Weryfikacja programów rekurencyjnych

Marcin Sulikowski

MIMUW

2 marca 2011

1. Programy rekurencyjne
2. Systemy ze stosem
3. Weryfikacja systemów ze stosem
4. Zastosowania
5. Narzędzia

Programy rekurencyjne

- Imperatywne
- Funkcje rekurencyjne
- Niedeterminizm
- Bez współbieżności
- Skończone dziedziny zmiennych

Stan programu rekurencyjnego

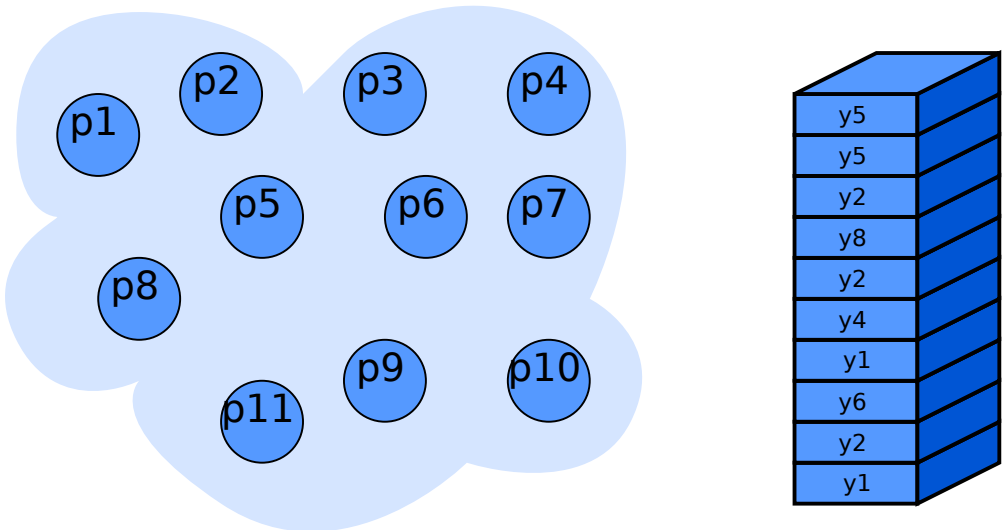
- Adres aktualnej instrukcji
- Zmienne globalne
- Zmienne lokalne aktualnej procedury
- Stos rekordów aktywacji (adresy powrotu, zmienne lokalne)

System ze stosem (PDS)

System ze stosem to krotka $\mathcal{P} = (P, \Gamma, \Delta)$

- P – zbiór stanów (punktów sterowania)
- Γ – alfabet stosowy
- $\Delta \subseteq (P \times \Gamma) \times (P \times \Gamma^*)$

Jeśli $(p, \gamma, p', w) \in \Delta$, to zapisujemy to $(p, \gamma) \hookrightarrow (p', w)$
Konfiguracją nazwiemy parę $\langle p, w \rangle$, gdzie $p \in P$, $w \in \Gamma^*$.



System ze stosem (PDS)

Interpretacja konfiguracji $\langle p, \gamma v \rangle$

- p – wartości zmiennych globalnych
- γ – aktualna instrukcja, wartości zmiennych lokalnych
- v – stos rekordów aktywacji

Interpretacja relacji Δ

- $(p, \gamma) \hookrightarrow (p', \gamma')$ – zwykła instrukcja
- $(p, \gamma) \hookrightarrow (p', \gamma'\gamma'')$ – wywołanie procedury
- $(p, \gamma) \hookrightarrow (p', \varepsilon)$ – powrót z procedury

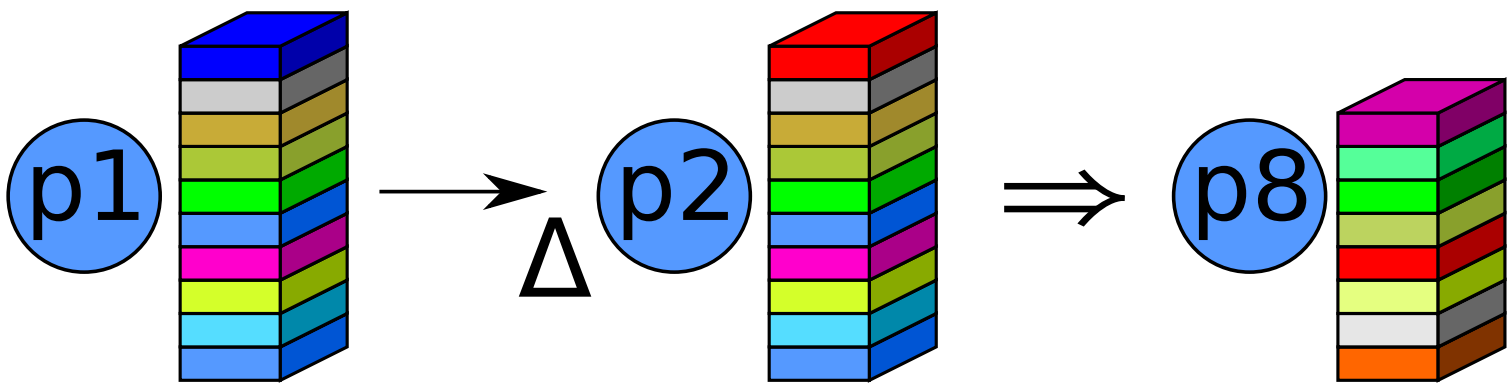
Weryfikacja programów rekurencyjnych

- LTL, CTL, CTL*
- Poprawność (asercje)

Weryfikacja systemów ze stosem

Osiągalność

- Relacja **bezpośredniego następnika** $\longrightarrow_\Delta$
  Jeśli $(p, \gamma) \hookrightarrow (p', w)$, to dla każdego $w' \in \Gamma^*$:
  $\langle p, \gamma w' \rangle \longrightarrow_\Delta \langle p', ww' \rangle$.
- **Relacja osiągalności** $\Rightarrow$
  Zwrotne i przechodnie domknięcie relacji $\longrightarrow_\Delta$



Weryfikacja systemów ze stosem

Osiągalność

- Dla danego zbioru konfiguracji C określamy funkcję pre:
  $pre(C)$ – zbiór bezpośrednich poprzedników,
  $pre^*(C)$ – zwrotne i przechodnie domknięcie pre.
  Mamy:

$$pre^*(C) = \left\{c \in P \times \Gamma^* | \exists c' \in C . c \Rightarrow c'\right\}$$

  Podobnie:

$$post^*(C) = \left\{c \in P \times \Gamma^* | \exists c' \in C . c' \Rightarrow c\right\}$$

- **Problem osiągalności**: czy dla danego $C \in P \times \Gamma^*$ stan początkowy należy do $pre^*(C)$?

Problemy

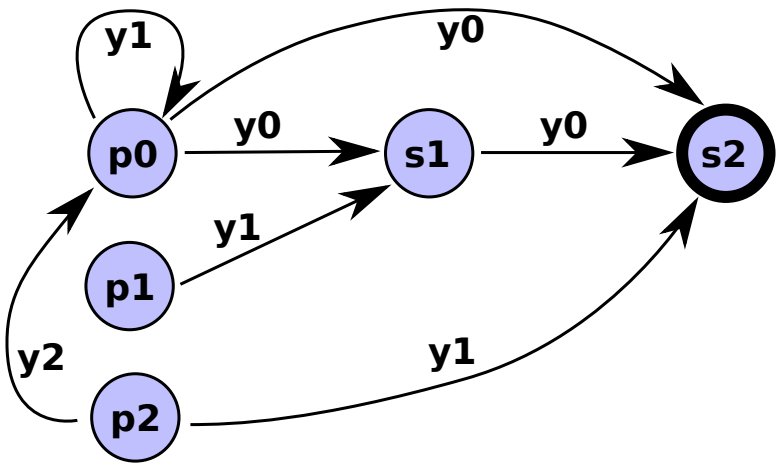
- Zbiór $pre^*(C)$ jest przeważnie **nieskończony**
- Należy znaleźć metodę reprezentowania takich zbiorów za pomocą struktur, które:
  - są skończone
  - są zamknięte ze względu na podstawowe operacje (suma, iloczyn)
  - zapewniają, że sprawdzenie czy element należy do reprezentowanego przez nie zbioru jest rozstrzygalne

Multiautomat (ang. *multi-automaton*)

Dla PDS $\mathcal{P} = (P, \Gamma, \Delta)$ określamy $\mathcal{P}$-multiautomat ($\mathcal{P}$-MA) jako krotkę $\mathcal{A} = (Q, \Gamma, \delta, P, F)$, gdzie:

- Q – zbiór stanów
- $\delta \subseteq Q \times \Gamma \times Q$ – relacja przejścia (zbiór tranzycji)
- $P \subseteq Q$, $F \subseteq Q$ – zbiory stanów początkowych i końcowych

Jeśli automat zaczynając w stanie q po wczytaniu słowa w może znaleźć się w stanie q' to piszemy $q \xrightarrow{w} q'$.
$\mathcal{A}$ **akceptuje** $\langle p, w \rangle$ jeśli $p \xrightarrow{w} q$ dla pewnego $q \in F$.
$Conf(\mathcal{A})$ – zbiór konfiguracji akceptowanych przez automat.



Regularność zbioru konfiguracji

Zbiór konfiguracji $C \subseteq P \times \Gamma^*$ nazwiemy **regularnym**, gdy dla każdego punktu sterowania $p \in P$ zbiór $\{w \in \Gamma^* | \langle p, w \rangle \in C\}$ jest regularny.
$C \subseteq P \times \Gamma^*$ jest regularny wtedy i tylko wtedy, gdy istnieje multiautomat akceptujący C.

Regularność zbioru konfiguracji

Zbiór konfiguracji $C \subseteq P \times \Gamma^*$ nazwiemy **regularnym**, gdy dla każdego punktu sterowania $p \in P$ zbiór $\{w \in \Gamma^* | \langle p, w \rangle \in C\}$ jest regularny.
$C \subseteq P \times \Gamma^*$ jest regularny wtedy i tylko wtedy, gdy istnieje multiautomat akceptujący C.

Twierdzenie

Jeśli C jest regularny, to $pre^*(C)$ też jest regularny.

Obliczanie $pre^*(C)$

- $X_0 = C$
- $X_{i+1} = X_i \cup pre(X_i)$
- $pre^*(C) = \bigcup_{i \geqslant 0} X_i$

Obliczanie $pre^*(C)$

- $X_0 = C$
- $X_{i+1} = X_i \cup pre(X_i)$
- $pre^*(C) = \bigcup_{i \geqslant 0} X_i$

Taka metoda **nie zbiega** do punktu stałego...

$(p, \gamma) \hookrightarrow (p, \varepsilon)$, $C = \{\langle p, \varepsilon \rangle\}$ daje:
$X_i = \{\langle p, \varepsilon \rangle, \langle p, \gamma \rangle, \ldots, \langle p, \gamma^i \rangle\}$

Obliczanie $pre^*(C)$

Skonstruujemy rosnący ciąg zbiorów konfiguracji Y_i, który posiada własności:

W1. $\exists i \geqslant 0.\ Y_{i+1} = Y_i$

W2. $\forall i \geqslant 0.\ X_i \subseteq Y_i$

W3. $\forall i \geqslant 0.\ Y_i \subseteq \bigcup_{j \geqslant 0} X_j$

Obliczanie $pre^*(C)$

Skonstruujemy rosnący ciąg zbiorów konfiguracji Y_i, który posiada własności:

W1. $\exists i \geqslant 0.\ Y_{i+1} = Y_i$

W2. $\forall i \geqslant 0.\ X_i \subseteq Y_i$

W3. $\forall i \geqslant 0.\ Y_i \subseteq \bigcup_{j \geqslant 0} X_j$

konstruując ciąg **automatów** $\mathcal{A}_0, \mathcal{A}_1, \ldots$ o tym samym zbiorze stanów, ale o rosnących zbiorach tranzycji. $Y_i = Conf(\mathcal{A}_i)$.

Obliczanie $pre^*(C)$

- $\mathcal{A}_0 = \mathcal{A}$, gdzie $Conf(\mathcal{A}) = C$
- $\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).

Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).

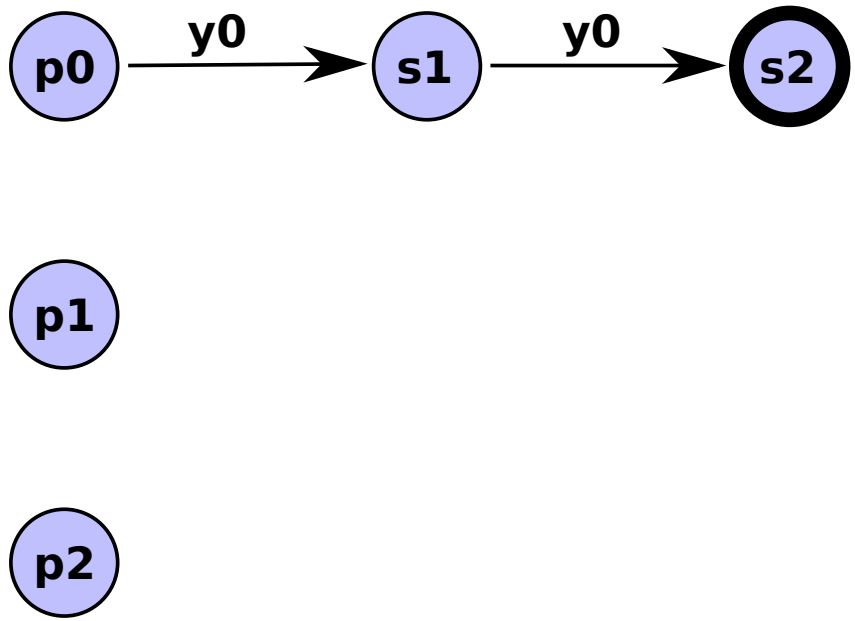
⟨p0, y0⟩ ↪ ⟨p1, y1y0⟩

⟨p1, y1⟩ ↪ ⟨p2, y2y0⟩

⟨p2, y2⟩ ↪ ⟨p0, y1⟩

⟨p0, y1⟩ ↪ ⟨p0, ε⟩

C={⟨p0,y0y0⟩}



Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).

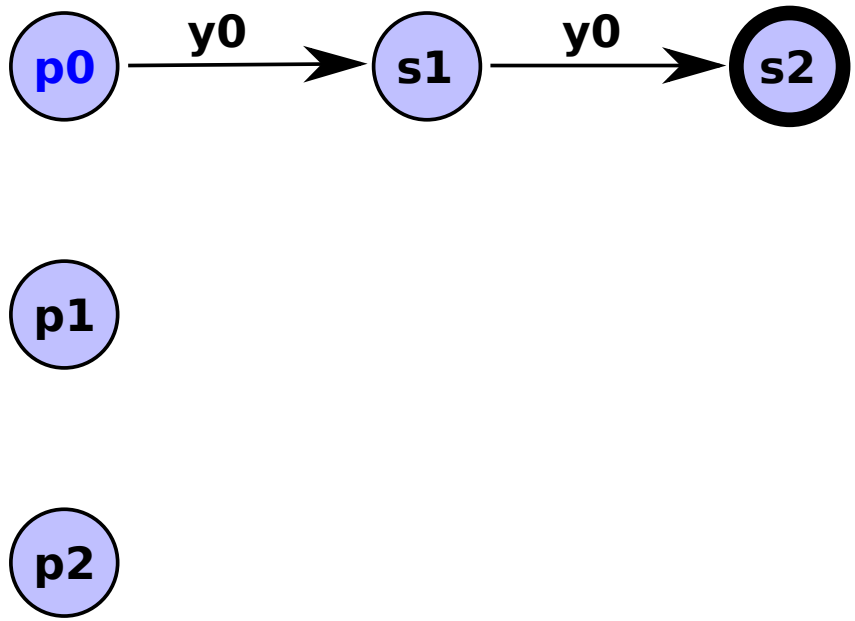
⟨p0, y0⟩ ↪ ⟨p1, y1y0⟩

⟨p1, y1⟩ ↪ ⟨p2, y2y0⟩

⟨p2, y2⟩ ↪ ⟨p0, y1⟩

⟨p0, y1⟩ ↪ ⟨p0, ε⟩

C={⟨p0,y0y0⟩}



Przykład

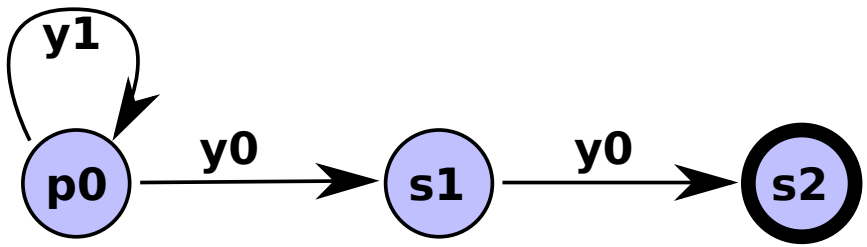
$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).

⟨p0, y0⟩ ↪ ⟨p1, y1y0⟩

⟨p1, y1⟩ ↪ ⟨p2, y2y0⟩

⟨p2, y2⟩ ↪ ⟨p0, y1⟩

⟨p0, y1⟩ ↪ ⟨p0, ε⟩



C={⟨p0,y0y0⟩}

Przykład

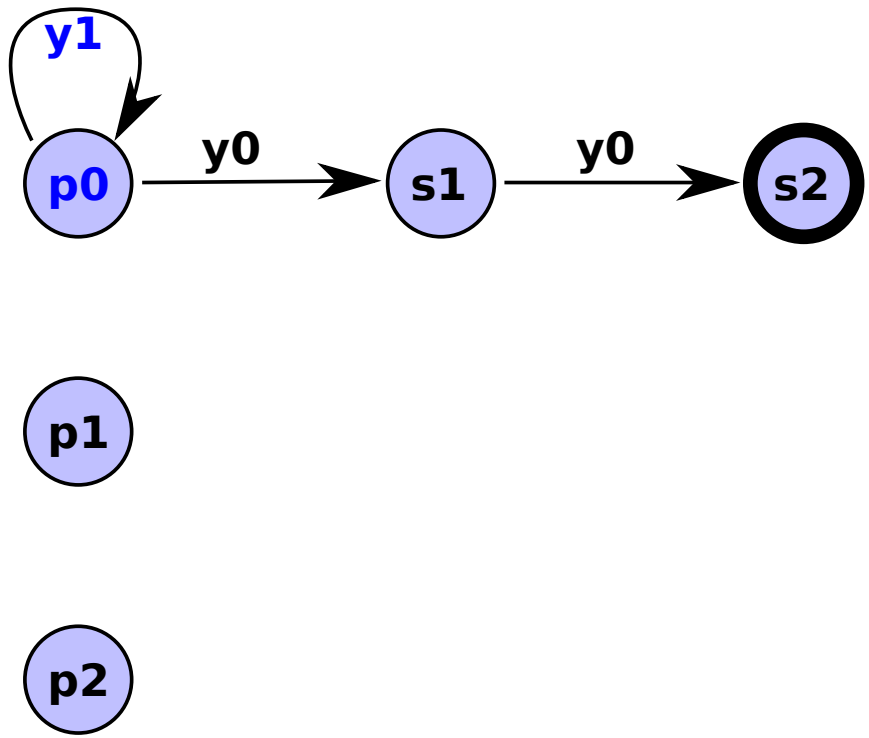
$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).

⟨p0, y0⟩ ↪ ⟨p1, y1y0⟩

⟨p1, y1⟩ ↪ ⟨p2, y2y0⟩

⟨p2, y2⟩ ↪ ⟨p0, y1⟩

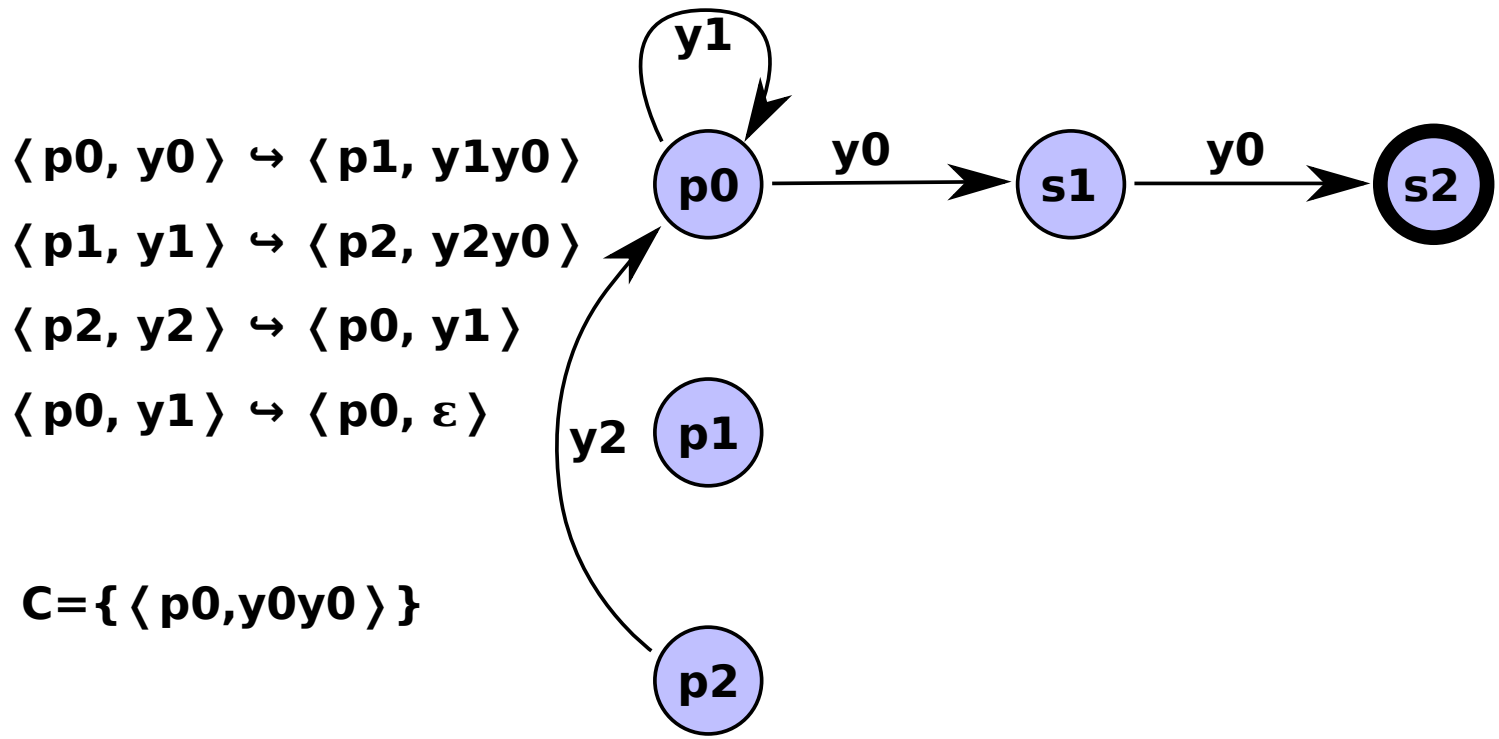
⟨p0, y1⟩ ↪ ⟨p0, ε⟩



C={⟨p0,y0y0⟩}

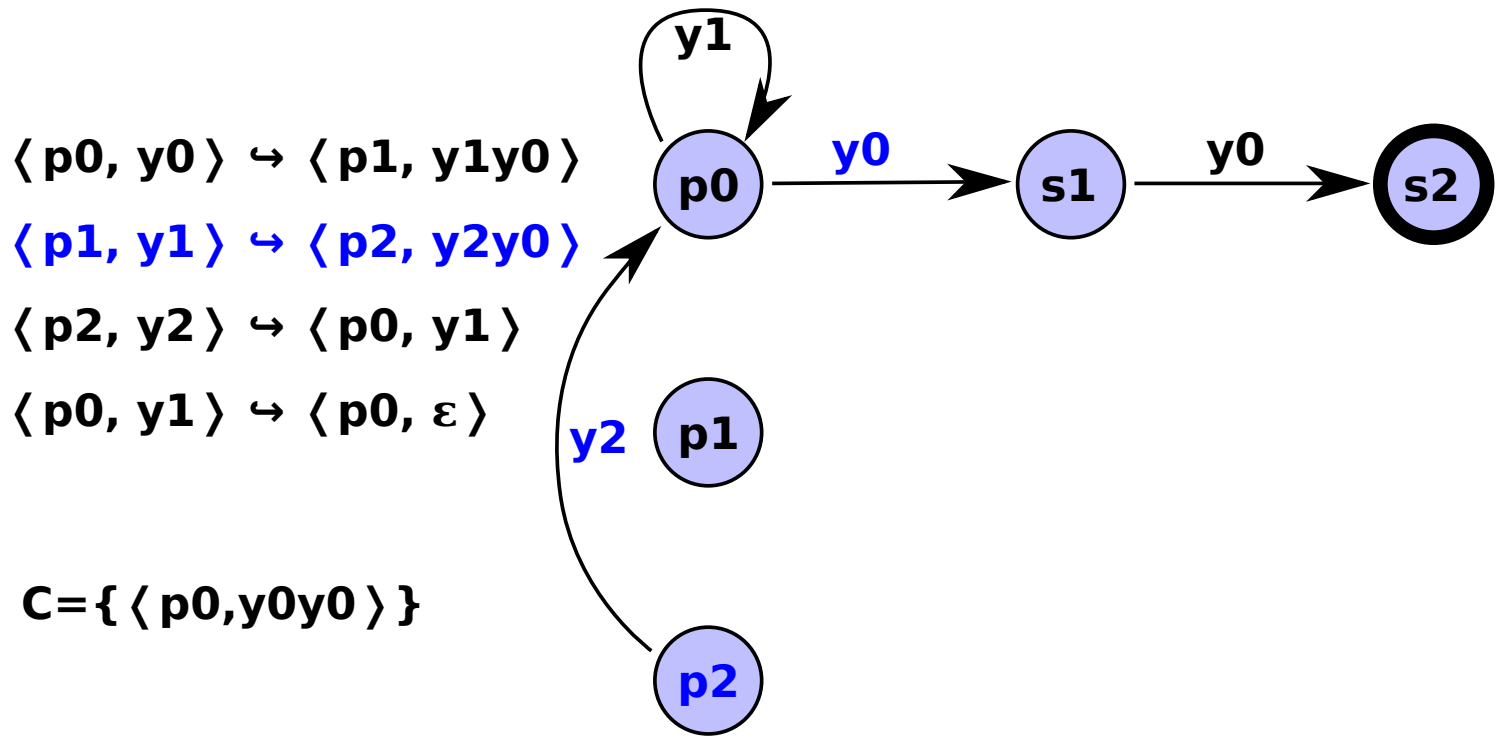
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



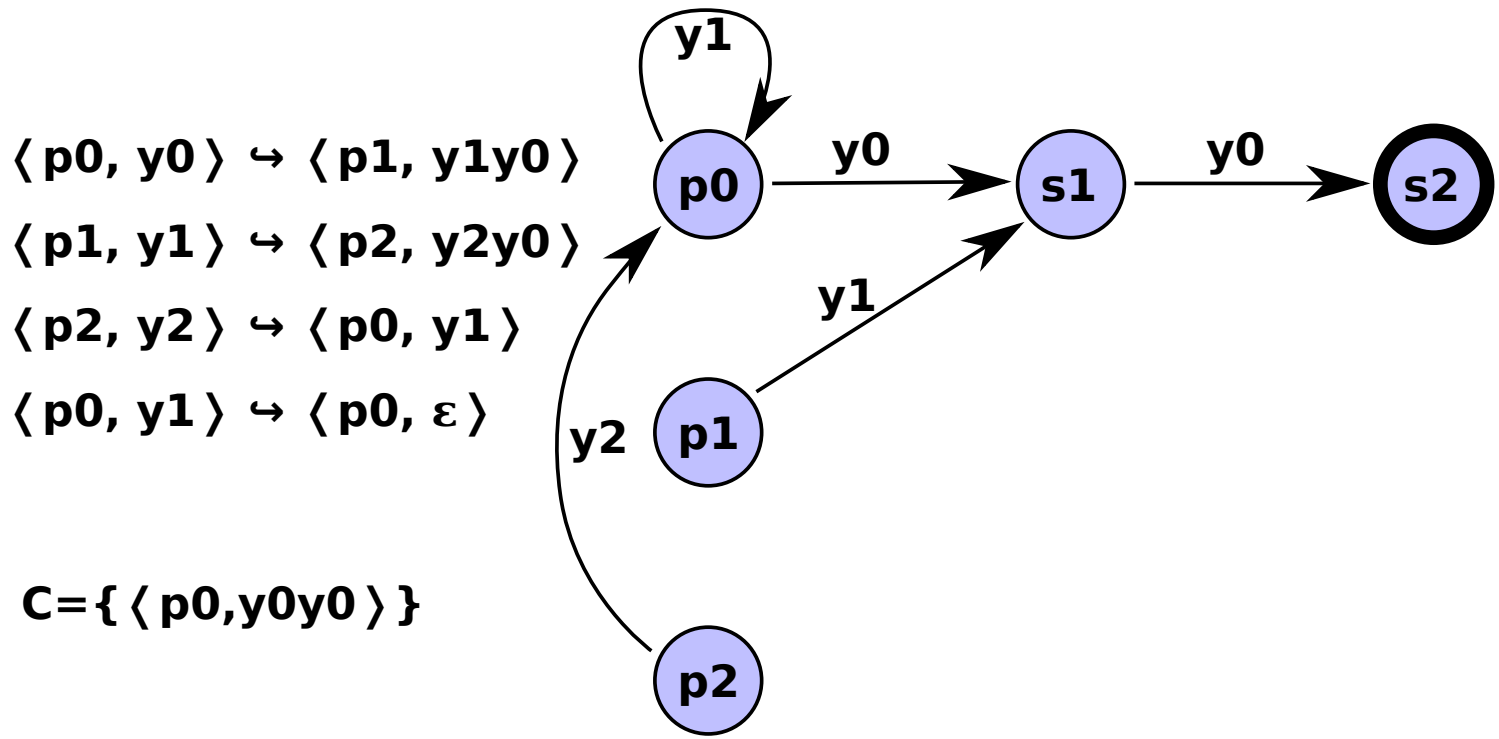
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



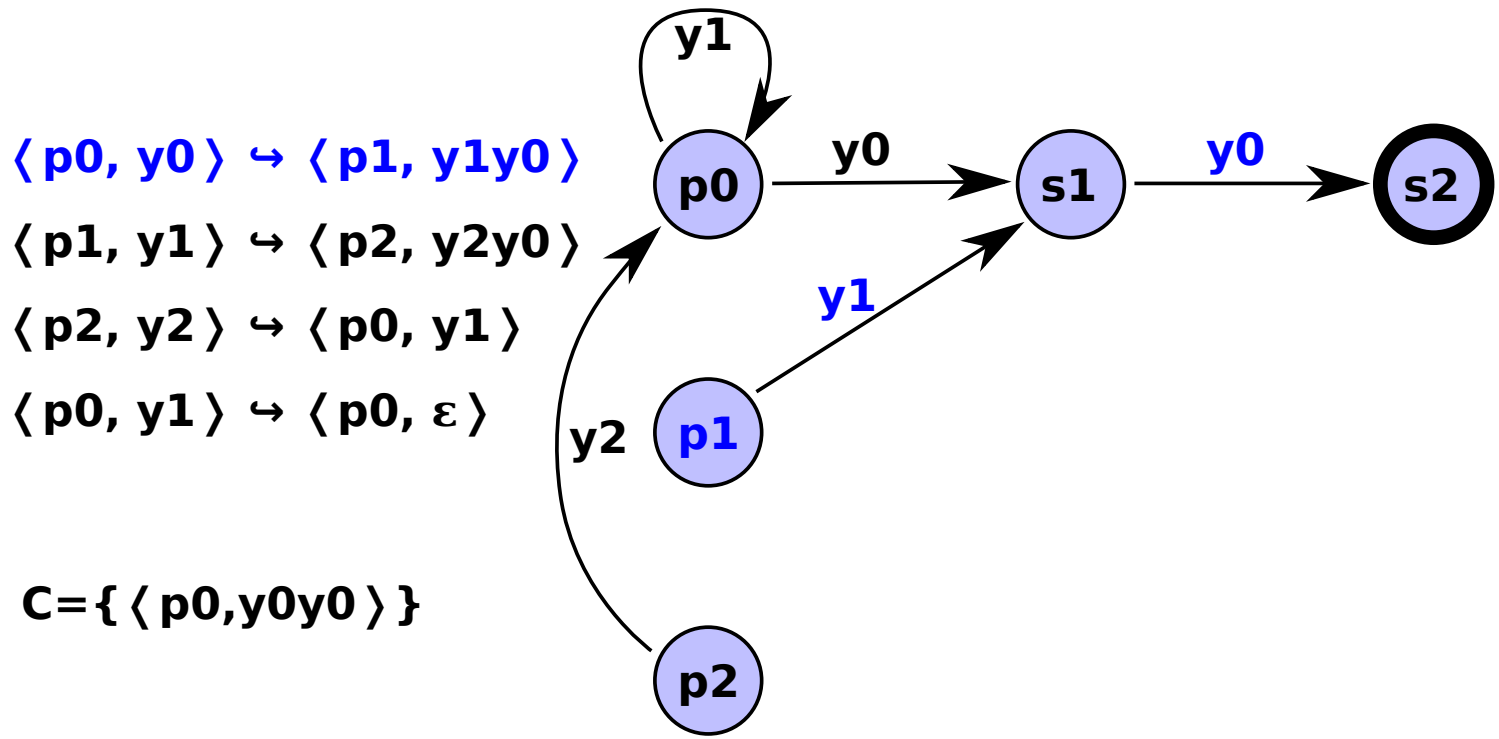
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



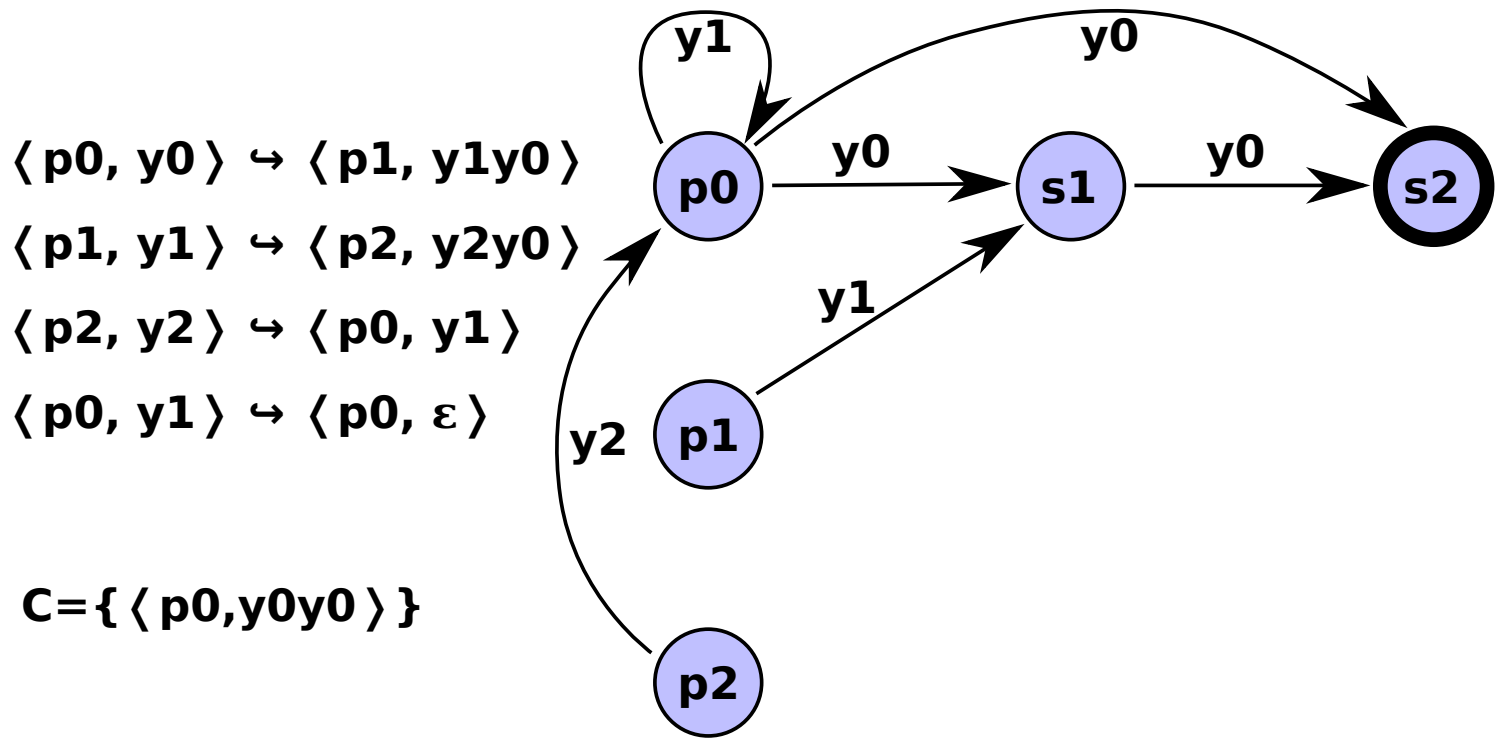
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



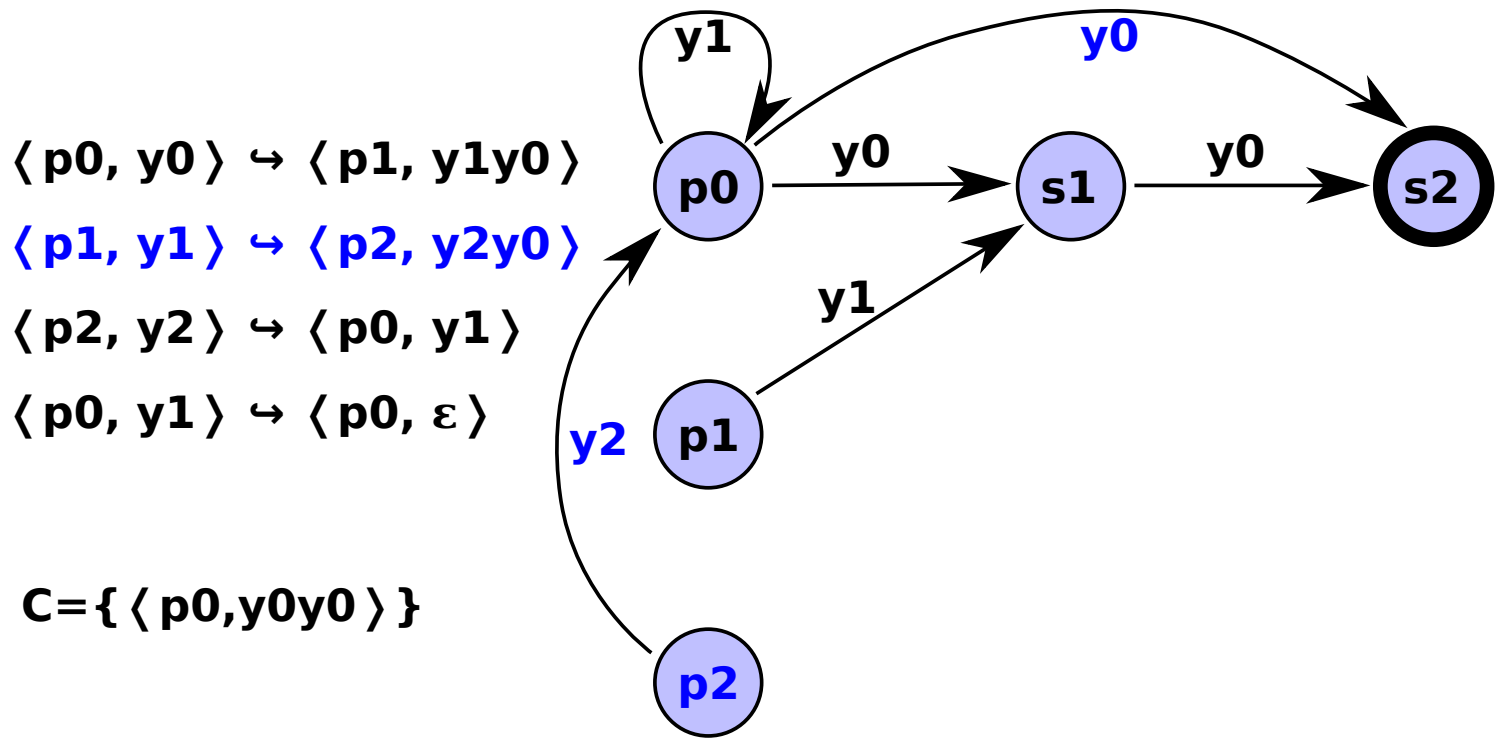
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



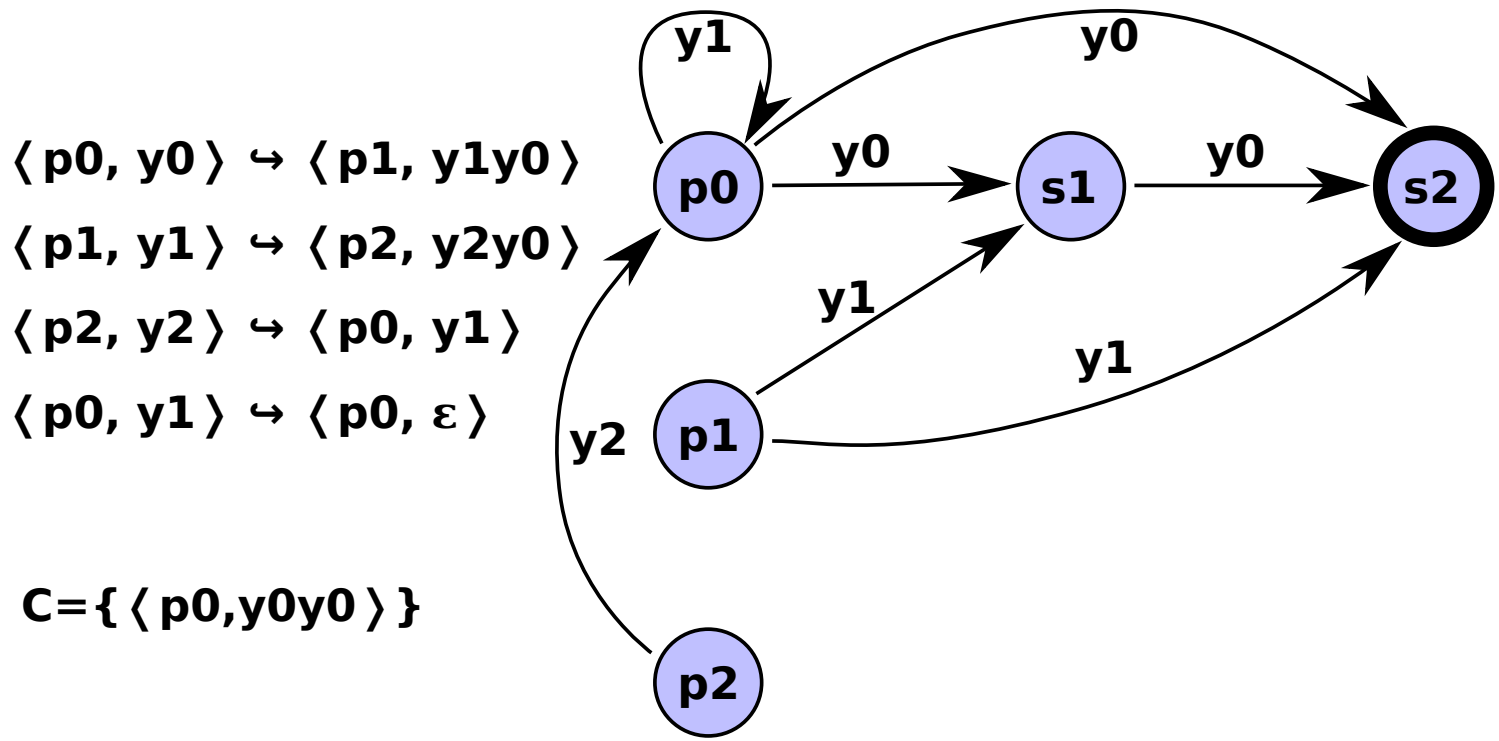
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



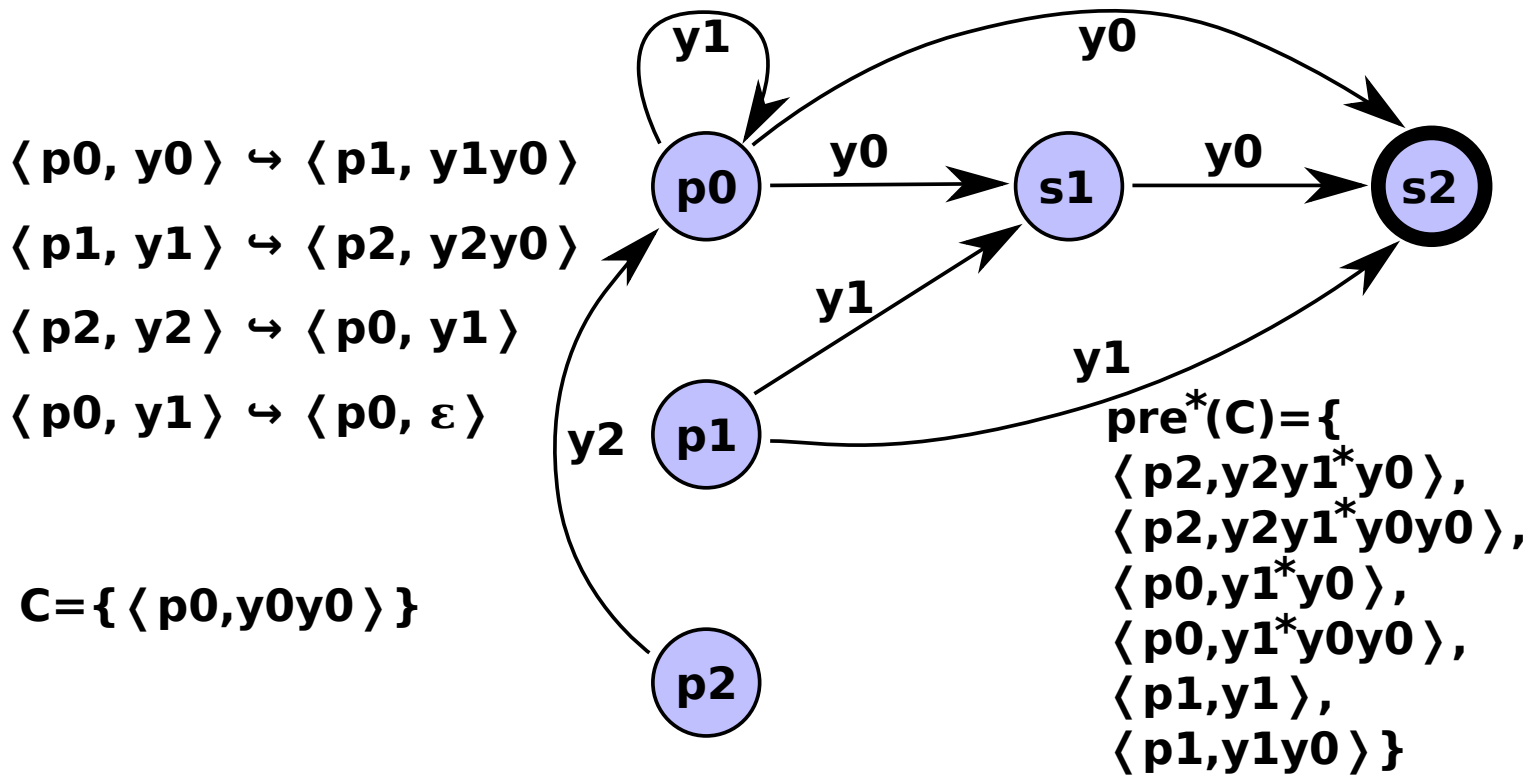
Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



Przykład

$\mathcal{A}_{i+1}$ powstaje z $\mathcal{A}_i$ przez dodanie dla każdej reguły $(p^j, \gamma) \hookrightarrow (p^k, w)$ oraz każdego stanu $q \in Q$ t. że. $p^k \xrightarrow{w}_i q$ nowej tranzycji (p^j, γ, q).



Obliczanie $post^*(C)$

Założenie: dla $(p, \gamma) \hookrightarrow (p', w)$ mamy $|w| \leqslant 2$, $C = Conf(\mathcal{A})$.

Oznaczmy relację $\left(\xrightarrow{\varepsilon}\right)^* \xrightarrow{\gamma} \left(\xrightarrow{\varepsilon}\right)^*$ przez $\overset{\gamma}{\Longrightarrow}$

krok I Dla każdego $r = (p, \gamma) \hookrightarrow (p', \gamma'\gamma'') \in \Delta$ dodaj do $\mathcal{A}$ stan r i tranzycję (p', γ', r).

krok II Dodaj tranzycje zgodnie z zasadami:

- Jeśli $(p, \gamma) \hookrightarrow (p', \varepsilon) \in \Delta$ i $p \overset{\gamma}{\Longrightarrow} q$, to dodaj (p', ε, q)
- Jeśli $(p, \gamma) \hookrightarrow (p', \gamma') \in \Delta$ i $p \overset{\gamma}{\Longrightarrow} q$, to dodaj (p', γ', q)
- Jeśli $(p, \gamma) \hookrightarrow (p', \gamma'\gamma'') \in \Delta$ i $p \overset{\gamma}{\Longrightarrow} q$, to dodaj (r, γ'', q)

Złożoność

$\mathcal{P} = (P, \Gamma, \Delta)$, $C = Conf(\mathcal{A})$, $\mathcal{A} = (Q, \Gamma, \delta, P, F)$:

- Obliczanie $pre^*(C)$ w czasie $\mathcal{O}\left(|Q|^2|\Delta|\right)$ i pamięci $\mathcal{O}\left(|Q||\Delta| + |\delta|\right)$
- Obliczanie $post^*(C)$ w czasie i pamięci $\mathcal{O}\left(|P||\Delta|(|Q| + |\Delta|) + |P||\delta|\right)$

Sprawdzanie asercji

Asercja może być niespełniona $\Rightarrow$

osiągalny stan błędu $\Rightarrow$

$pre^*(Error) \cap Init$ – **kontrprzykłady**

Weryfikacja LTL

- $\mathcal{P} = (P, \Gamma, \Delta)$
- Formuła LTL $\varphi \longrightarrow$ **automat Büchiego** $\mathcal{B}$ dla $\neg\varphi$
- $\mathcal{P} \times \mathcal{B} = \mathcal{BP}$ – **system Büchiego** ze stosem
  G – zbiór stanów akceptujących. Określmy relację na konfiguracjach: $c \stackrel{r}{\Longrightarrow} c'$ gdy $c \Rightarrow \langle g, u\rangle \Rightarrow^{+} c'$ dla $g \in G$.
  $\langle p, \gamma\rangle$ nazwiemy **powracającą głową** jeśli $\langle p, \gamma\rangle \stackrel{r}{\Longrightarrow} \langle p, \gamma v\rangle$.
  R – zbiór powracających głów, obliczalny w czasie wielomianowym.
- $post^*(Init) \cap pre^*(R\Gamma^*)$ – zbiór kontrprzykładów

Narzędzia

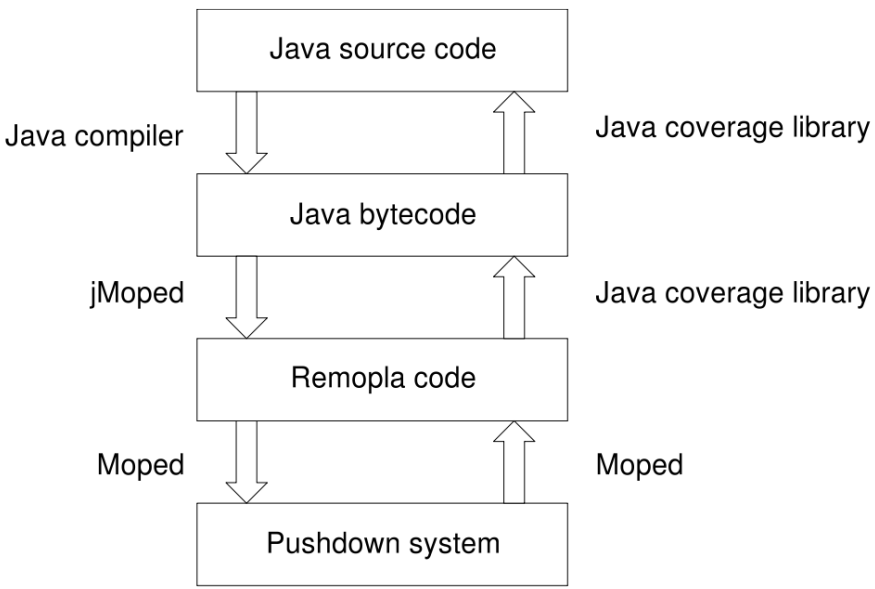
Moped

- Weryfikacja osiągalności dla programów rekurencyjnych w języku Remopla
- Remopla $\longrightarrow$ PDS
- BDD dla efektywnej reprezentacji
- CEGAR

Narzędzia

jMoped

- Testowanie i weryfikacja programów w Javie
- Translator: bajtkod Java $\longrightarrow$ Remopla
- Dodatkowo: ograniczona weryfikacja programów współbieżnych



Bibliografia


A. Bouajjani, J. Esparza, O. Maler
Reachability analysis of pushdown automata: Application to model-checking
CONCUR '97, t. 1243 LNCS, ss. 135-150. Springer, 1997.


F. Berger
A Test and Verification Environment For Java Programs
praca dyplomowa, Institut für Formale Methoden der Informatik, Universität Stuttgart


J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon
Efficient Algorithms for Model Checking Pushdown Systems
CAV, t. 1855 LNCS, ss. 232-247. Springer, 2000.


`http://www7.in.tum.de/tools/jmoped`