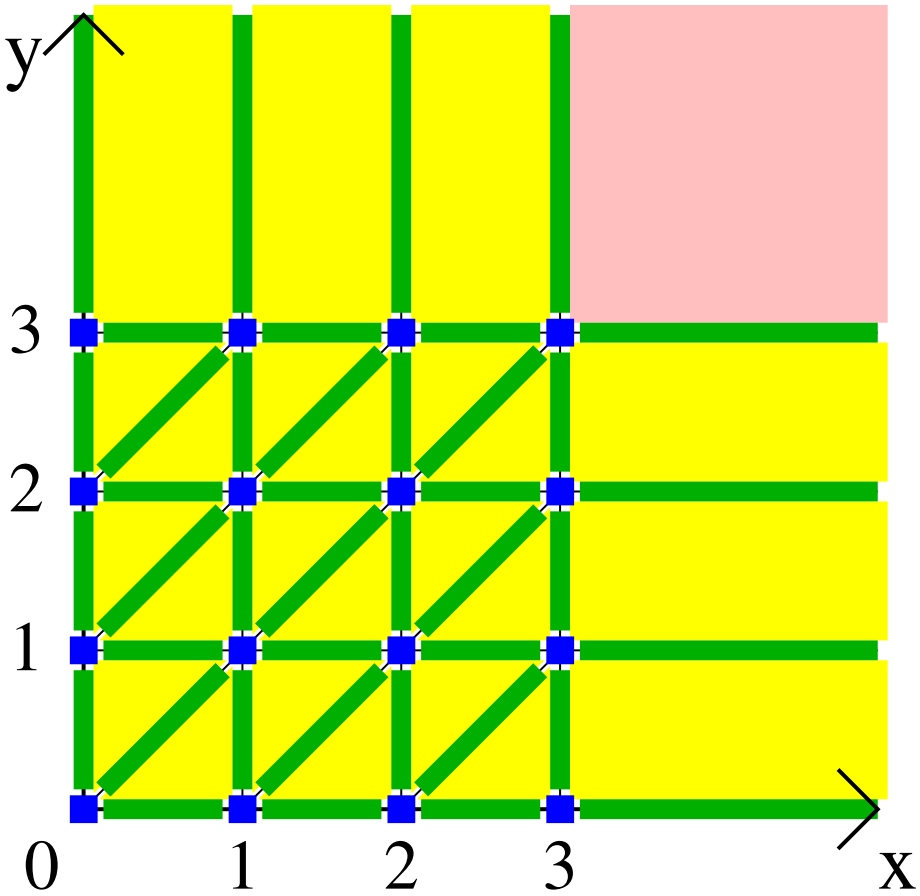


$post_\tau((l,Z),(l,Z'))$
$= (l, (Z \Uparrow Z') \nearrow \bigcap Z')$



Warunek stopu

W najgorszym przypadku dojdziemy do automatu regionów.



Implementacja stref

Strefa

Strefę można opisać zbiorem warunków

$\{ \quad \mathbf{x}_i - \mathbf{x}_j \sim \mathbf{c} \quad \mid x_i, x_j \in \text{Clocks}, \sim \in \{<, \leq\}, c \in \mathbb{Z}\}$

- $x_i - x_j = c$ zapisujemy jako dwa warunki:
  1. $x_i - x_j \leq c$
  2. $x_j - x_i \leq -c$
- wprowadzamy specjalny zegar x_0 stale równy 0
- przykład: $x_1 < 2$ kodujemy jako $x_1 - x_0 < 2$
- przykład: $x_2 \geq 3$ kodujemy jako $x_0 - x_2 \leq -3$

Macierz ograniczników różnic

DBM

- macierz o **A** wymiarach (n + 1) $\times$ (n + 1), gdzie n to liczba zegarów
- kolumny i wiersze indeksowane liczbami 0..n
- A[i,j] = (a_{ij}, $\sim_{ij}$) oznacza $x_i - x_j \sim_{ij} a_{ij}$

Przykład

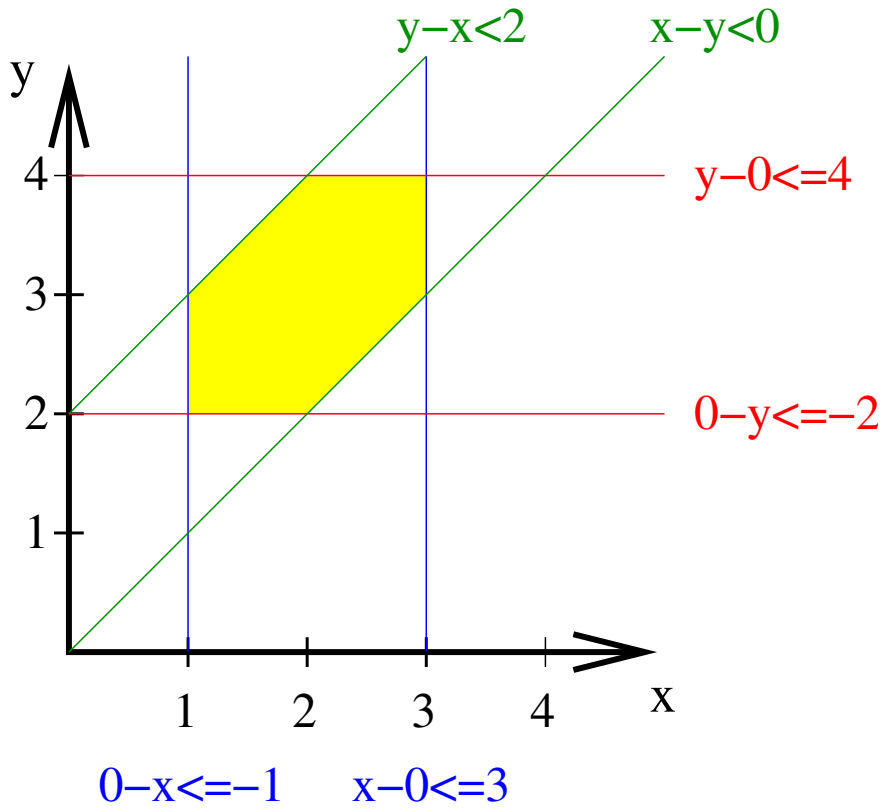
Zegary x, y (oraz 0).
A = $1 \leq x \leq 3$, $2 \leq y \leq 4$, $y > x$, $y < x + 2$

$$\mathbf{A} = \begin{pmatrix} (0, \leq) & (-1, \leq) & (-2, \leq) \\ (3, \leq) & (0, \leq) & (0, <) \\ (4, \leq) & (2, <) & (0, \leq) \end{pmatrix}$$

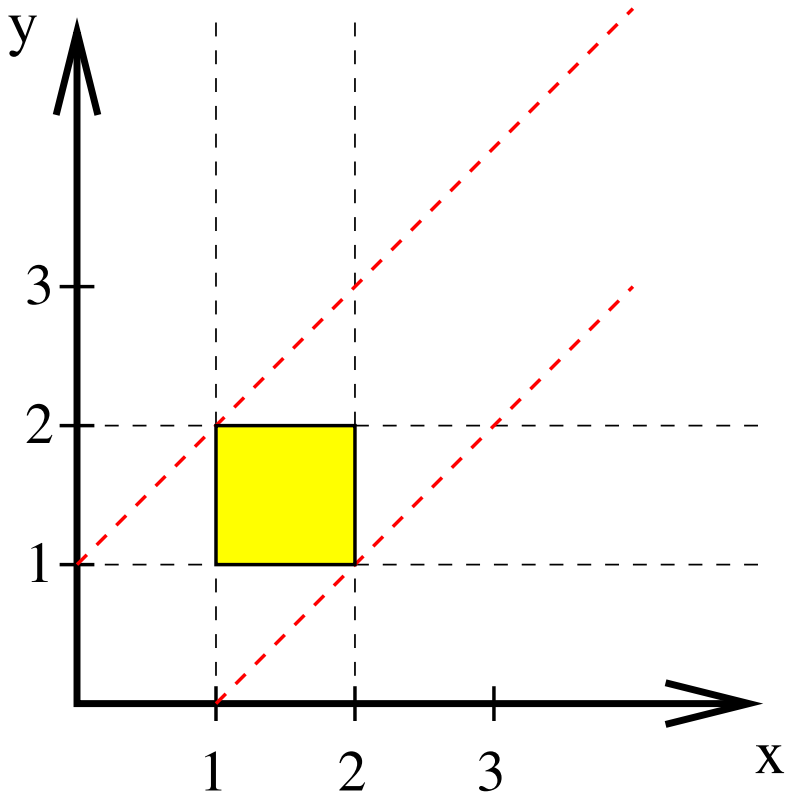
DBM – przykład

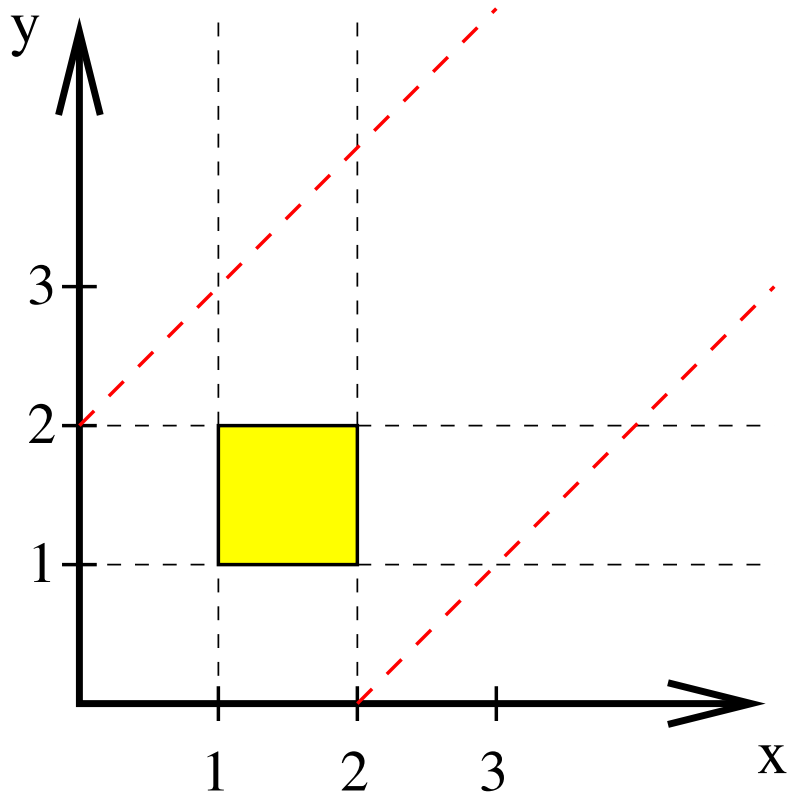
Zegary x, y (oraz 0).
A = $1 \leq x \leq 3$, $2 \leq y \leq 4$, $y > x$, $y < x + 2$

$$\mathbf{A} = \begin{pmatrix} (0,\leq) & (-1,\leq) & (-2,\leq) \\ (3,\leq) & (0,\leq) & (0,<) \\ (4,\leq) & (2,<) & (0,\leq) \end{pmatrix}$$

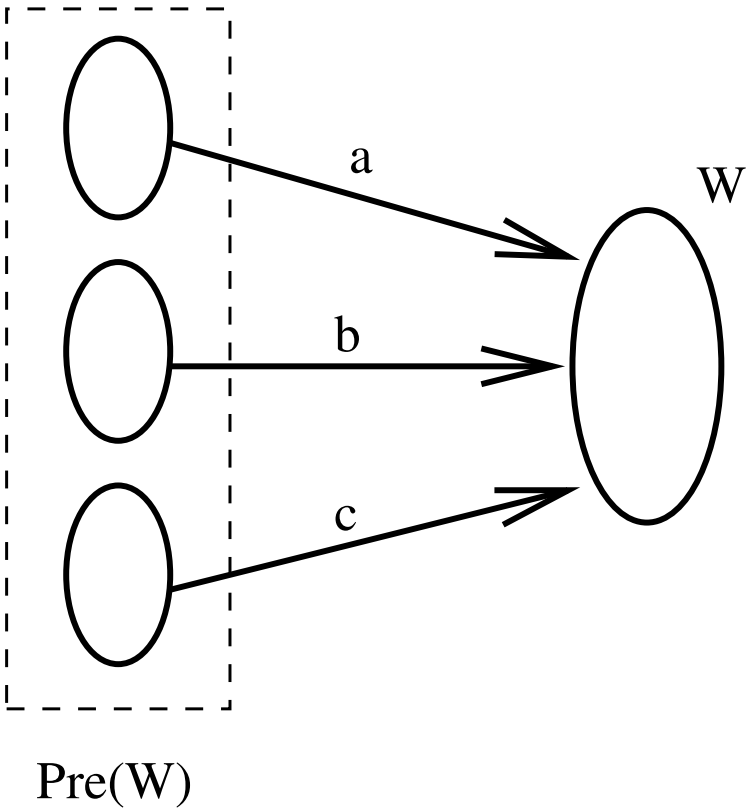


Niejednoznaczność reprezentacji strefy





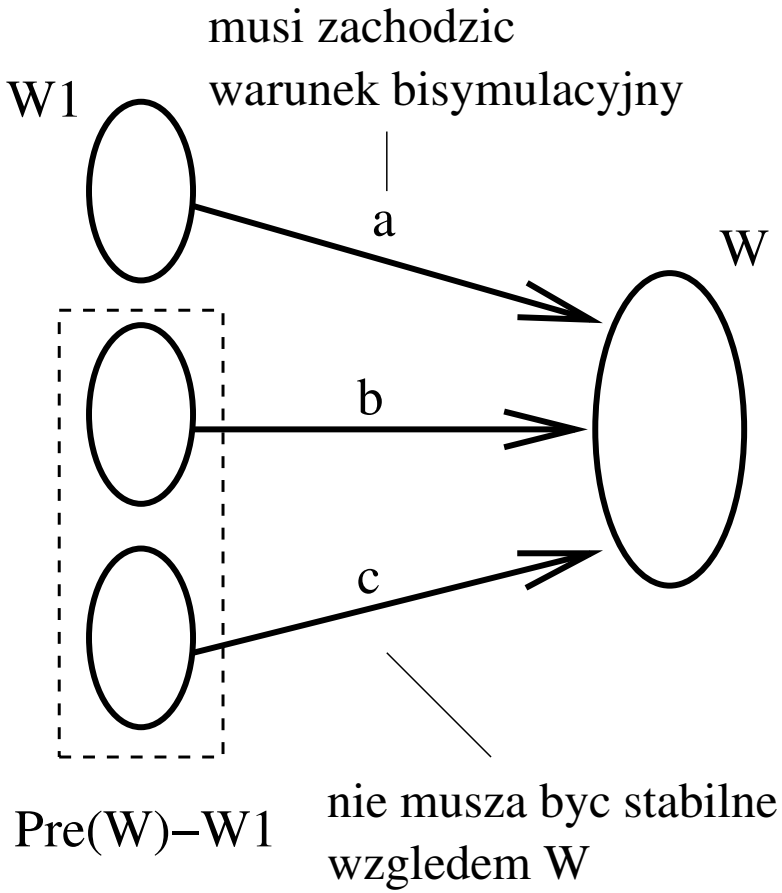
Modele pseudobisymulacyjne a bisymulacyjne



Model bisymulacyjny

- niech W - osiągalna klasa
- każdy z Pre(W) musi być stabilny względem W

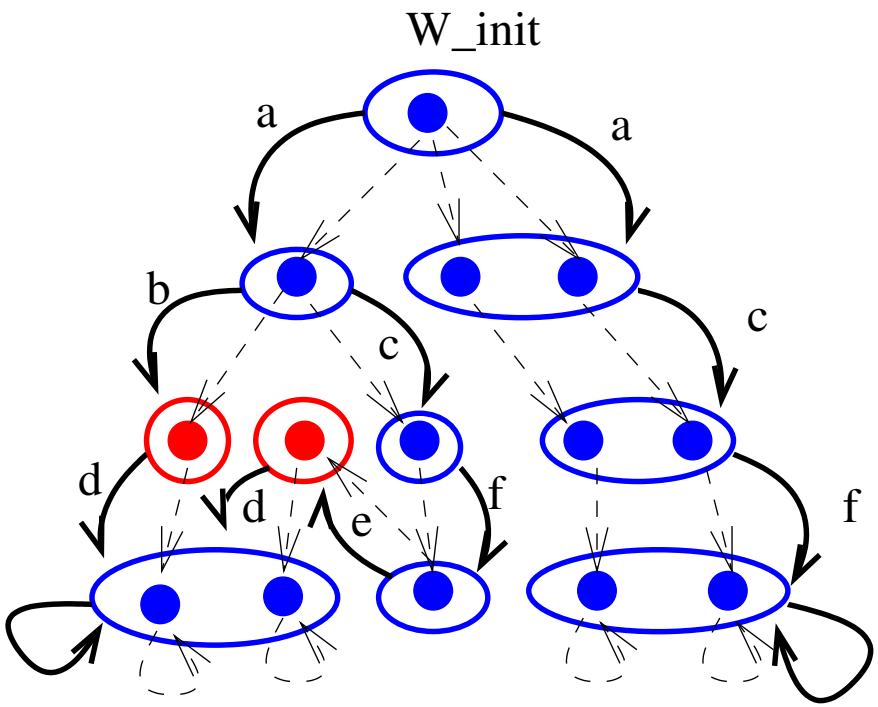
Modele pseudobisymulacyjne a bisymulacyjne



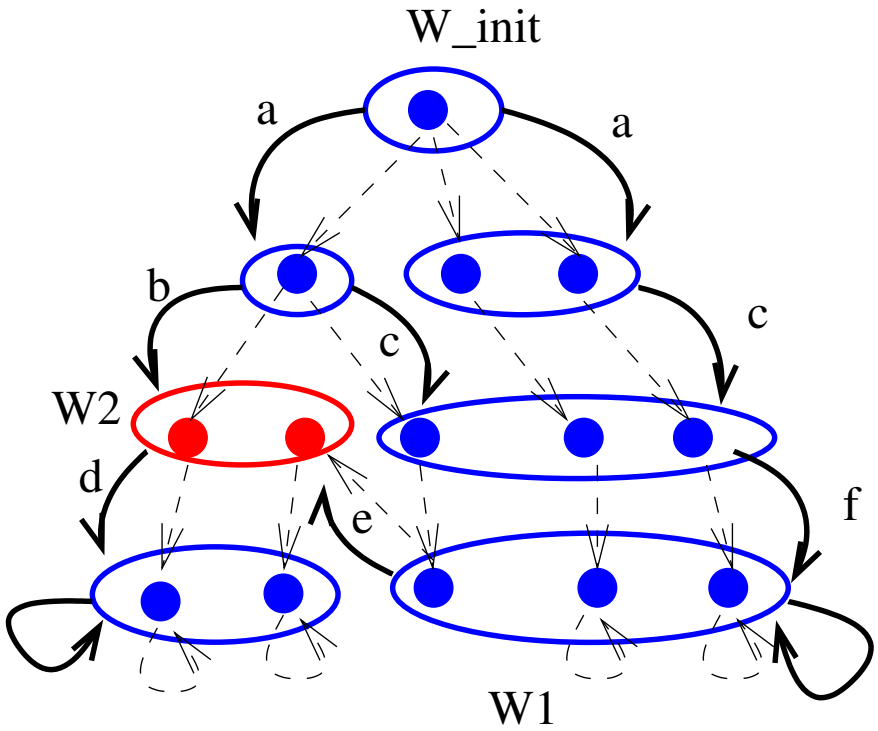
Model pseudobisymulacyjny

- niech W1 będzie węzłem osiągalnym z W_{init} w najmniejszej liczbie kroków
- warunek bisymulacyjny musi zachodzić jedynie między W1 a W
- dept(W) – głębokość W, długość najkrótszej ścieżki z W_{init} do W

Modele pseudobisymulacyjne



model bisymulacyjny



model pseudobisymulacyjny