Sieci Petriego – czy to potrzebne?

Filip Mazowiecki

Uniwersytet Warszawski

65. Szkoła Matematyki Poglądowej

Sieci Petriego – czy to potrzebne?

Spoiler: prawo nagłówków Betteridge'a (NIE).

Filip Mazowiecki

Uniwersytet Warszawski

65. Szkoła Matematyki Poglądowej

Plan

1. Wstęp i przykłady

2. Coś o zastosowaniach

3. Jakiś dowód

Prawie sieci Petriego

Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

Prawie sieci Petriego

Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

Prawie sieci Petriego

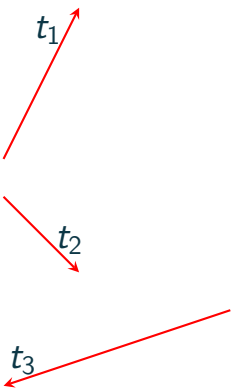
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$



Prawie sieci Petriego

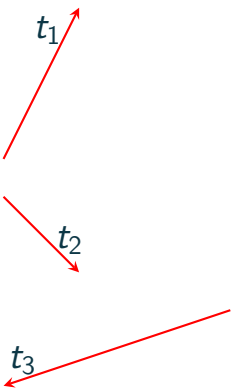
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$



Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

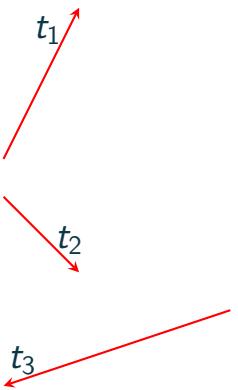
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

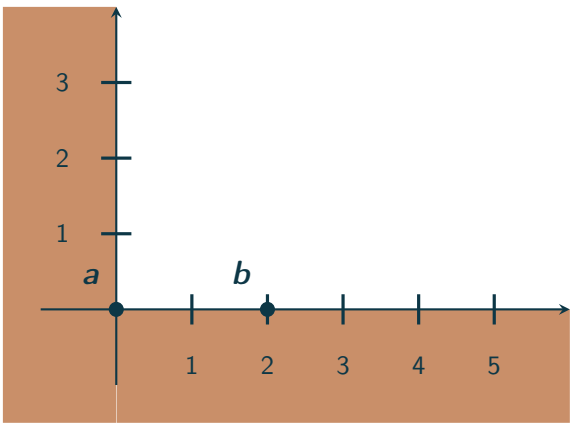
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

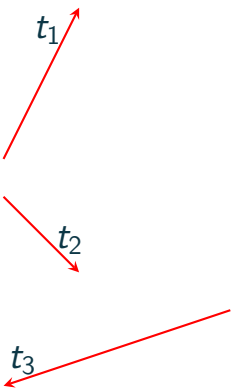
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

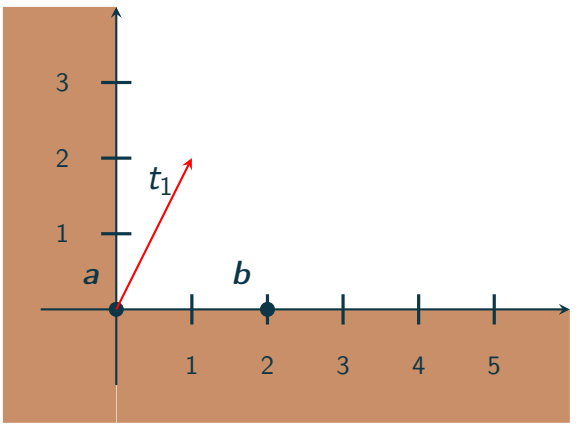
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

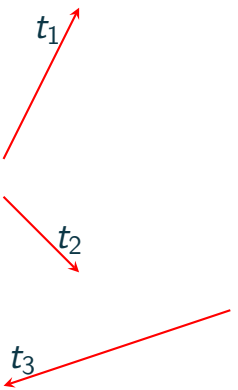
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

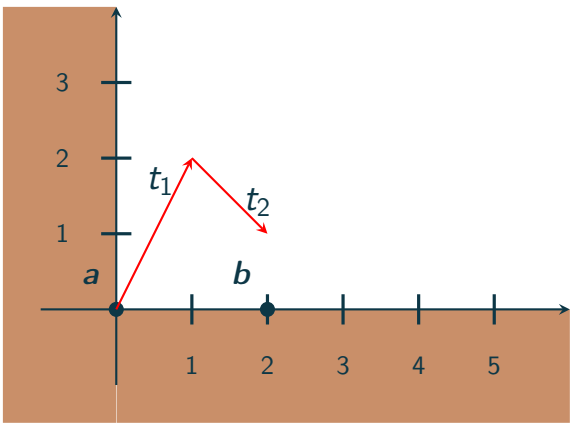
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

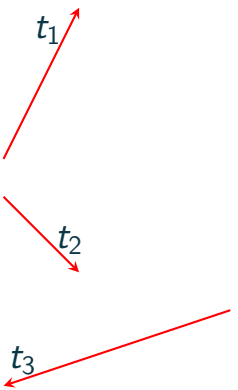
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

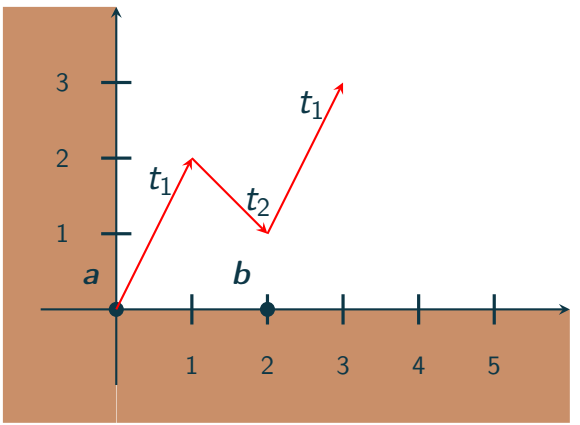
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

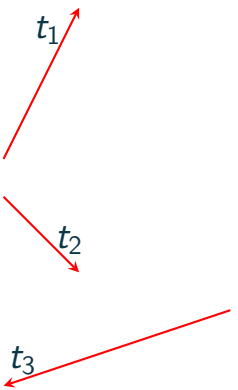
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

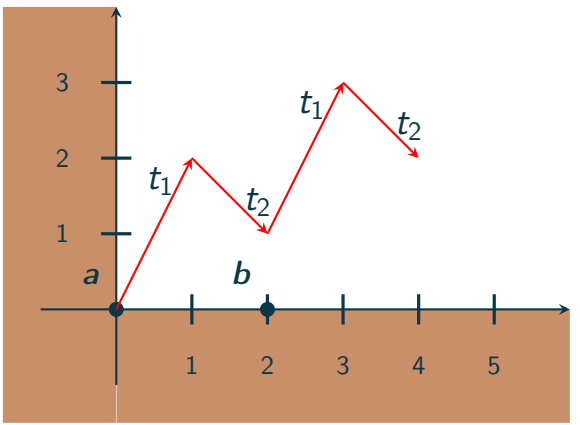
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

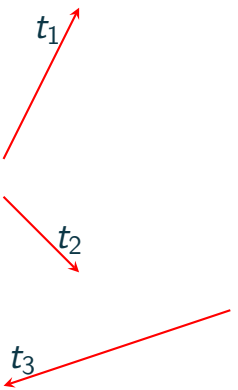
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

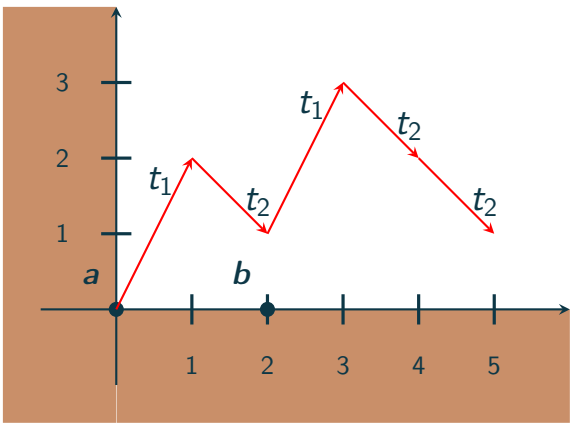
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

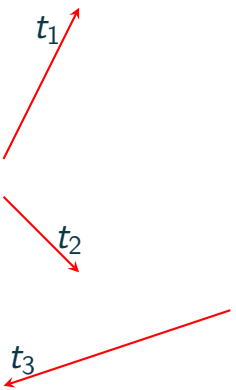
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

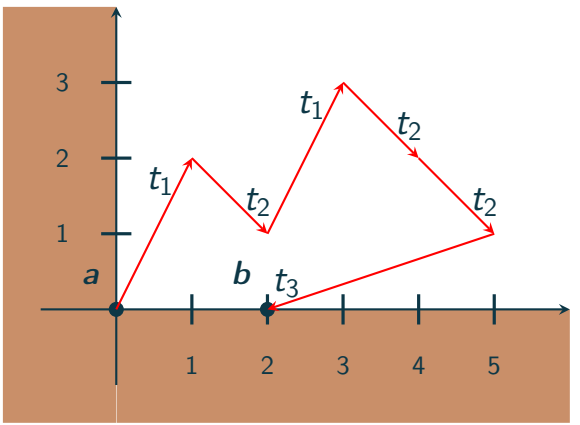
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

Prawie sieci Petriego

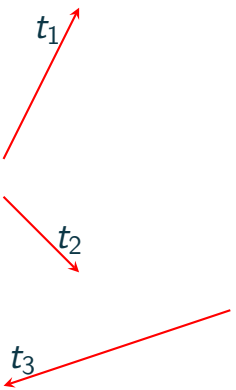
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

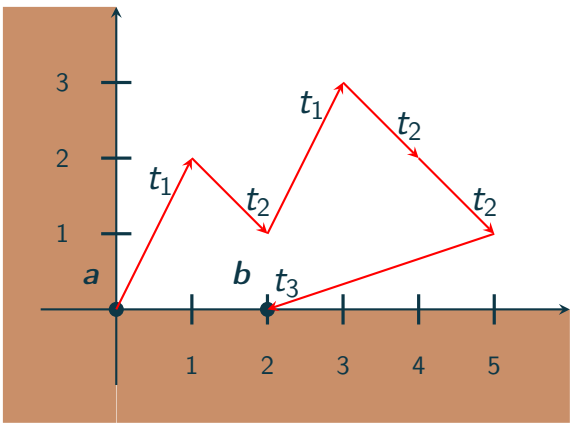
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

(bez t_3 się nie da)

Prawie sieci Petriego

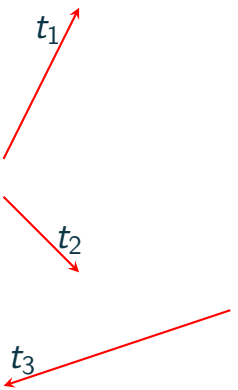
Prawie sieć Petriego (d, T): d – wymiar, $T \subseteq \mathbb{Z}^d$ (skończony)

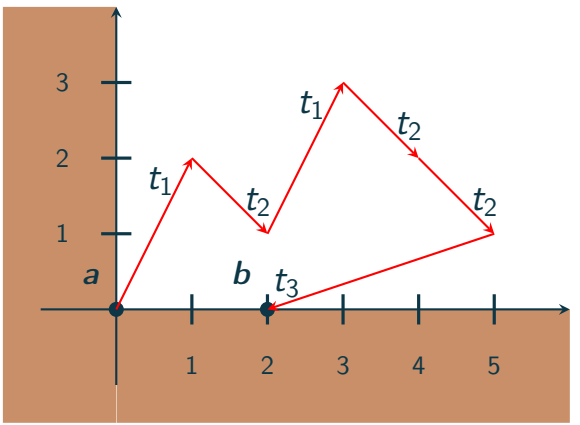
Przykład: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Problem osiągalności: dane (d, T) i dwa wektory $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Czy można przejść z $\boldsymbol{a}$ do $\boldsymbol{b}$ pozostając w $\mathbb{N}^d$?

(bez t_3 się nie da)

Chcemy algorytm który sprawdza takie rzeczy

Sieci Petriego

(d, T): d – wymiar, $T \subseteq \not{\mathbb{Z}}^d \quad \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Sieci Petriego

(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

Sieci Petriego

(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$

Sieci Petriego

(d, T): d – wymiar, $T \subseteq \not{\mathbb{Z}}^d$ $\mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ (,) = -(,) + (,)

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$

Sieci Petriego

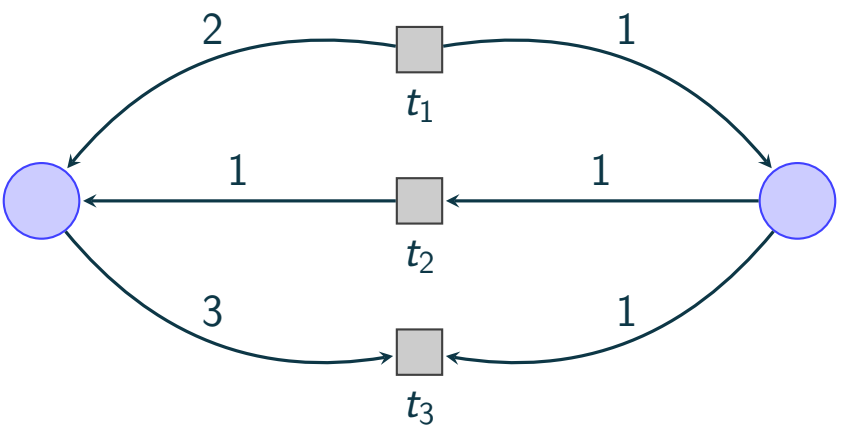
(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



Sieci Petriego

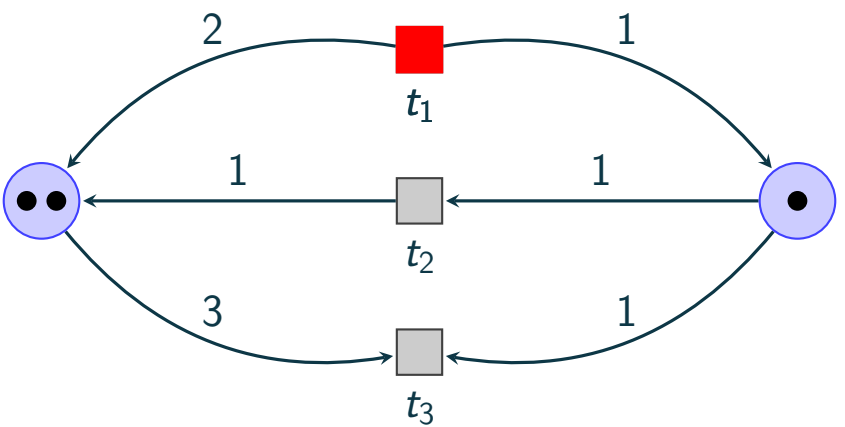
(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



Sieci Petriego

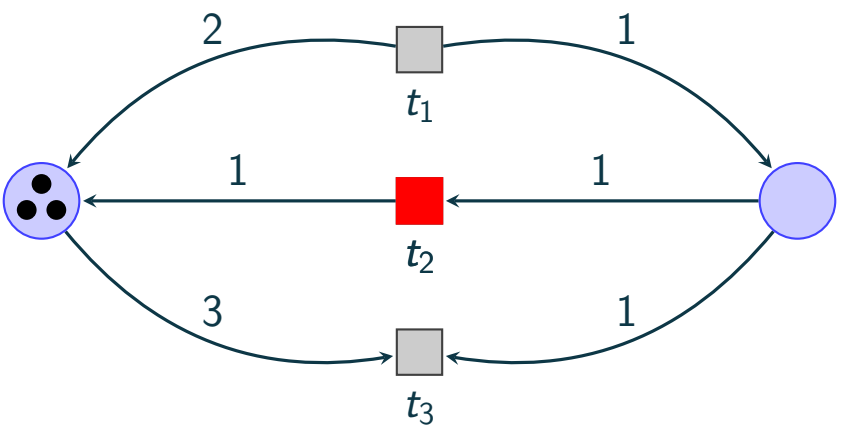
(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



Sieci Petriego

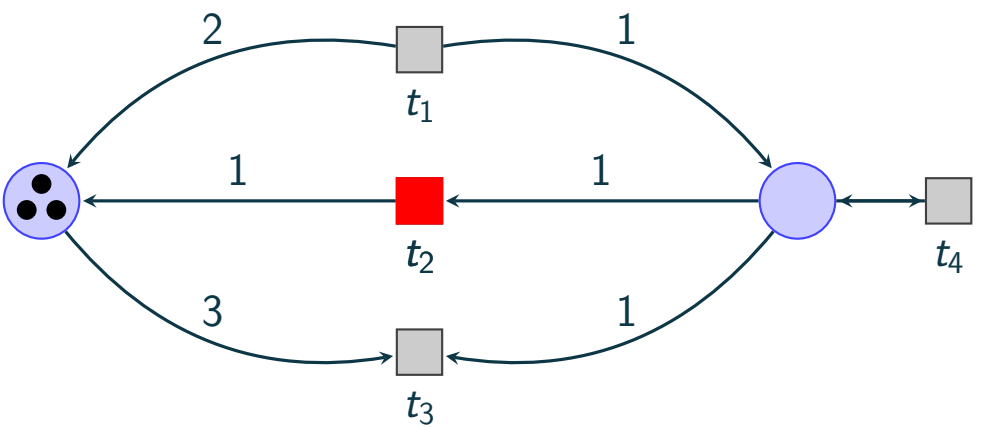
(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



Sieci Petriego

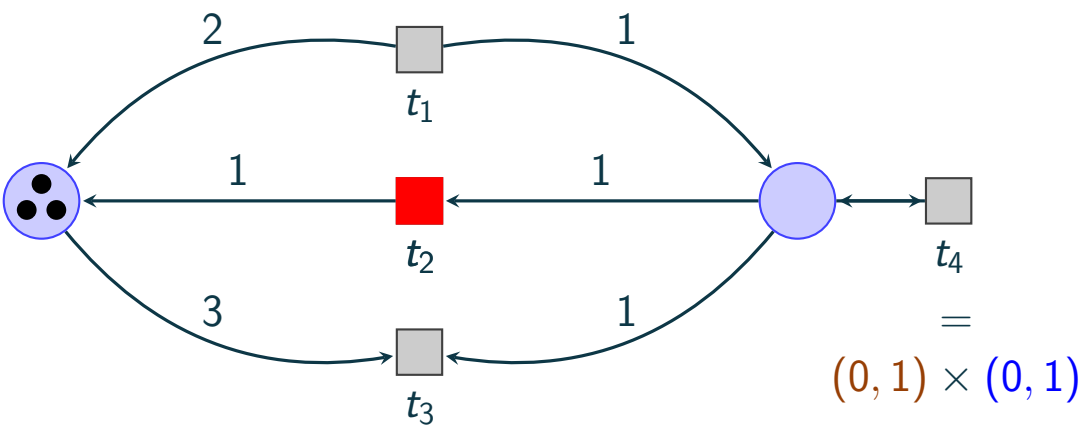
(d, T): d – wymiar, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (skończony)

Poprzedni przykład $d = 2$, $T = \{t_1', t_2', t_3'\}$

zamiast $t_1 = (1, 2)$ jest $t_1' = (0, 0) \times (1, 2)$

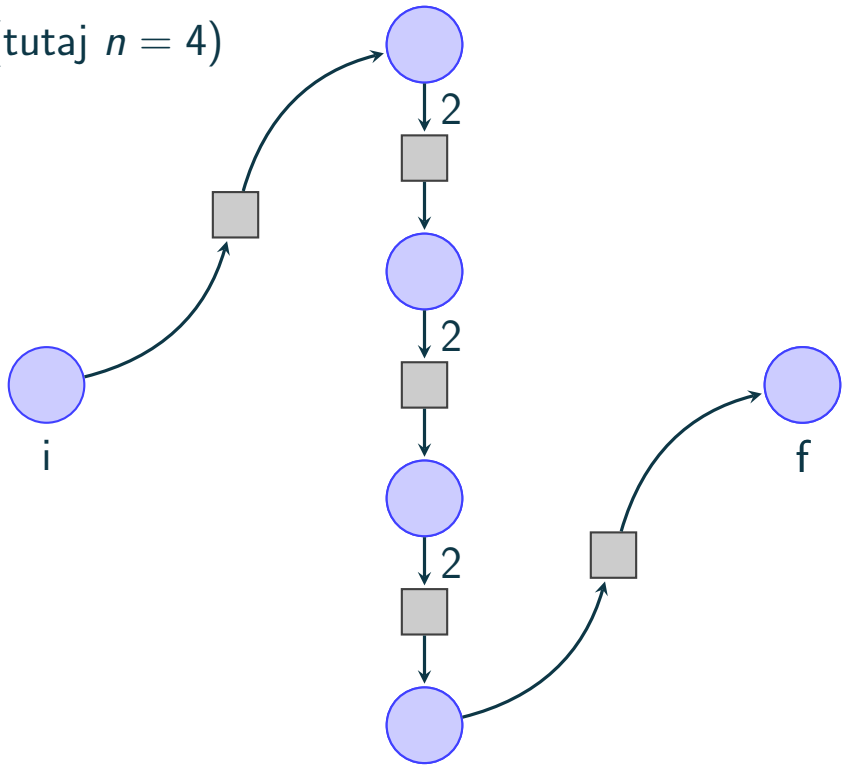
zamiast $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

zamiast $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



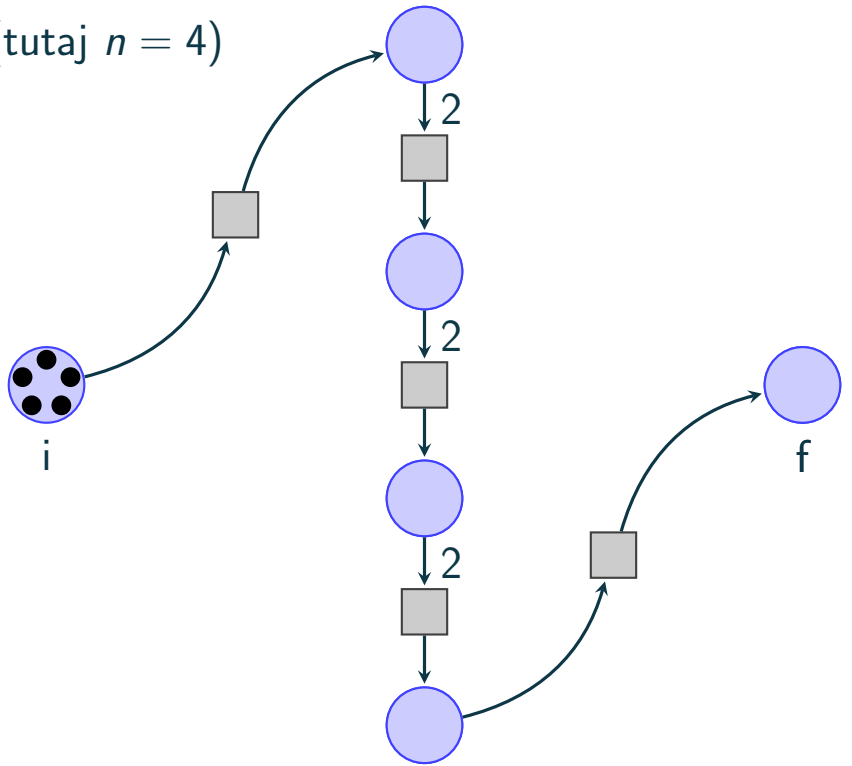
Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



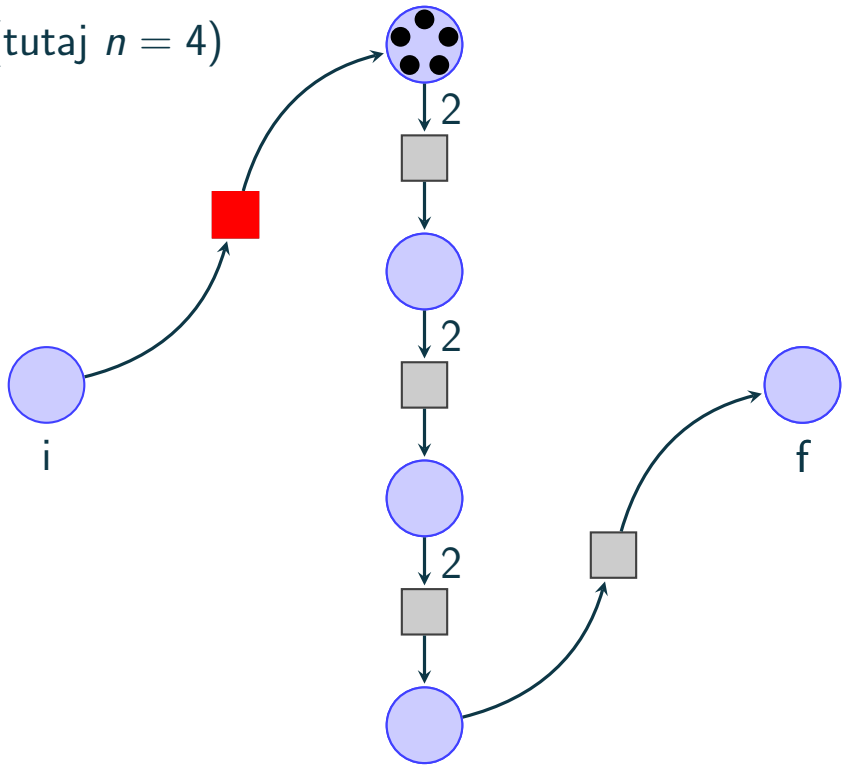
Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



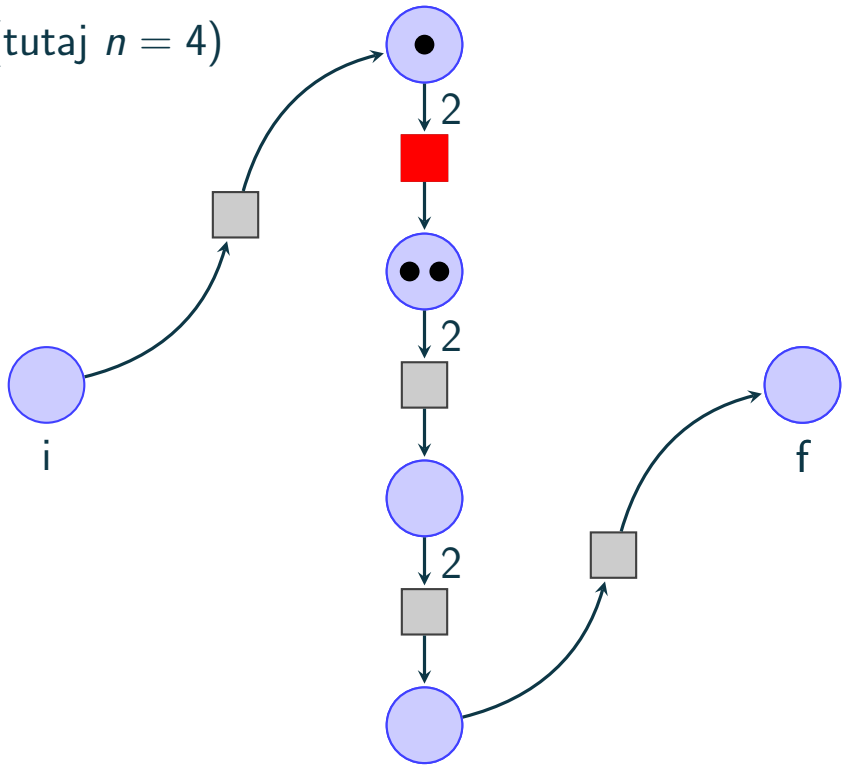
Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



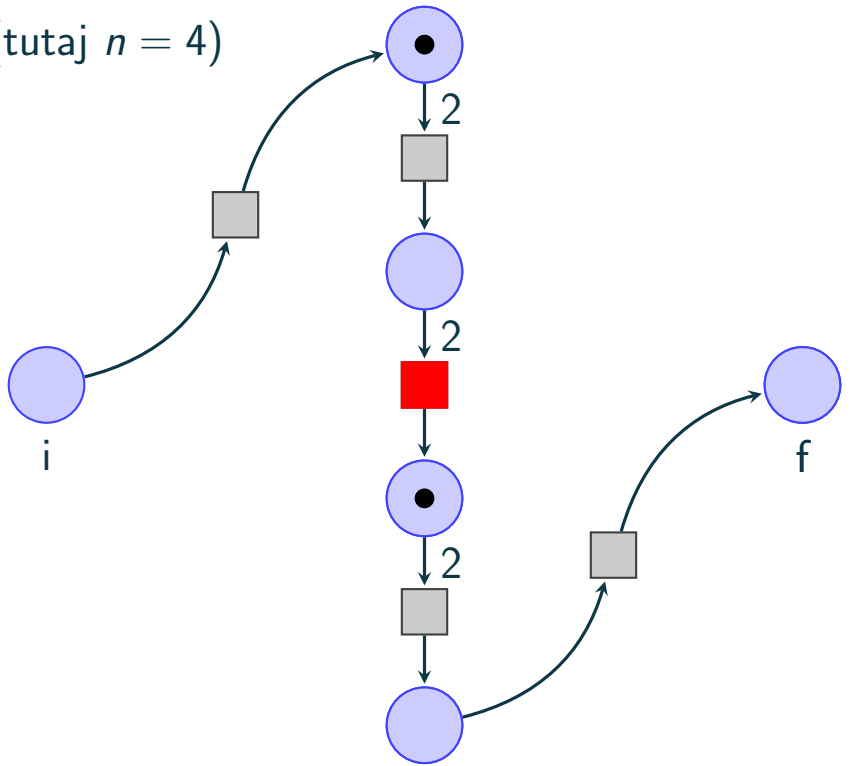
Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



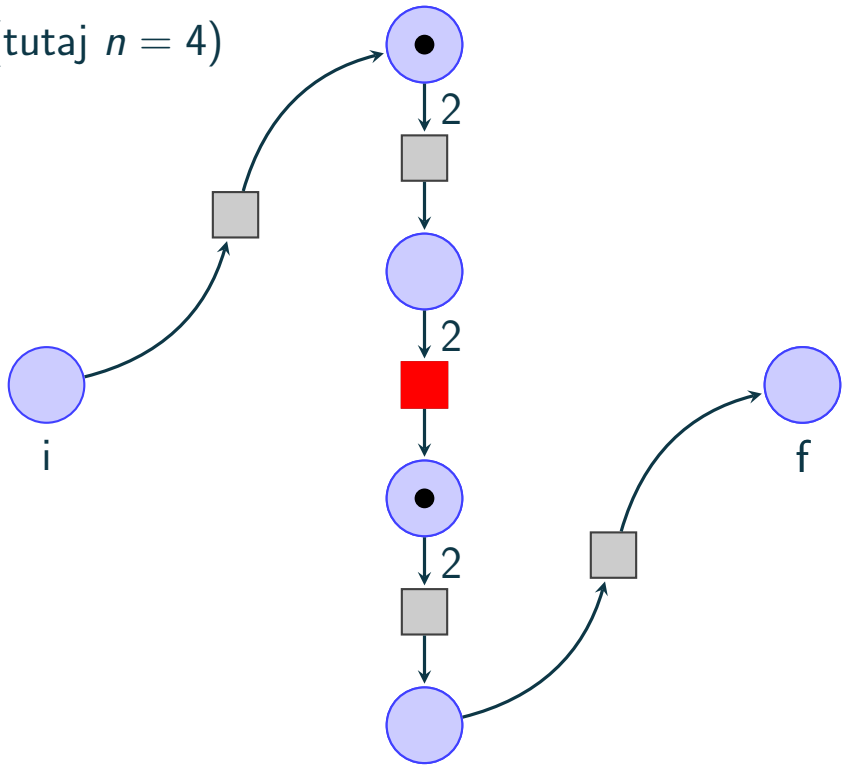
Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



Ciekawszy przykład

$d = 2 + n$ (tutaj $n = 4$)



Jeśli zaczniemy z k żetonami w i

to dojdziemy do f tylko jeśli $k \geqslant 2^{n-1}$

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Co to właściwie znaczy?

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Co to właściwie znaczy?

Trzeba jakoś sformalizować.

> Jeśli z $(k, 0, 0, 0, 0, 0)$ można dojść do $\boldsymbol{b}$ i $\boldsymbol{b}[6] \geqslant 1$ to $k \geqslant 2^{n-1}$

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Co to właściwie znaczy?

Trzeba jakoś sformalizować.

Jeśli z $(k, 0, 0, 0, 0, 0)$ można dojść do $\boldsymbol{b}$ i $\boldsymbol{b}[6] \geqslant 1$ to $k \geqslant 2^{n-1}$

To nie jest dokładnie problem osiągalności

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Co to właściwie znaczy?

Trzeba jakoś sformalizować.

Jeśli z $(k, 0, 0, 0, 0, 0)$ można dojść do $\boldsymbol{b}$ i $\boldsymbol{b}[6] \geqslant 1$ to $k \geqslant 2^{n-1}$

To nie jest dokładnie problem osiągalności

Jakie inne rzeczy można sprawdzać?

Osiągalność i inne problemy

Powiemy że poprzednia sieć sprawdza $\geqslant 2^{n-1}$

Co to właściwie znaczy?

Trzeba jakoś sformalizować.

Jeśli z $(k, 0, 0, 0, 0, 0)$ można dojść do $\boldsymbol{b}$ i $\boldsymbol{b}[6] \geqslant 1$ to $k \geqslant 2^{n-1}$

To nie jest dokładnie problem osiągalności

Jakie inne rzeczy można sprawdzać?

O tym następna część.

Plan

1. Wstęp i przykłady

2. Coś o zastosowaniach

3. Jakiś dowód

Weryfikacja programów z równoległymi procesami

$\{x = \text{True}\}$

1: **goto** 2

2: **if** x **then goto** 3 **else goto** 1

3: x = !x

4: # sekcja krytyczna

5: x = !x, **exit**

Weryfikacja programów z równoległymi procesami

{x = True }
1: **goto** 2
2: **if** x **then goto** 3 **else goto** 1
3: x = !x
4: # sekcja krytyczna
5: x = !x, **exit**

Do pierwszej linijki może wejść dowolnie wiele procesów

Pytanie: czy dwa procesy mogą się znaleźć naraz w sekcji krytycznej?

Weryfikacja programów z równoległymi procesami

{x = True }		{łazienka wolna}
1: **goto** 2		idź z korytarza do drzwi łazienki
2: **if** x **then goto** 3 **else goto** 1		jeśli wolna wejdź, jeśli nie wróć na korytarz
3: x = !x		zmień oznaczenie czy wolna
4: # sekcja krytyczna		# łazienka
5: x = !x, **exit**		zmień oznaczenie czy wolna i wyjdź

Do pierwszej linijki może wejść dowolnie wiele procesów

Pytanie: czy dwa procesy mogą się znaleźć naraz w sekcji krytycznej?

Weryfikacja programów z równoległymi procesami

{x = True }	{łazienka wolna}
1: **goto** 2	idź z korytarza do drzwi łazienki
2: **if** x **then goto** 3 **else goto** 1	jeśli wolna wejdź, jeśli nie wróć na korytarz
3: x = !x	zmień oznaczenie czy wolna
4: # sekcja krytyczna	# łazienka
5: x = !x, **exit**	zmień oznaczenie czy wolna i wyjdź

Do pierwszej linijki może wejść dowolnie wiele procesów

Pytanie: czy dwa procesy mogą się znaleźć naraz w sekcji krytycznej?

Zamodelujemy to sieciami Petriego (d, T)

- wymiar d: każda linijka i możliwe wartości x,

Weryfikacja programów z równoległymi procesami

{x = True }	{łazienka wolna}
1: **goto** 2	idź z korytarza do drzwi łazienki
2: **if** x **then goto** 3 **else goto** 1	jeśli wolna wejdź, jeśli nie wróć na korytarz
3: x = !x	zmień oznaczenie czy wolna
4: # sekcja krytyczna	# łazienka
5: x = !x, **exit**	zmień oznaczenie czy wolna i wyjdź

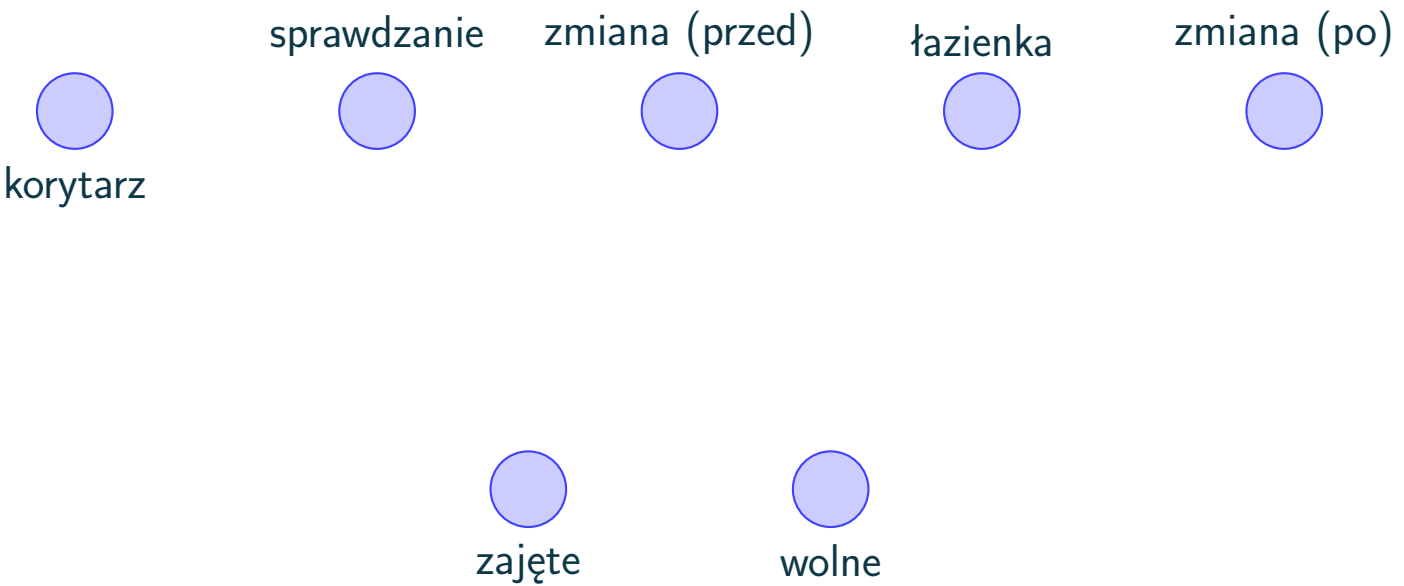
Do pierwszej linijki może wejść dowolnie wiele procesów

Pytanie: czy dwa procesy mogą się znaleźć naraz w sekcji krytycznej?

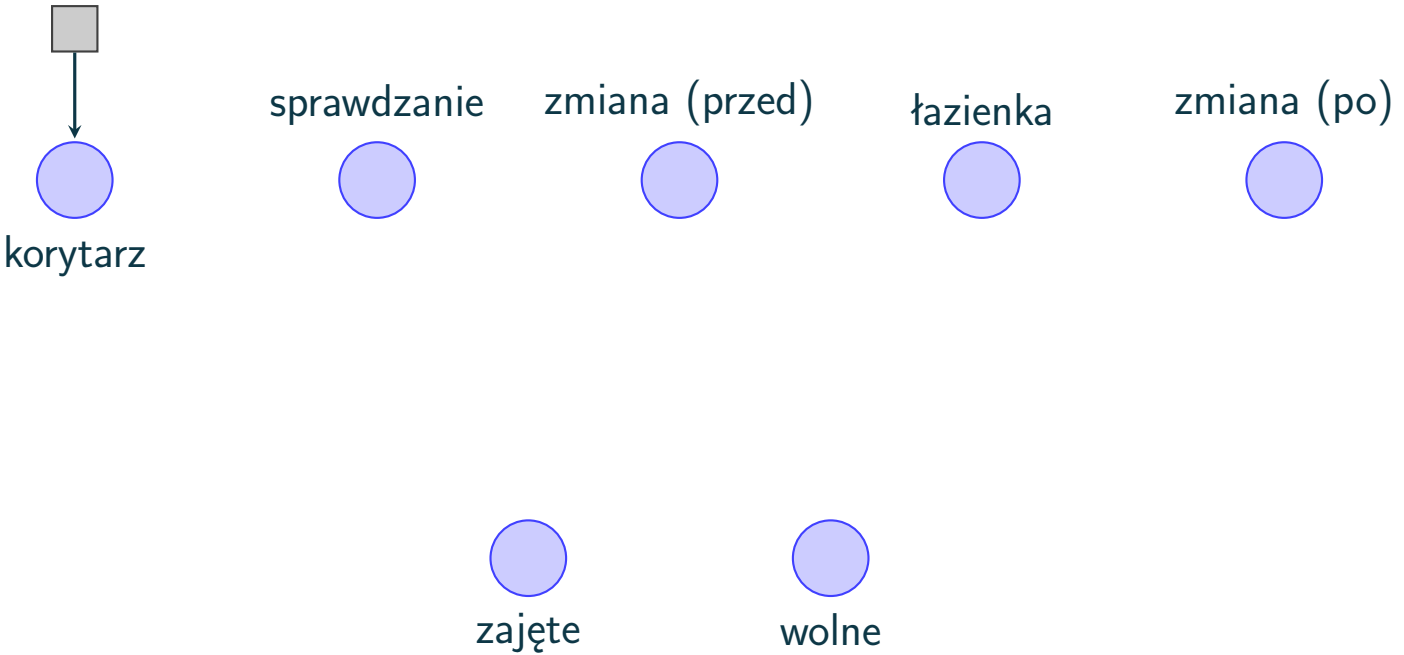
Zamodelujemy to sieciami Petriego (d, T)

- wymiar d: każda linijka i możliwe wartości x,
- Tranzycje: jak ludzie się przemieszczają i zmieniają wartości x

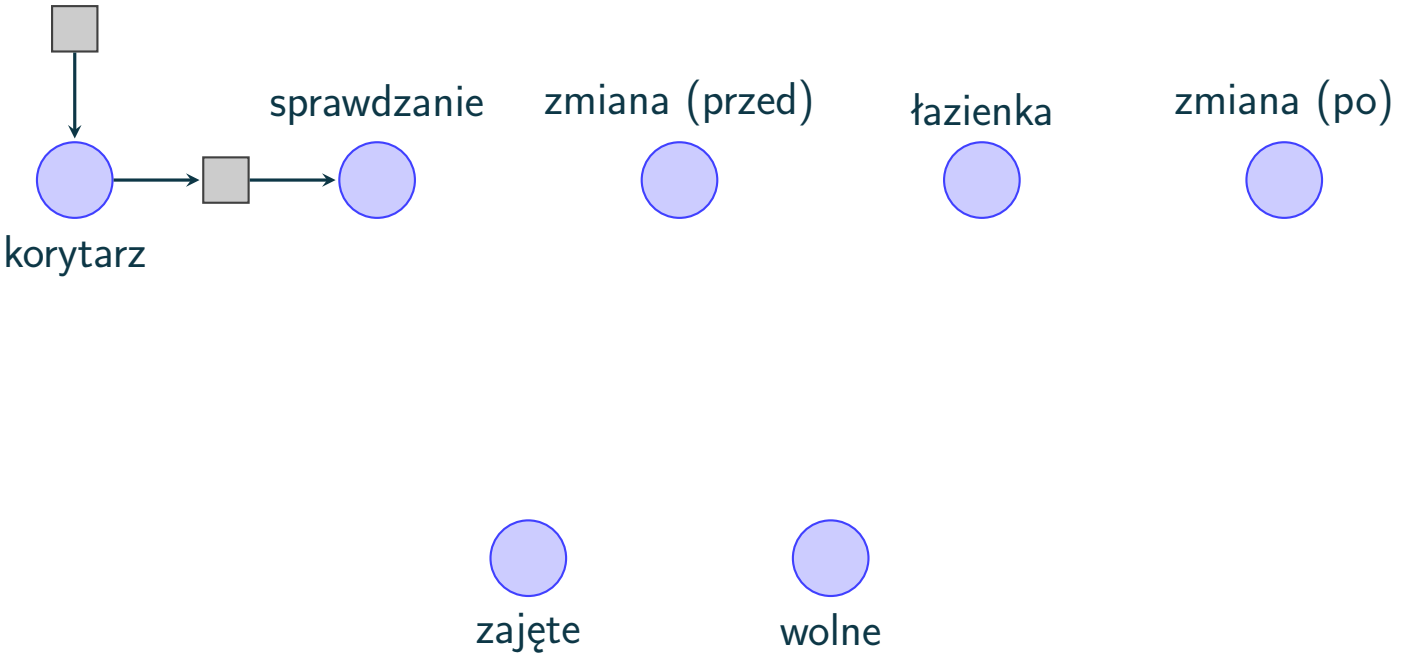
Modelowanie poprzedniego przykładu



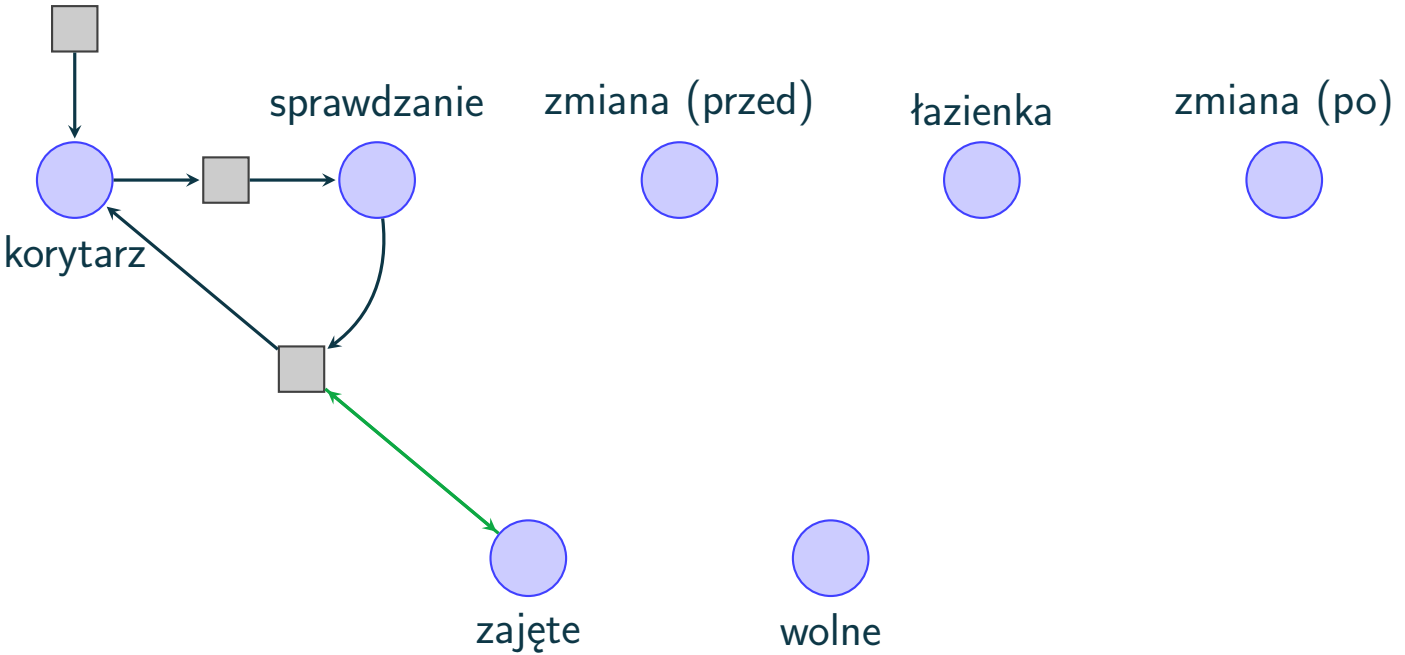
Modelowanie poprzedniego przykładu



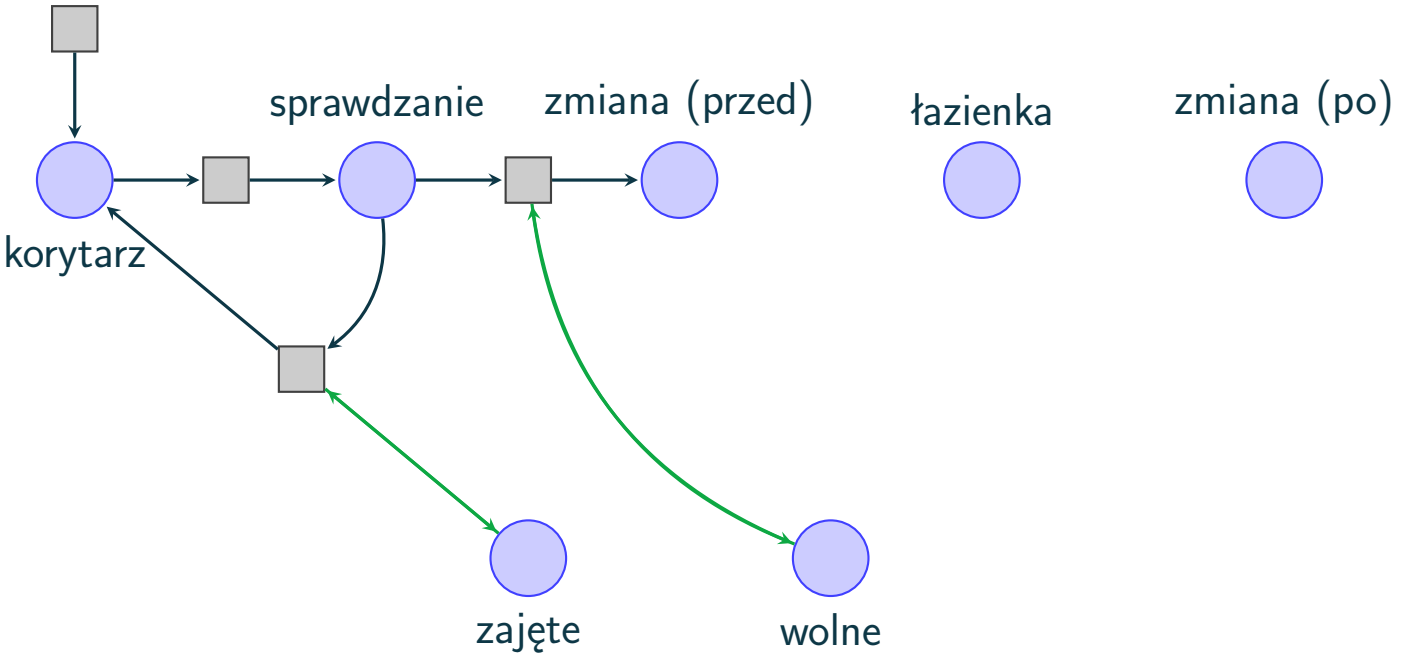
Modelowanie poprzedniego przykładu



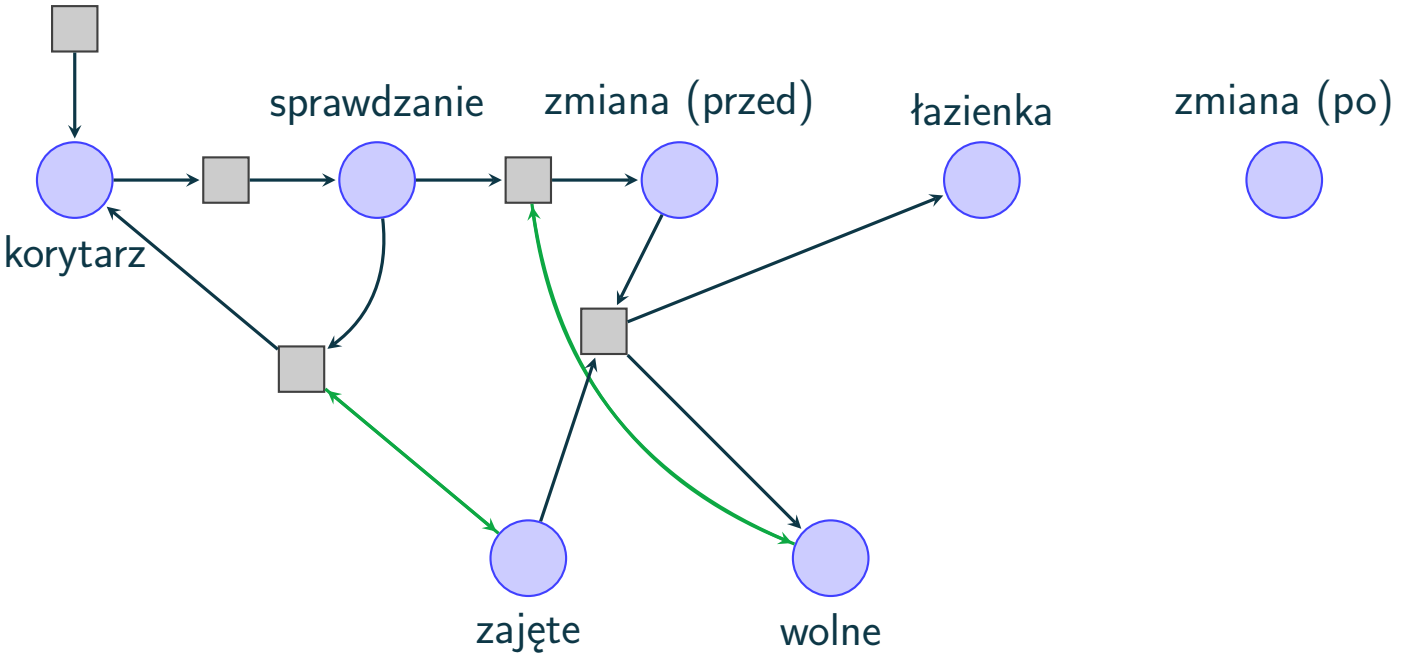
Modelowanie poprzedniego przykładu



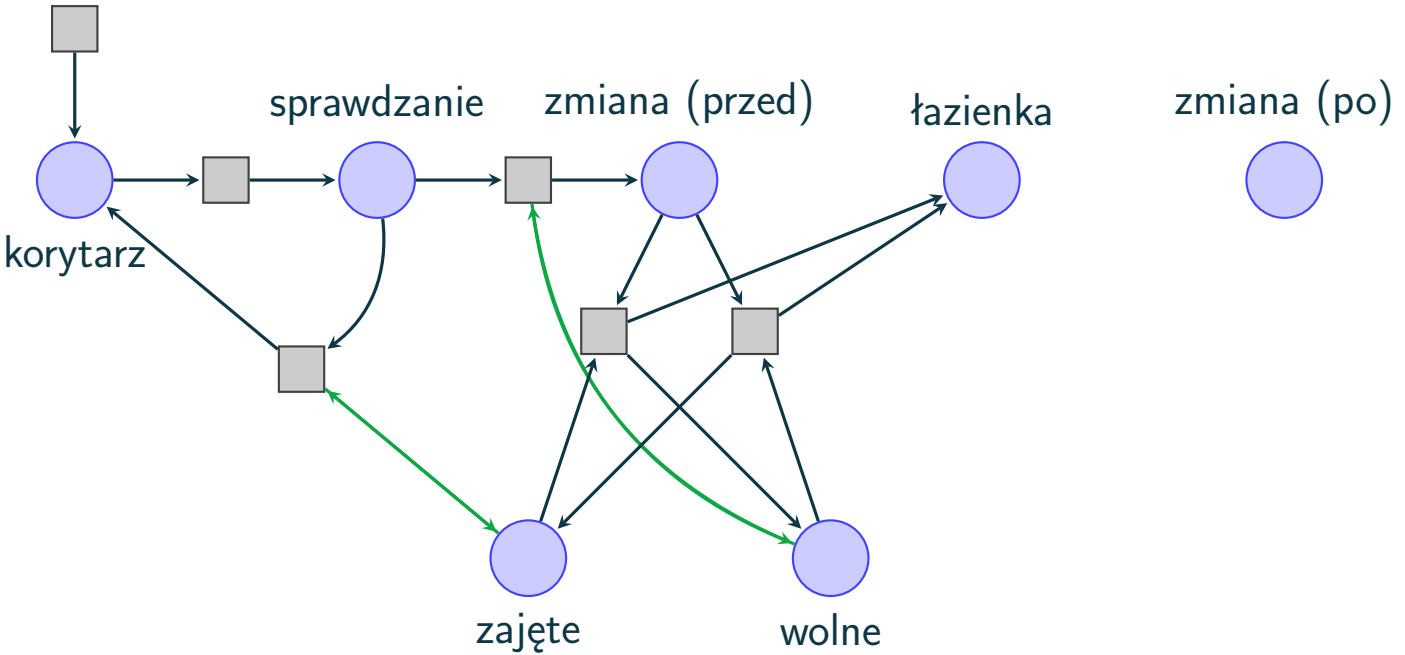
Modelowanie poprzedniego przykładu



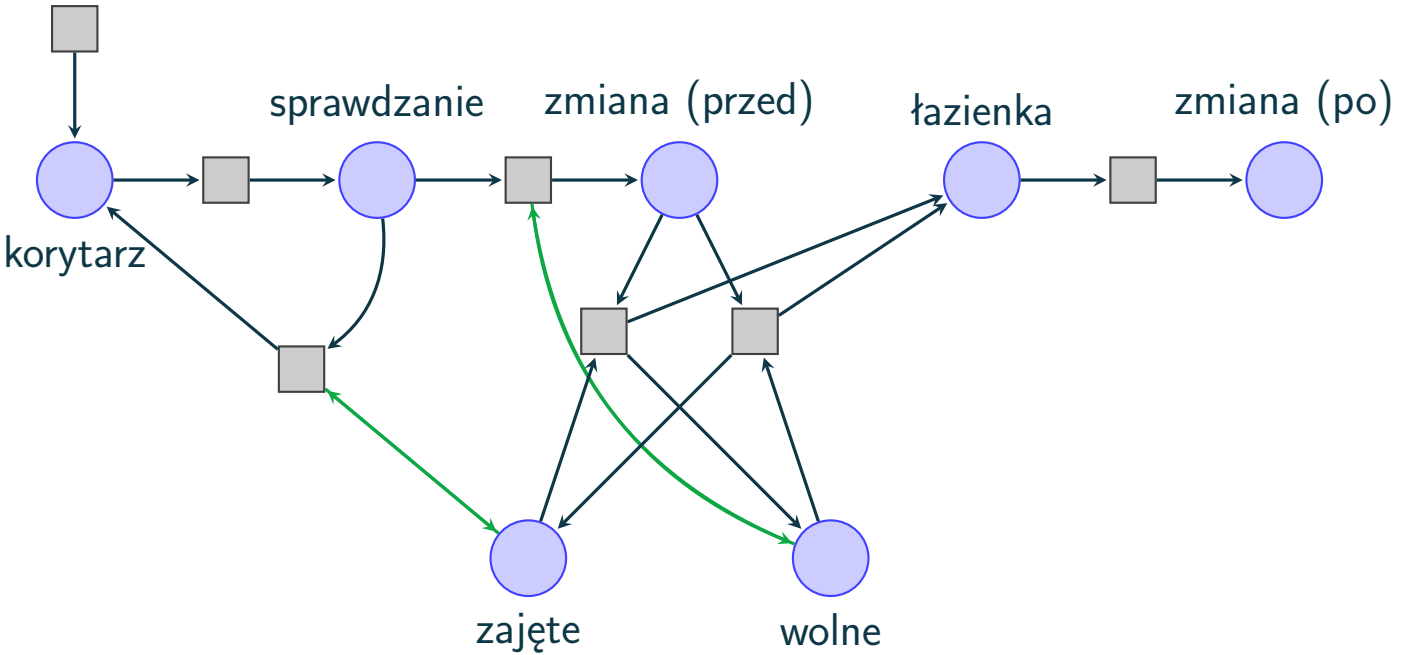
Modelowanie poprzedniego przykładu



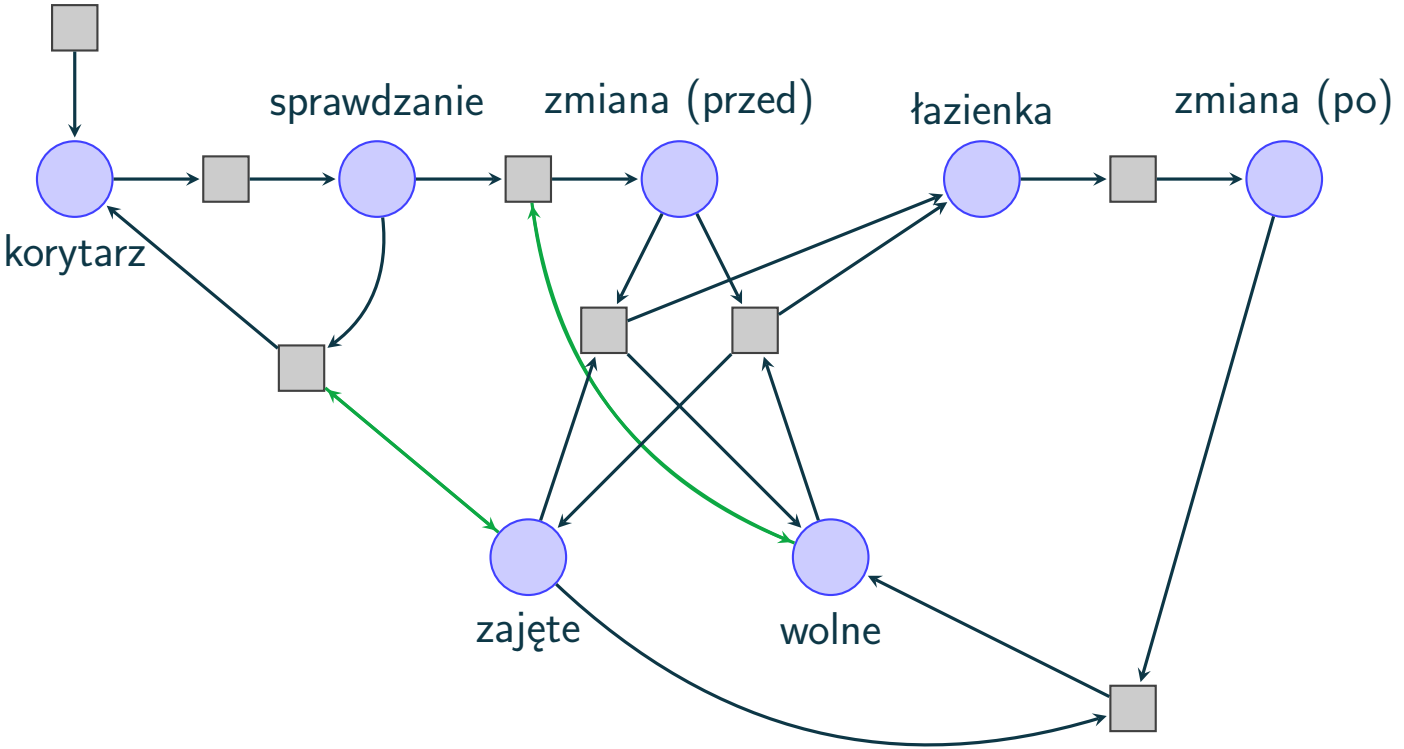
Modelowanie poprzedniego przykładu



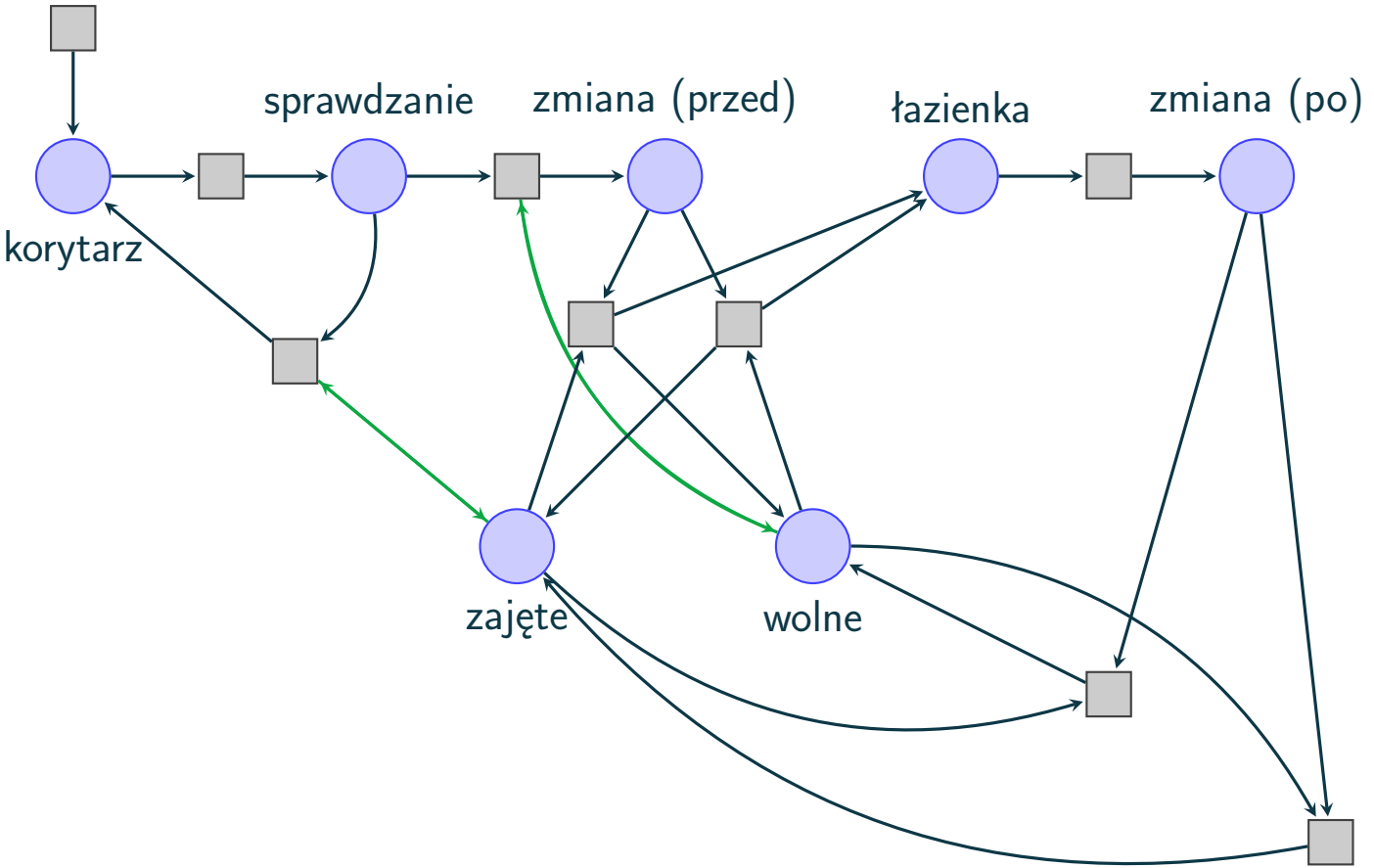
Modelowanie poprzedniego przykładu



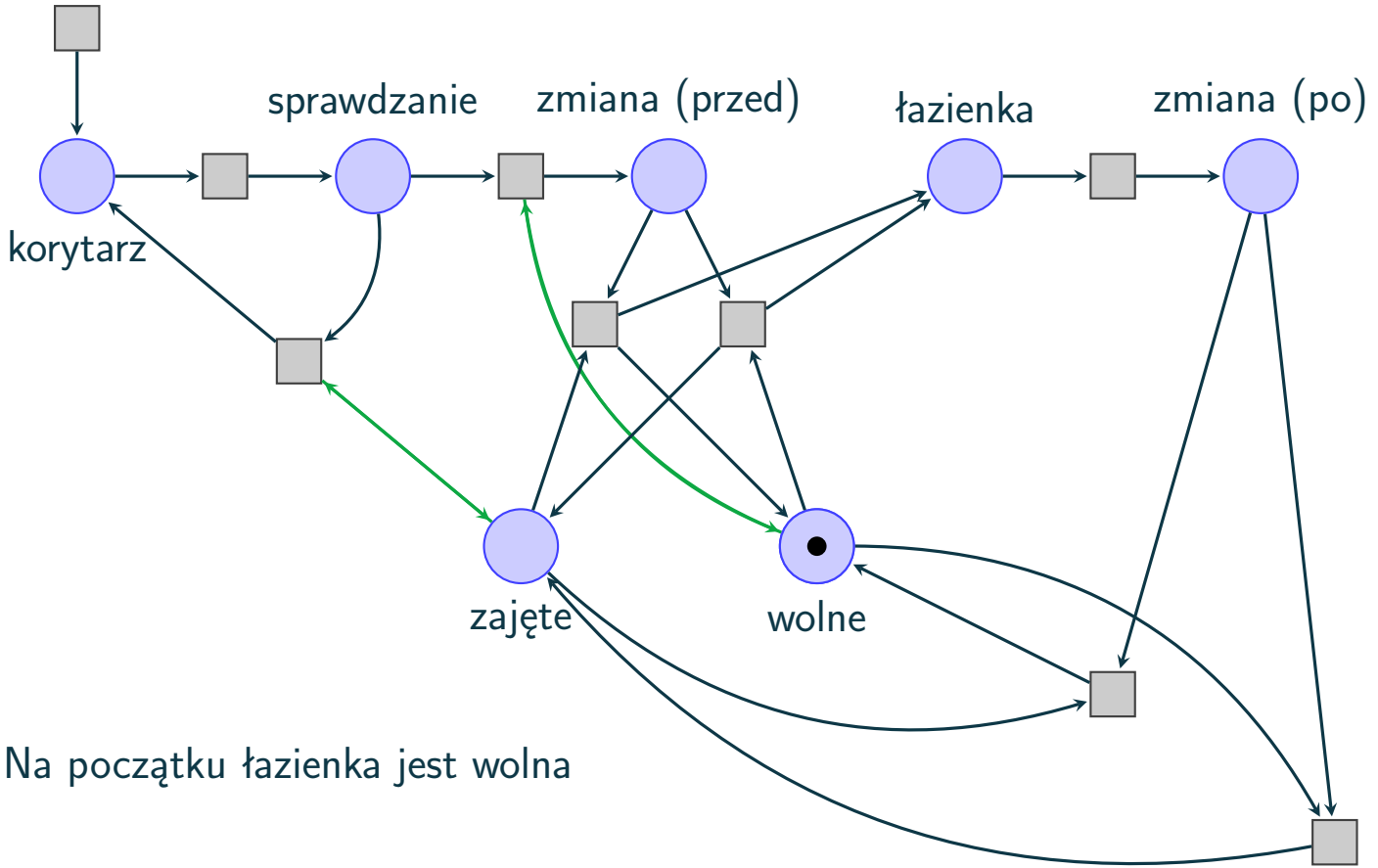
Modelowanie poprzedniego przykładu



Modelowanie poprzedniego przykładu

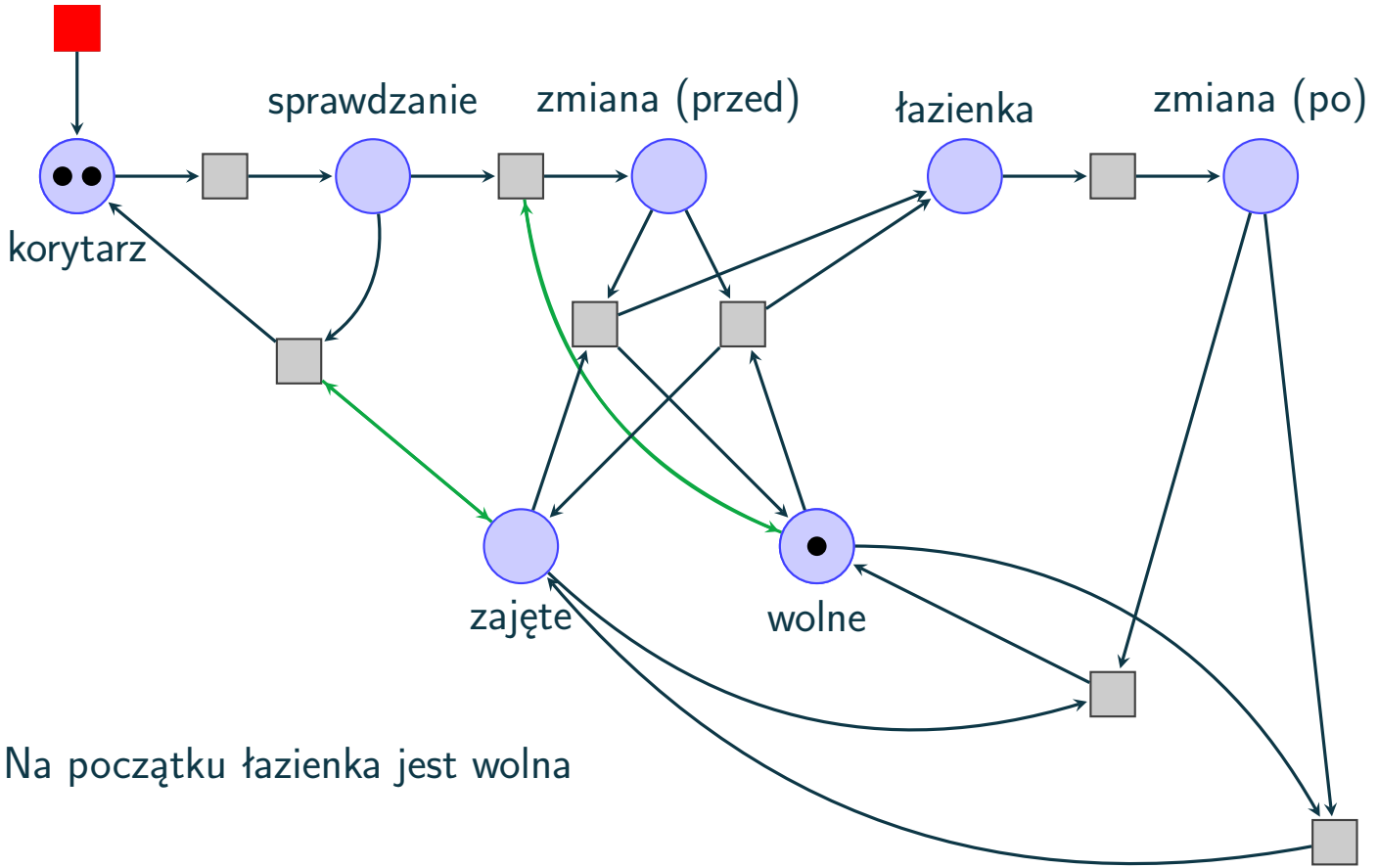


Modelowanie poprzedniego przykładu



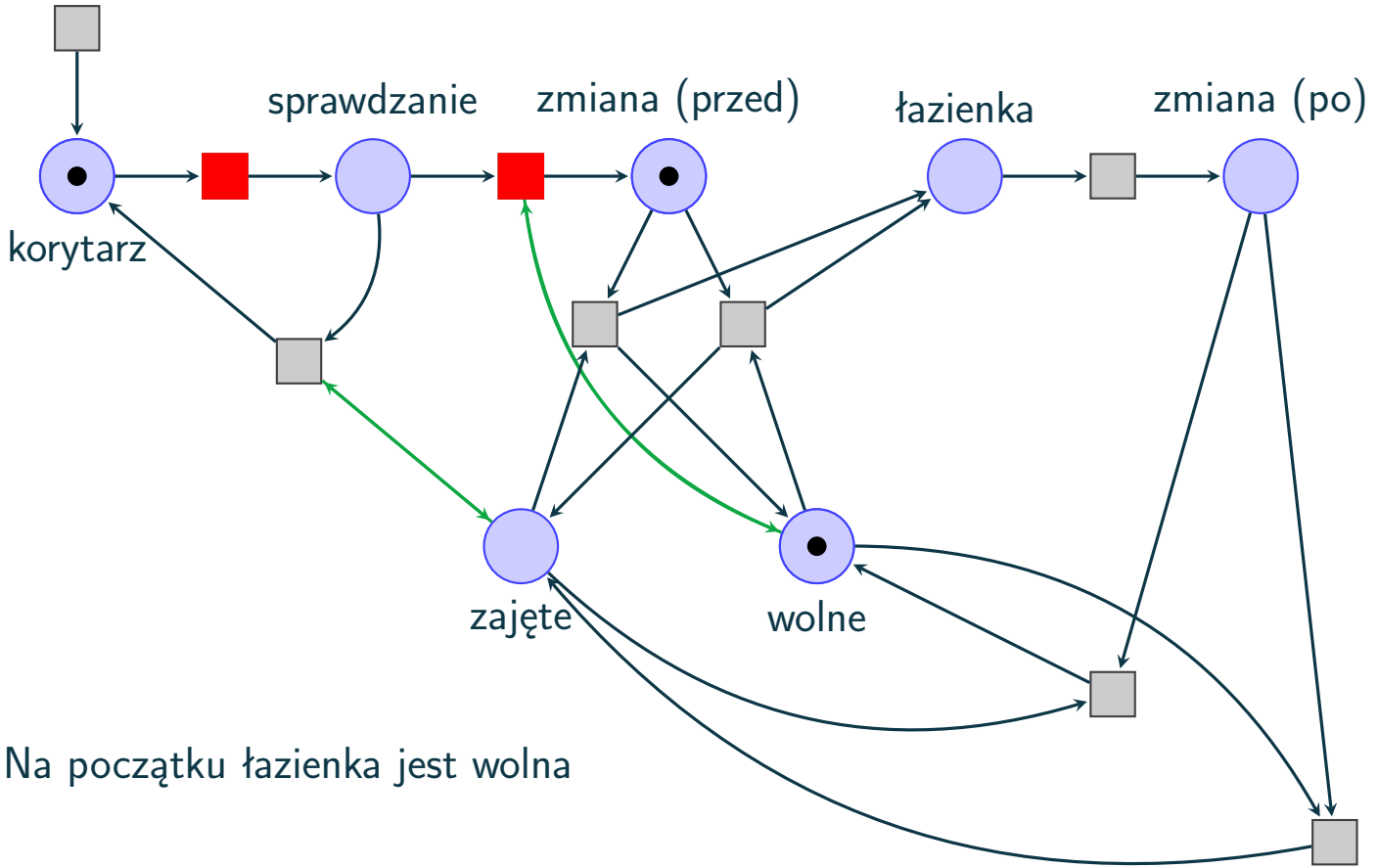
Na początku łazienka jest wolna

Modelowanie poprzedniego przykładu



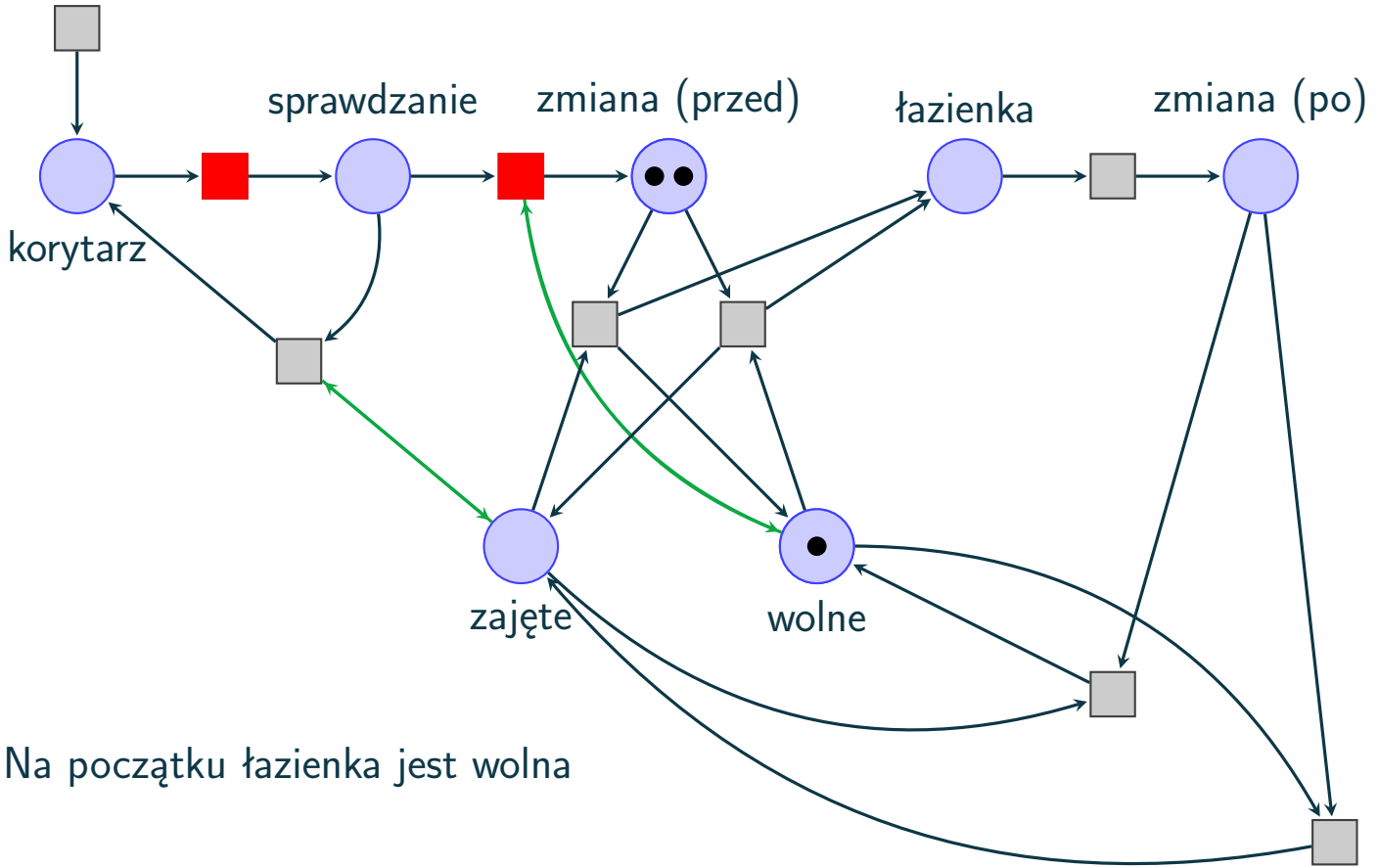
Na początku łazienka jest wolna

Modelowanie poprzedniego przykładu



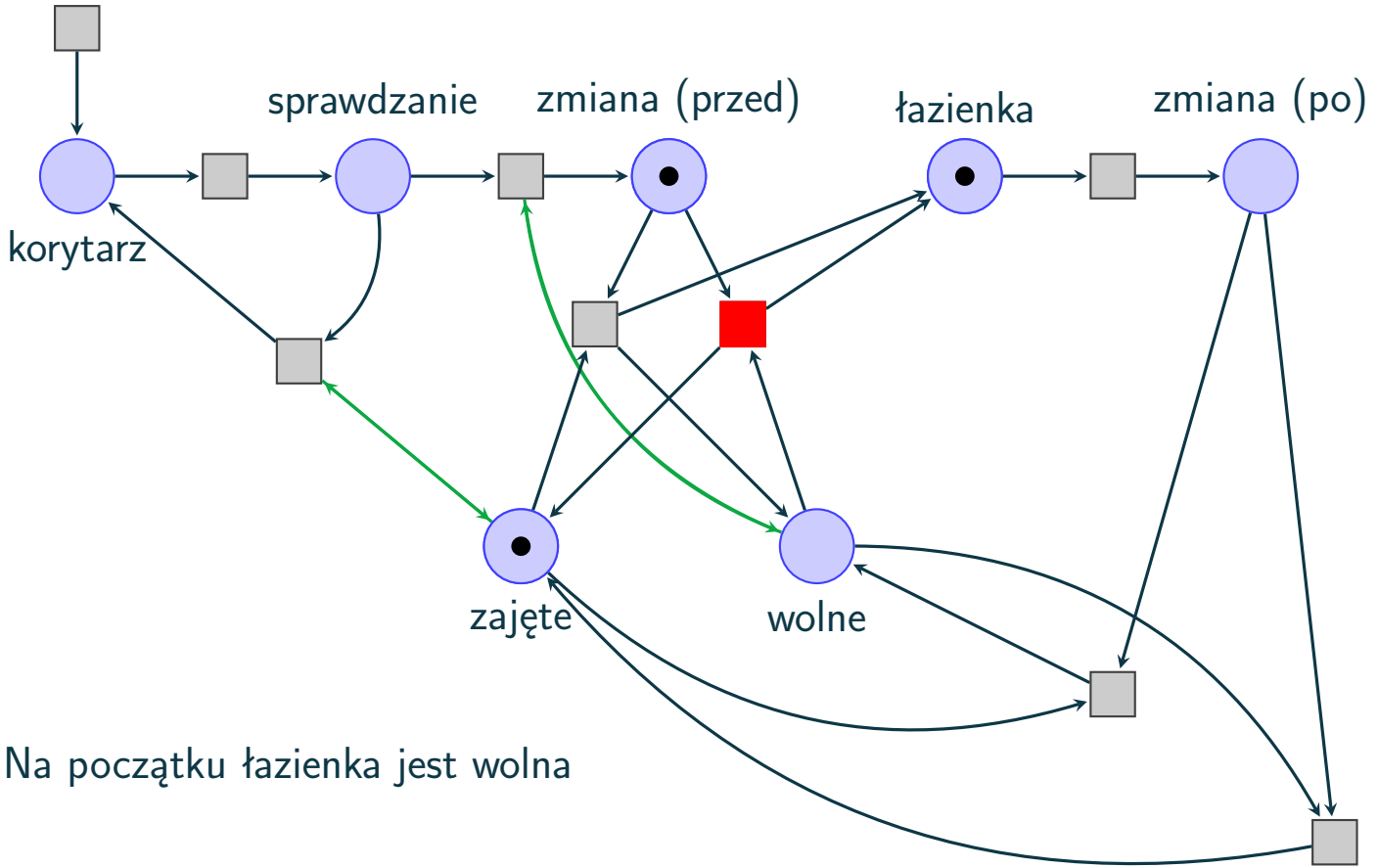
Na początku łazienka jest wolna

Modelowanie poprzedniego przykładu



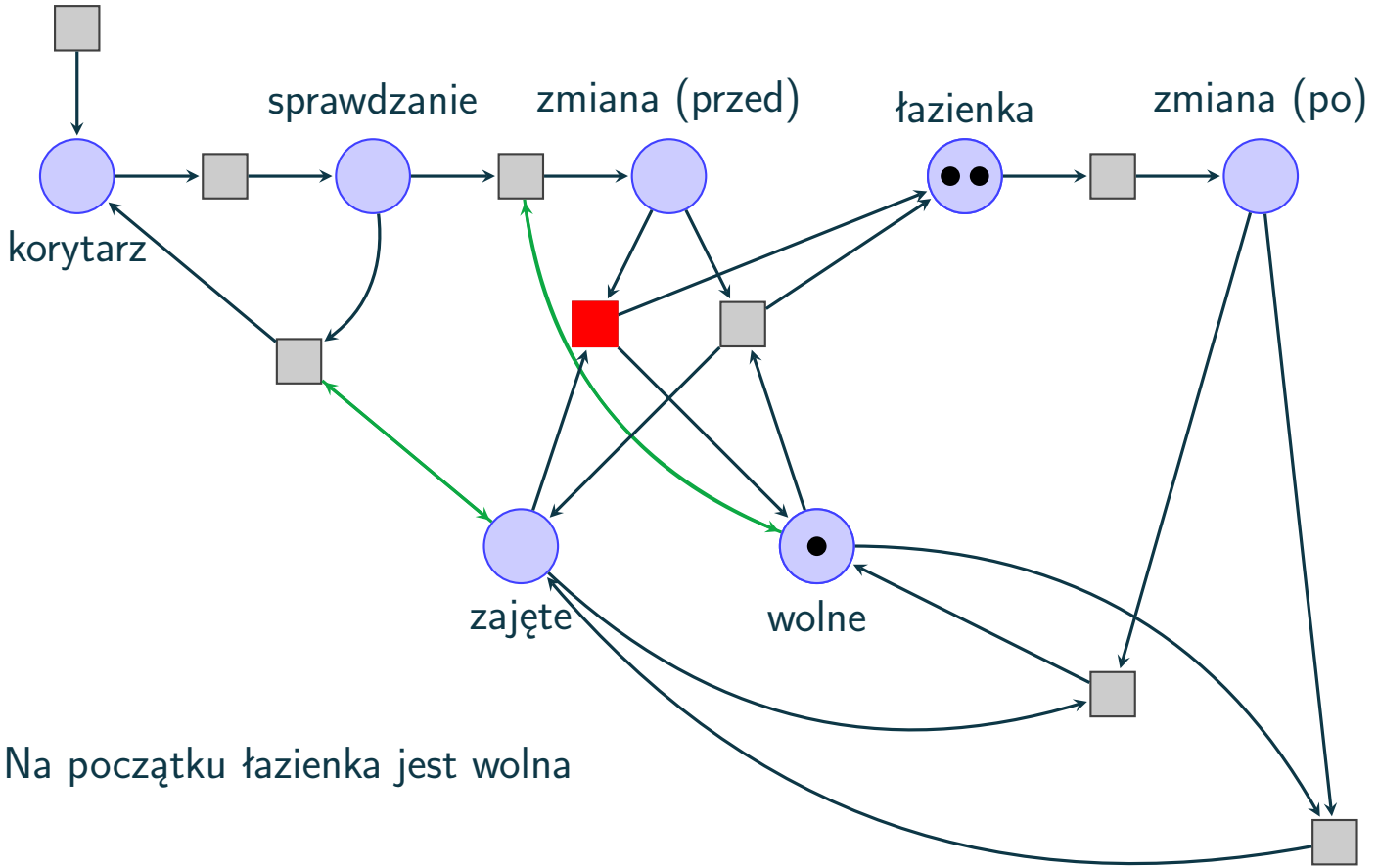
Na początku łazienka jest wolna

Modelowanie poprzedniego przykładu



Na początku łazienka jest wolna

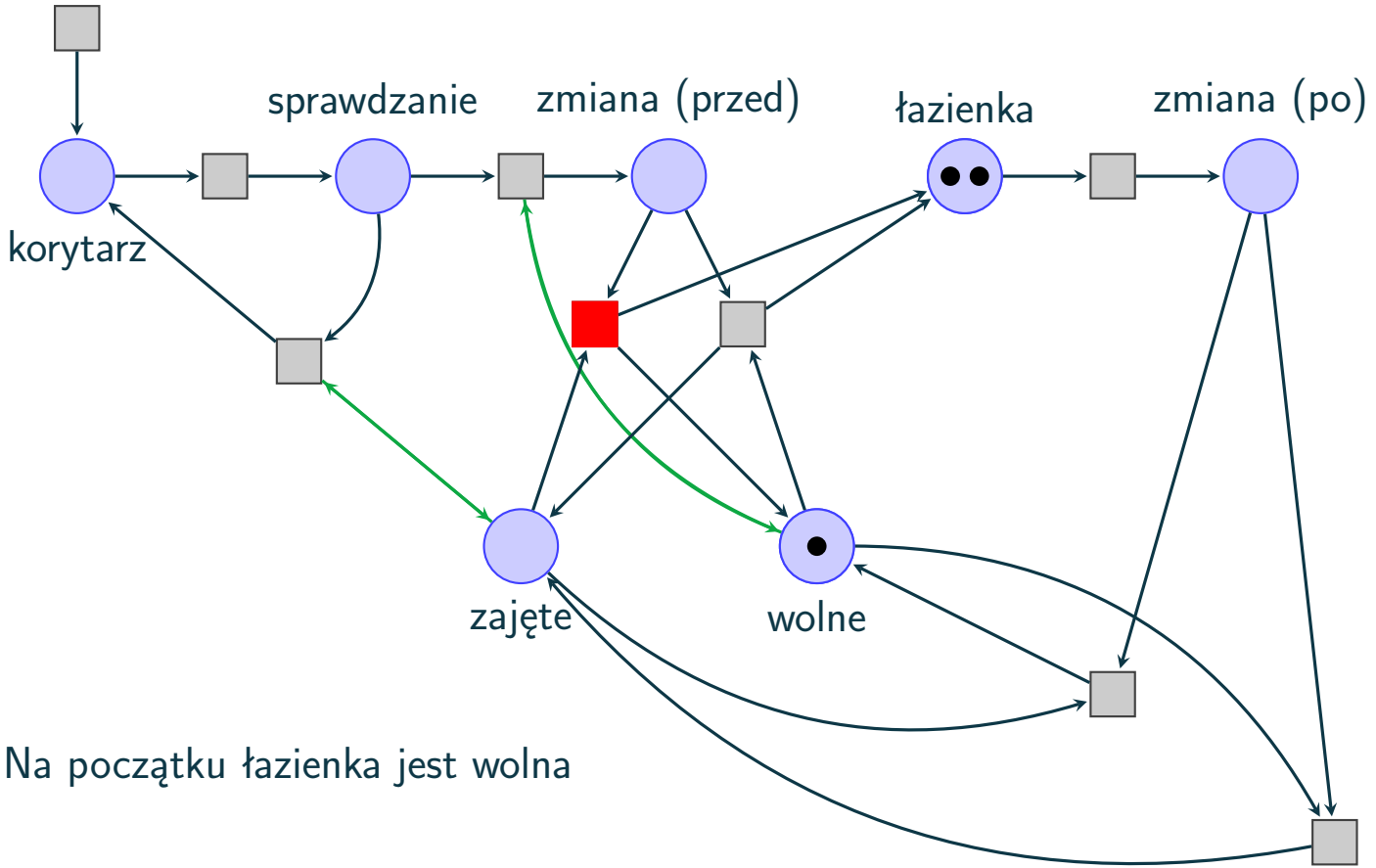
Modelowanie poprzedniego przykładu



Na początku łazienka jest wolna

Znaleźliśmy błąd

Modelowanie poprzedniego przykładu



Na początku łazienka jest wolna

Znaleźliśmy błąd

Ten przykład można naprawić, ale my tylko wykrywamy błędy

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$
- wektor błędu $\boldsymbol{b} = (0, 0, 0, 2, 0, 0, 0)$

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$
- wektor błędu $\boldsymbol{b} = (0, 0, 0, 2, 0, 0, 0)$

Pytanie:

Czy da się przejść z $\boldsymbol{a}$ do $\boldsymbol{c} \geqslant \boldsymbol{b}$

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$
- wektor błędu $\boldsymbol{b} = (0, 0, 0, 2, 0, 0, 0)$

Pytanie:

Czy da się przejść z $\boldsymbol{a}$ do $\boldsymbol{c} \geqslant \boldsymbol{b}$

Poprzednio TAK, doszliśmy do $(0, 0, 0, 2, 0, 1, 0) \geqslant (0, 0, 0, 2, 0, 0, 0)$

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$
- wektor błędu $\boldsymbol{b} = (0, 0, 0, 2, 0, 0, 0)$

Pytanie:

Czy da się przejść z $\boldsymbol{a}$ do $\boldsymbol{c} \geqslant \boldsymbol{b}$

Poprzednio TAK, doszliśmy do $(0, 0, 0, 2, 0, 1, 0) \geqslant (0, 0, 0, 2, 0, 0, 0)$

To się nazywa problem pokrywalności

Problem pokrywalności

Co właściwie chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- wektor początkowy $\boldsymbol{a} = (0, 0, 0, 0, 0, 1, 0)$
- wektor błędu $\boldsymbol{b} = (0, 0, 0, 2, 0, 0, 0)$

Pytanie:

Czy da się przejść z $\boldsymbol{a}$ do $\boldsymbol{c} \geqslant \boldsymbol{b}$

Poprzednio TAK, doszliśmy do $(0, 0, 0, 2, 0, 1, 0) \geqslant (0, 0, 0, 2, 0, 0, 0)$

To się nazywa problem pokrywalności

Czy jest to trudny problem?

Jeśli można pokryć $\boldsymbol{b}$ to ile kroków z $\boldsymbol{a}$ trzeba wykonać?

Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

Procesy biznesowe

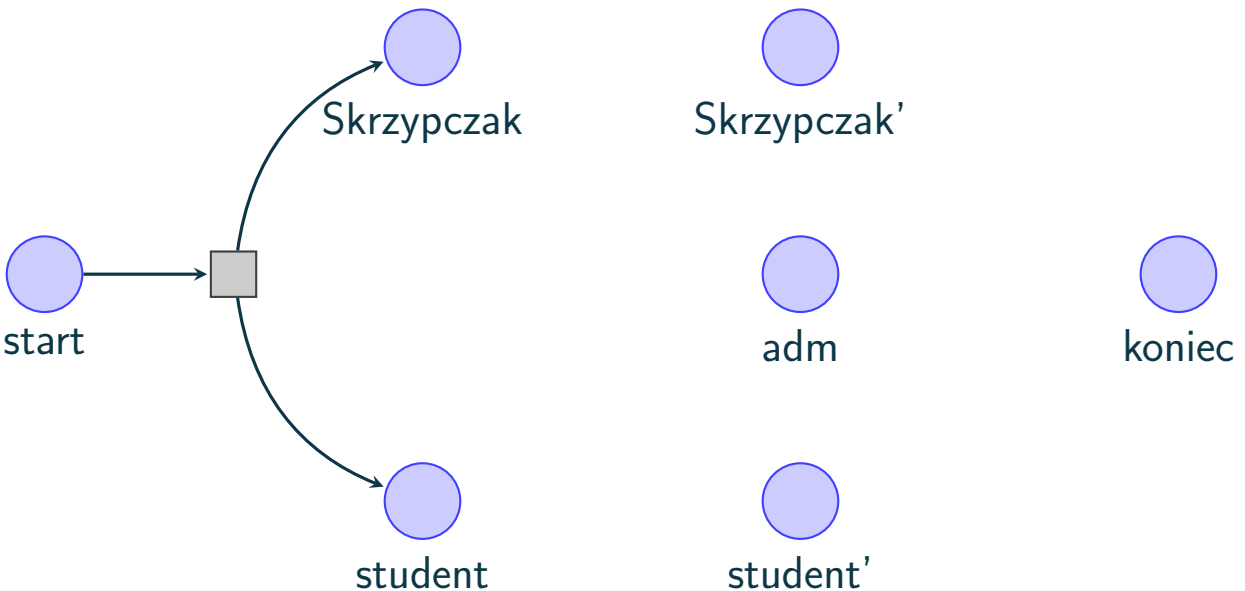
Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności

Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

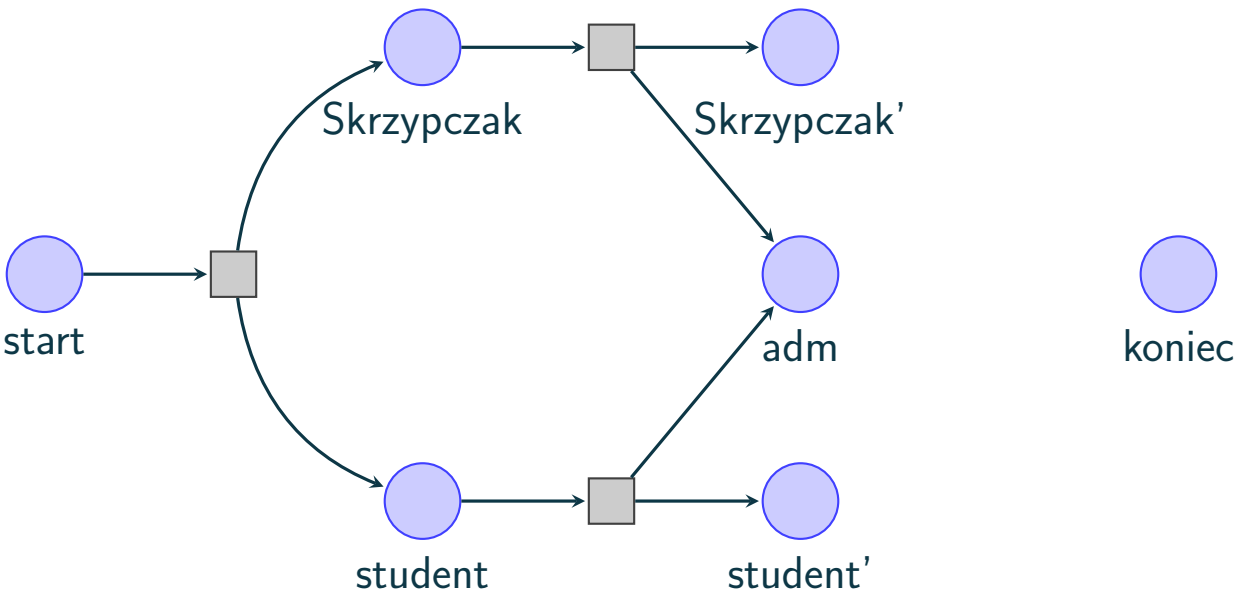
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

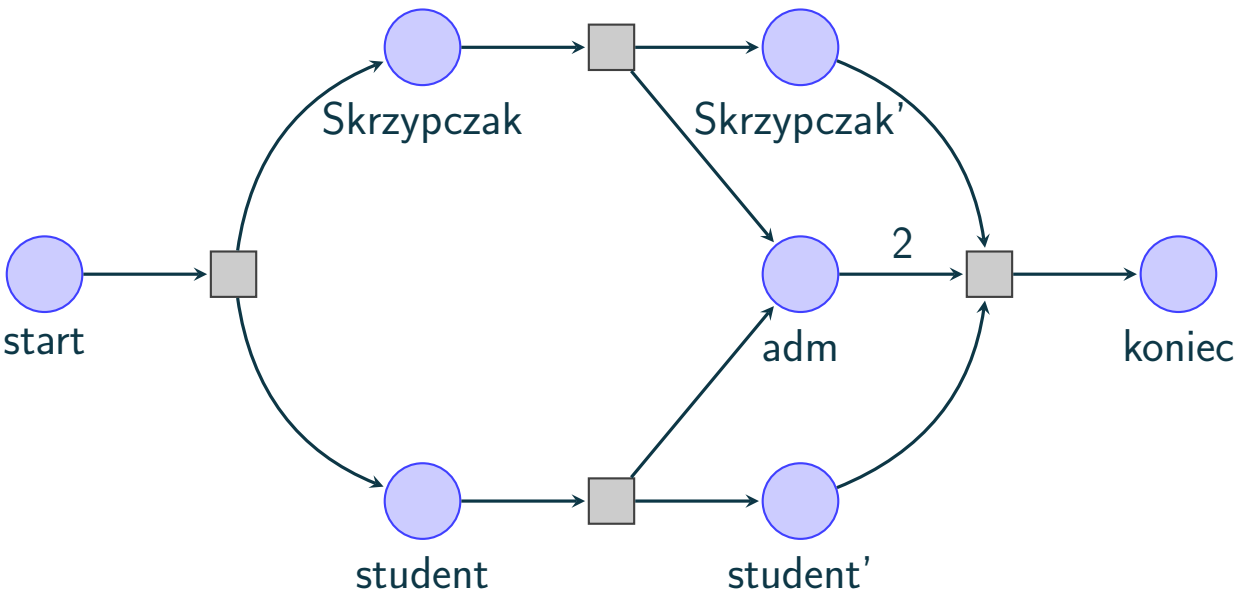
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

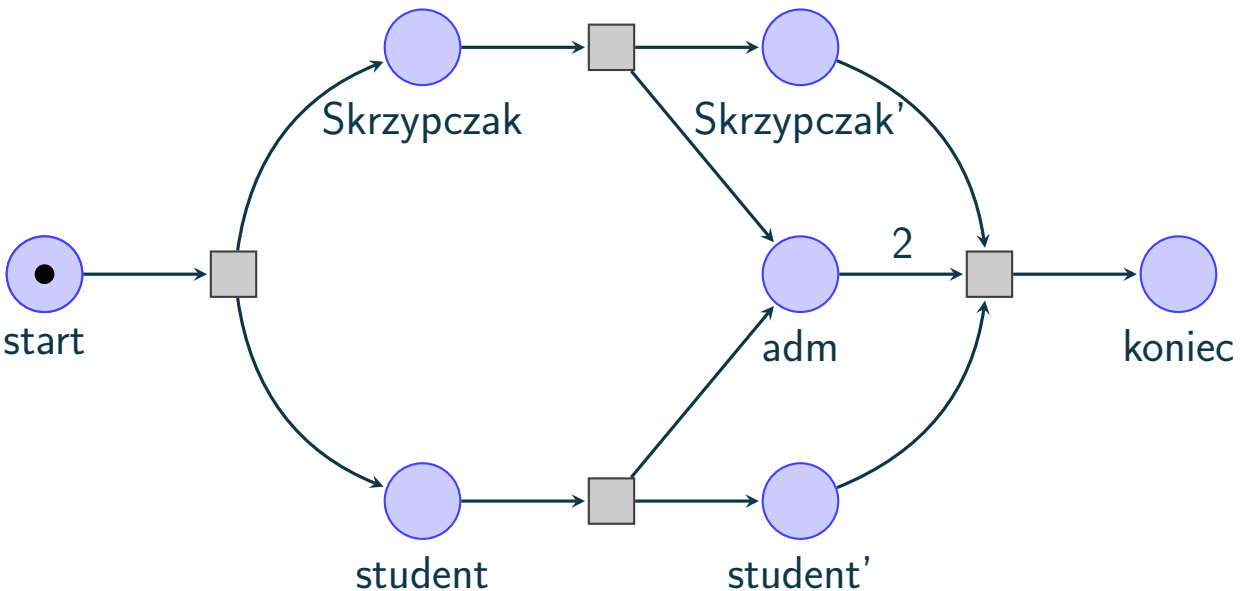
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

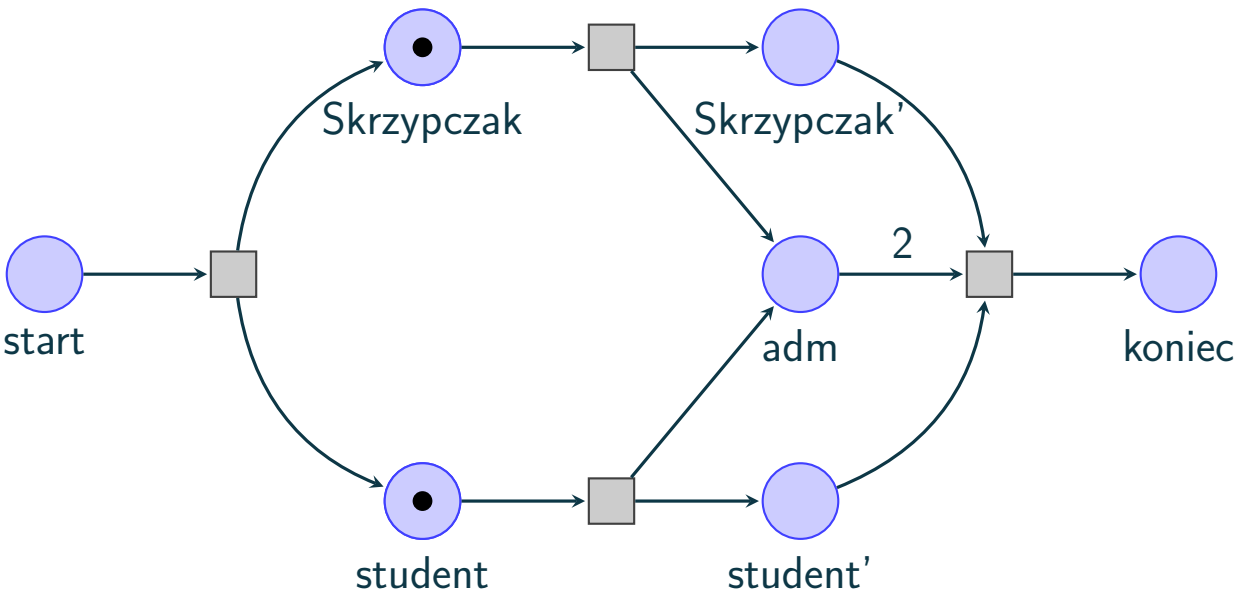
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

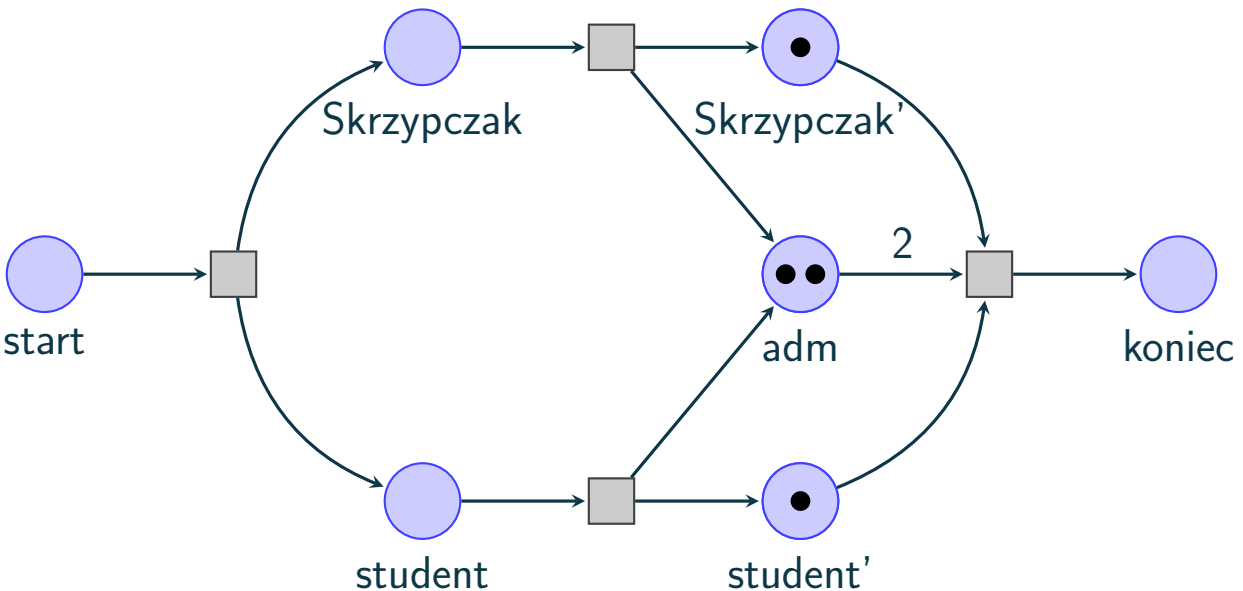
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

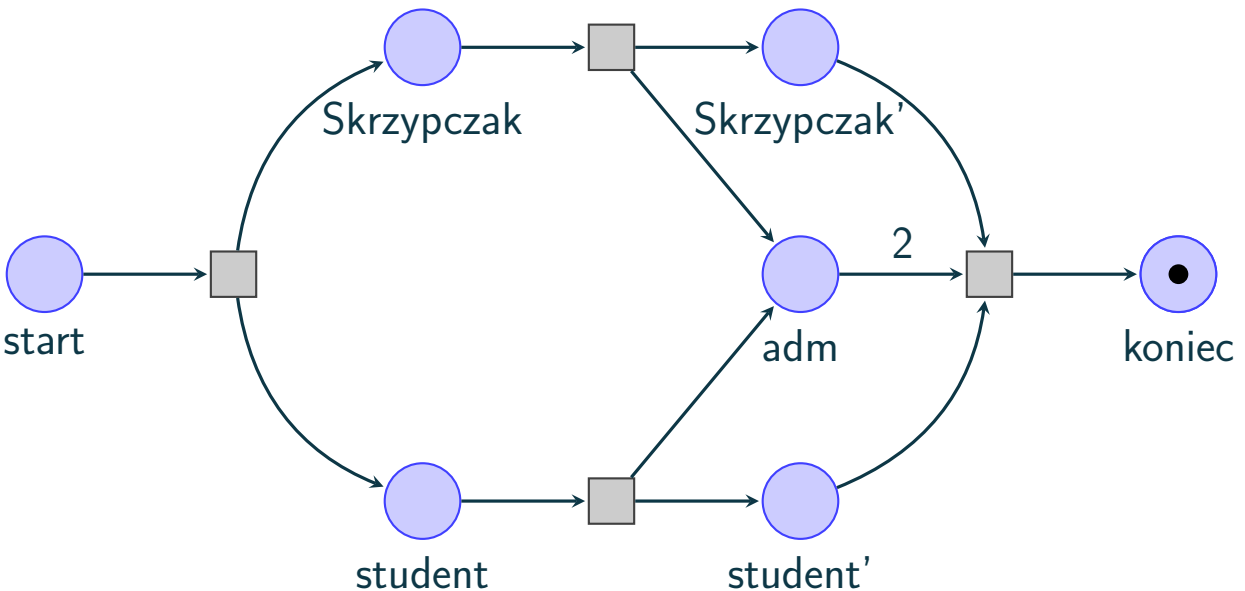
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

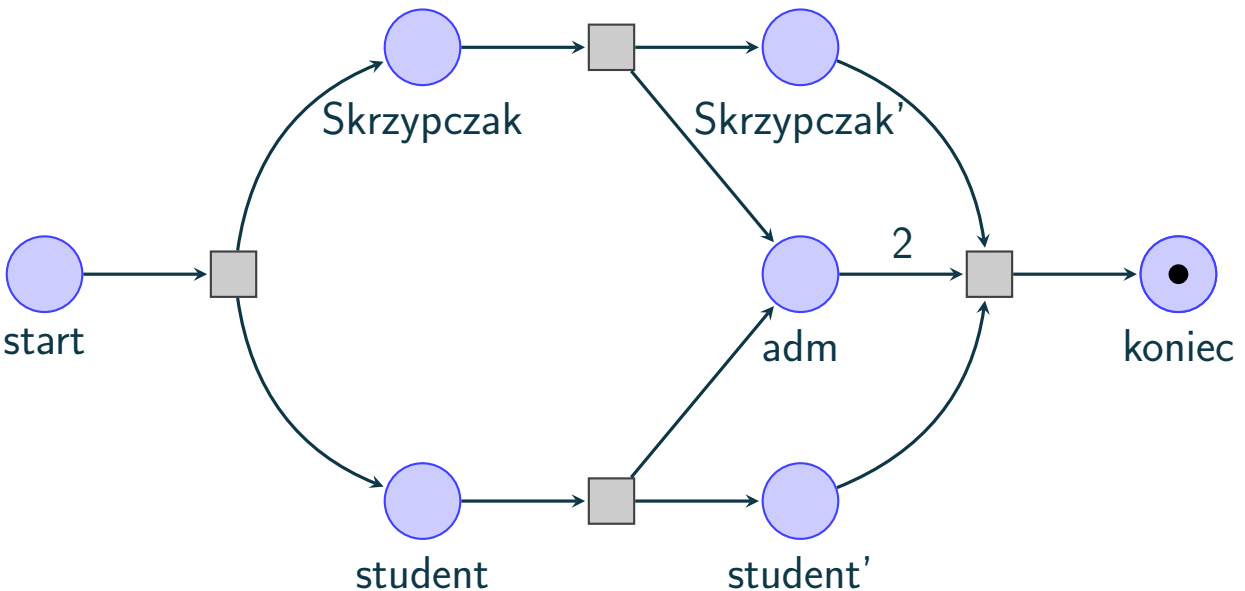
- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Procesy biznesowe

Przypuśćmy, że profesor Skrzypczak chce zatrudnić doktoranta

- Po udanej rozmowie kwalifikacyjnej trzeba załatwić formalności



Zachowuje się dobrze, nawet przy zatrudnianiu kilku studentów naraz

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- start: $(1, 0, 0, 0, 0, 0, 0)$ i koniec $(0, 0, 0, 0, 0, 0, 1)$

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- start: $(1, 0, 0, 0, 0, 0, 0)$ i koniec $(0, 0, 0, 0, 0, 0, 1)$

Pytanie:

Czy gdziekolwiek wyjdziemy ze startu, uda się dojść do końca?

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- start: $(1, 0, 0, 0, 0, 0, 0)$ i koniec $(0, 0, 0, 0, 0, 0, 1)$

Pytanie:

Czy gdziekolwiek wyjdziemy ze startu, uda się dojść do końca?

Jeśli zaczniemy z $(k, 0, 0, 0, 0, 0, 0)$, czy zawsze dojdziemy do $(0, 0, 0, 0, 0, 0, k)$?

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- start: $(1, 0, 0, 0, 0, 0, 0)$ i koniec $(0, 0, 0, 0, 0, 0, 1)$

Pytanie:

Czy gdziekolwiek wyjdziemy ze startu, uda się dojść do końca?

Jeśli zaczniemy z $(k, 0, 0, 0, 0, 0, 0)$, czy zawsze dojdziemy do $(0, 0, 0, 0, 0, 0, k)$?

To są: problem poprawności i problem k-poprawności

Problem poprawności

Co chcieliśmy sprawdzić w poprzednim przykładzie?

Dane:

- sieć Petriego (d, T)
- start: $(1, 0, 0, 0, 0, 0, 0)$ i koniec $(0, 0, 0, 0, 0, 0, 1)$

Pytanie:

Czy gdziekolwiek wyjdziemy ze startu, uda się dojść do końca?

Jeśli zaczniemy z $(k, 0, 0, 0, 0, 0, 0)$, czy zawsze dojdziemy do $(0, 0, 0, 0, 0, 0, k)$?

To są: problem poprawności i problem k-poprawności

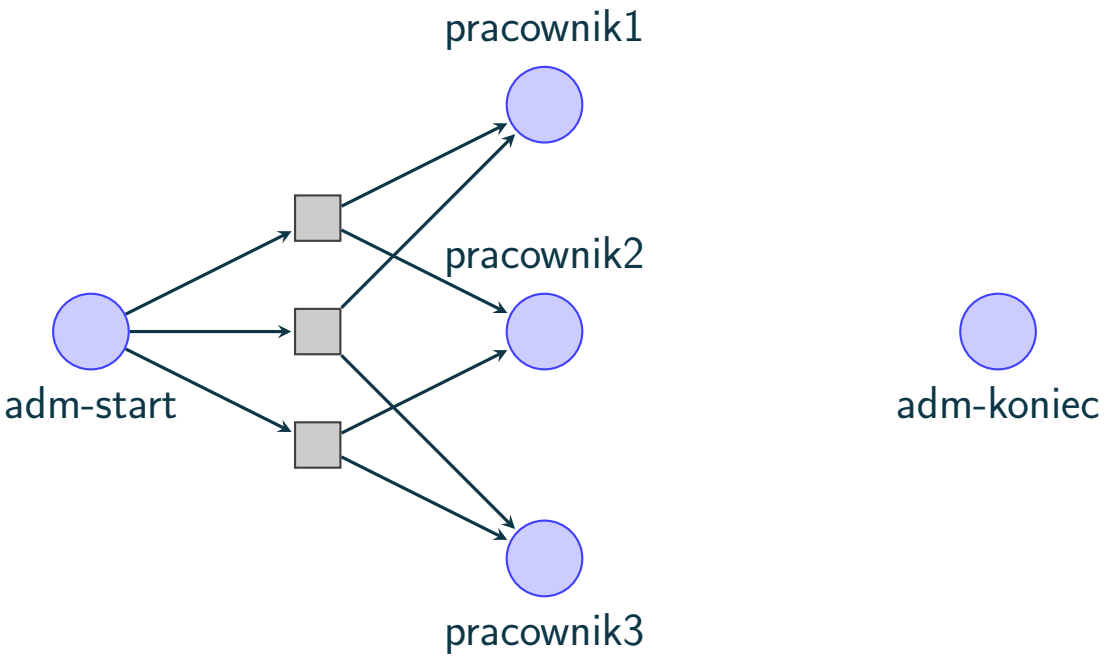
Poprzedni przykład jest poprawny i nawet k-poprawny dla dowolnego k

Rozbudowany przykład

Przypuśćmy, że administracja ma swoje własne procesy

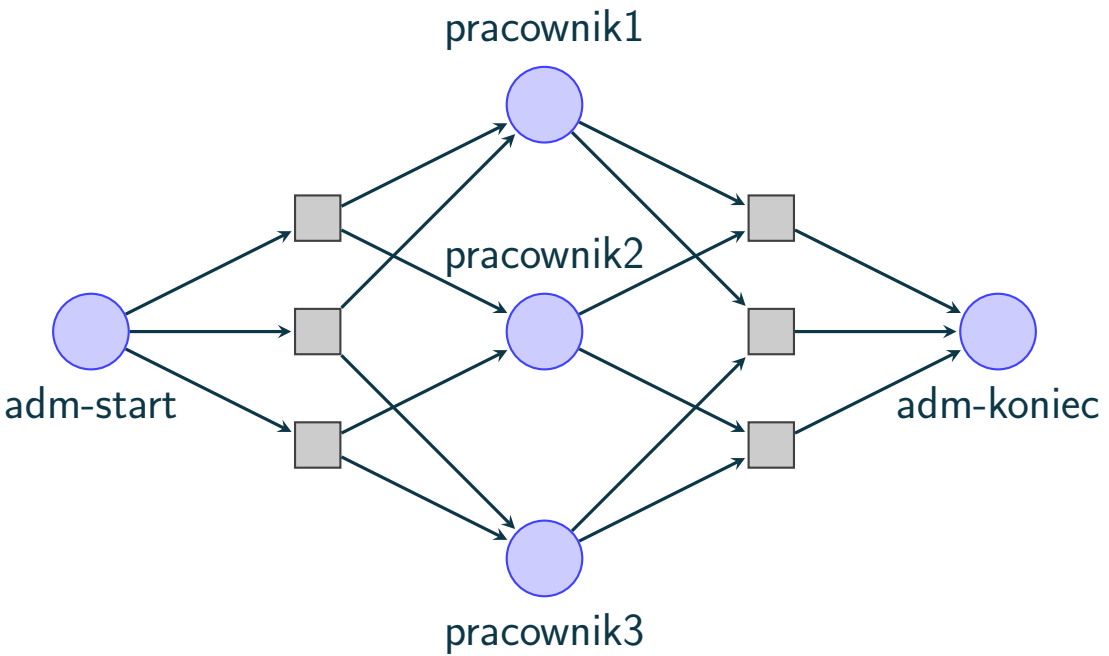
Rozbudowany przykład

Przypuśćmy, że administracja ma swoje własne procesy



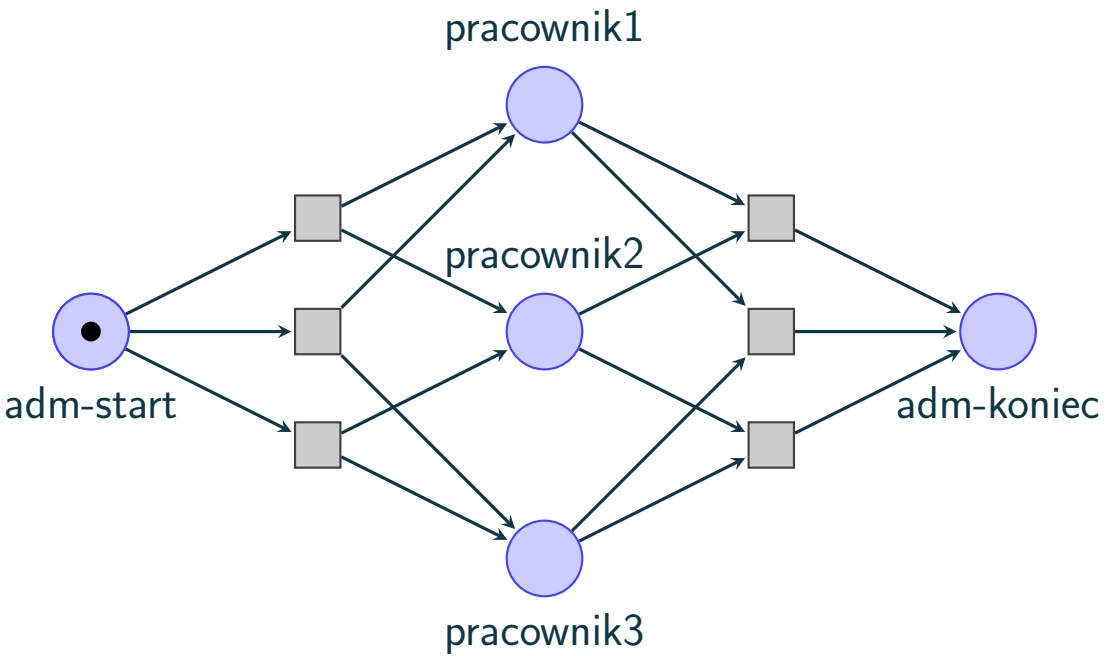
Rozbudowany przykład

Przypuśćmy, że administracja ma swoje własne procesy



Rozbudowany przykład

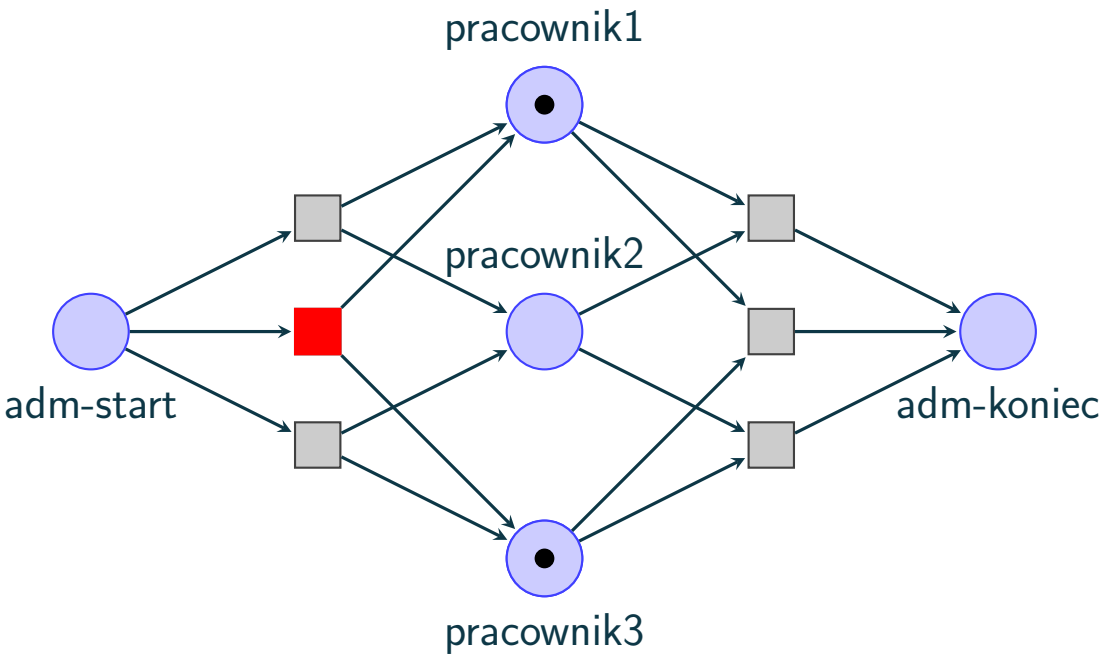
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne

Rozbudowany przykład

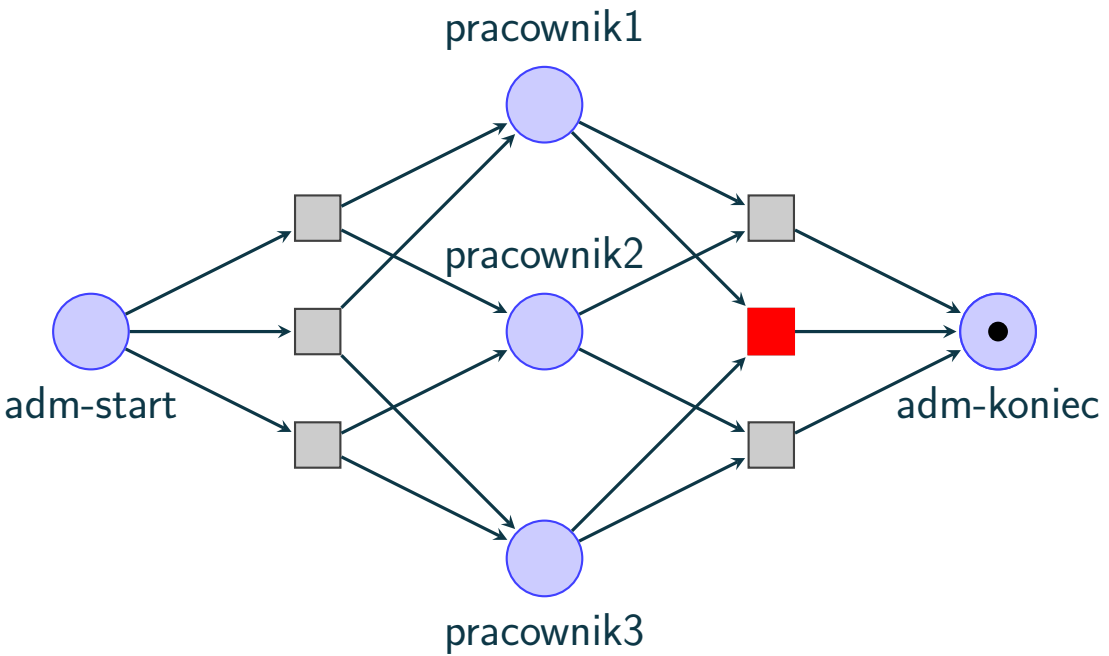
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne

Rozbudowany przykład

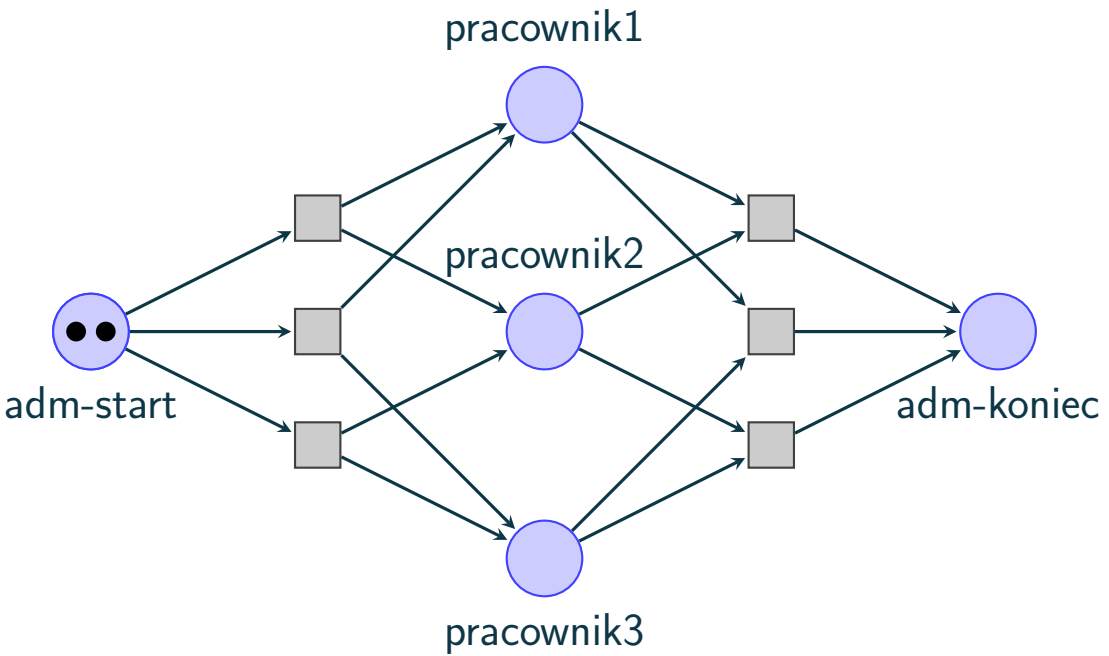
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne

Rozbudowany przykład

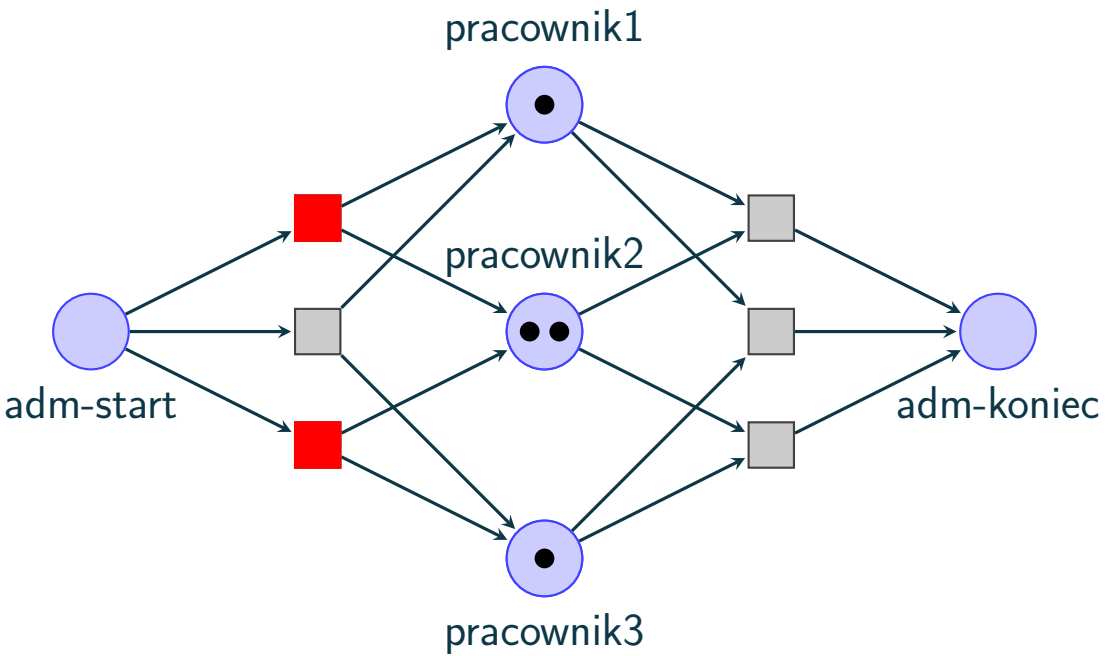
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne
- Ale nie jest 2-poprawne

Rozbudowany przykład

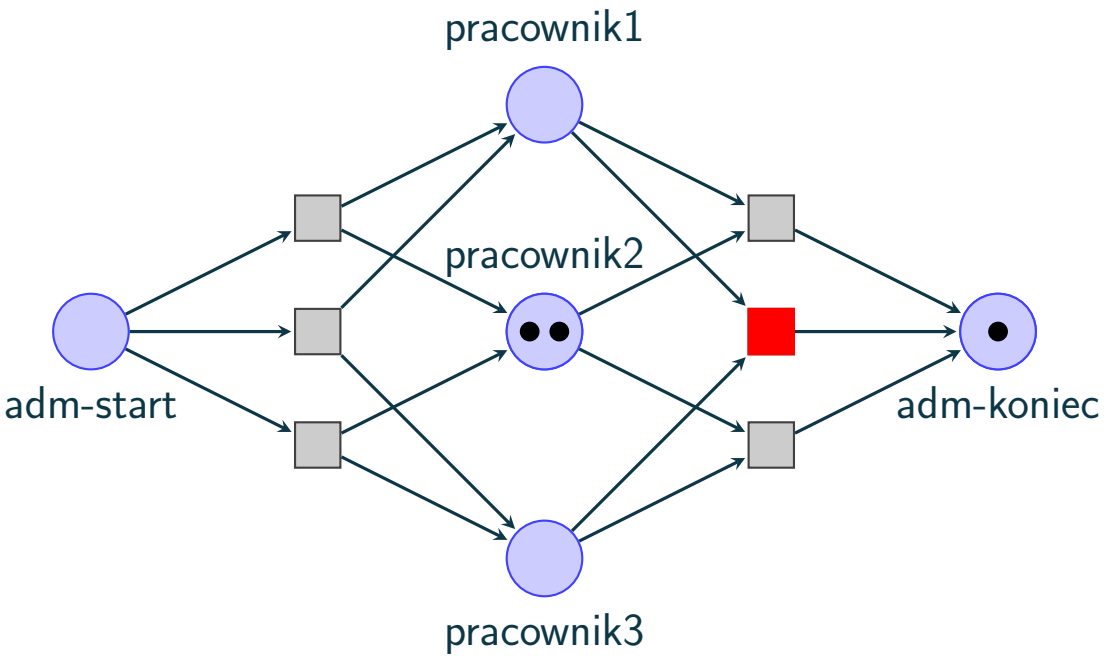
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne
- Ale nie jest 2-poprawne

Rozbudowany przykład

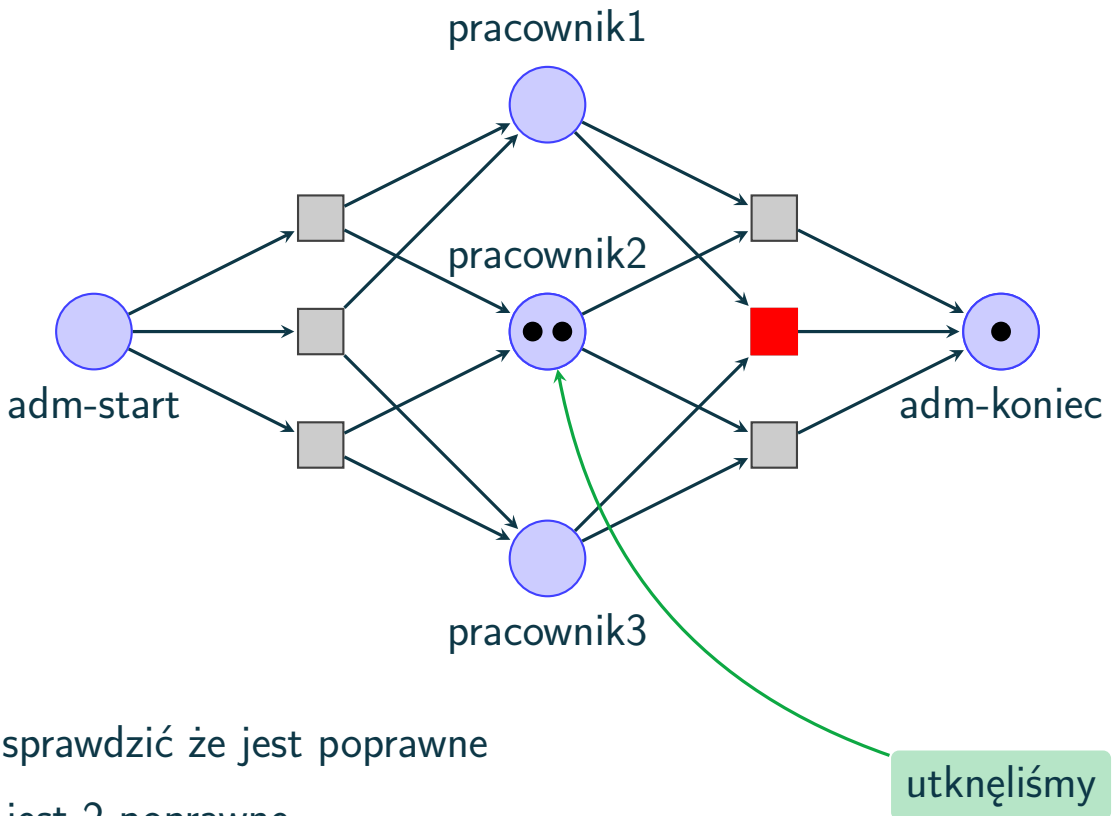
Przypuśćmy, że administracja ma swoje własne procesy



- Można sprawdzić że jest poprawne
- Ale nie jest 2-poprawne

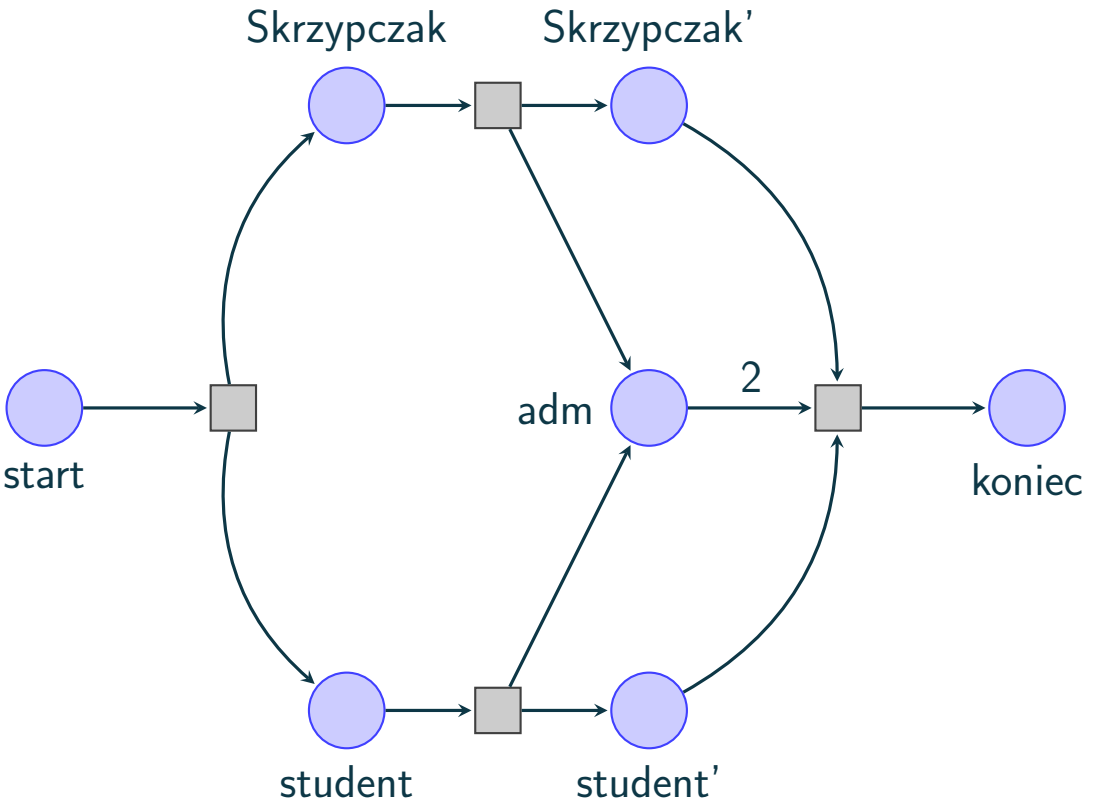
Rozbudowany przykład

Przypuśćmy, że administracja ma swoje własne procesy

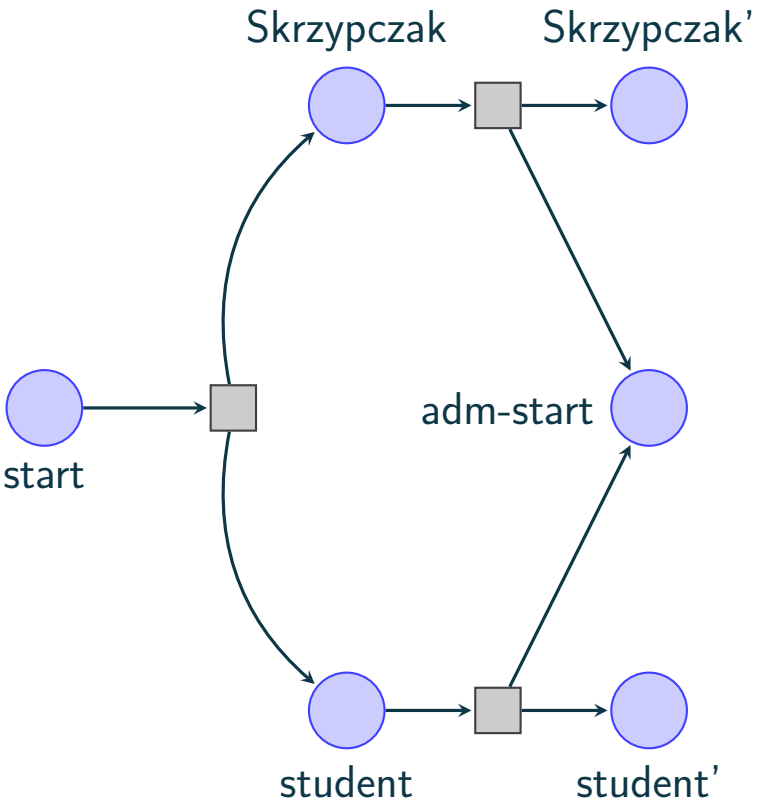


- Można sprawdzić że jest poprawne
- Ale nie jest 2-poprawne

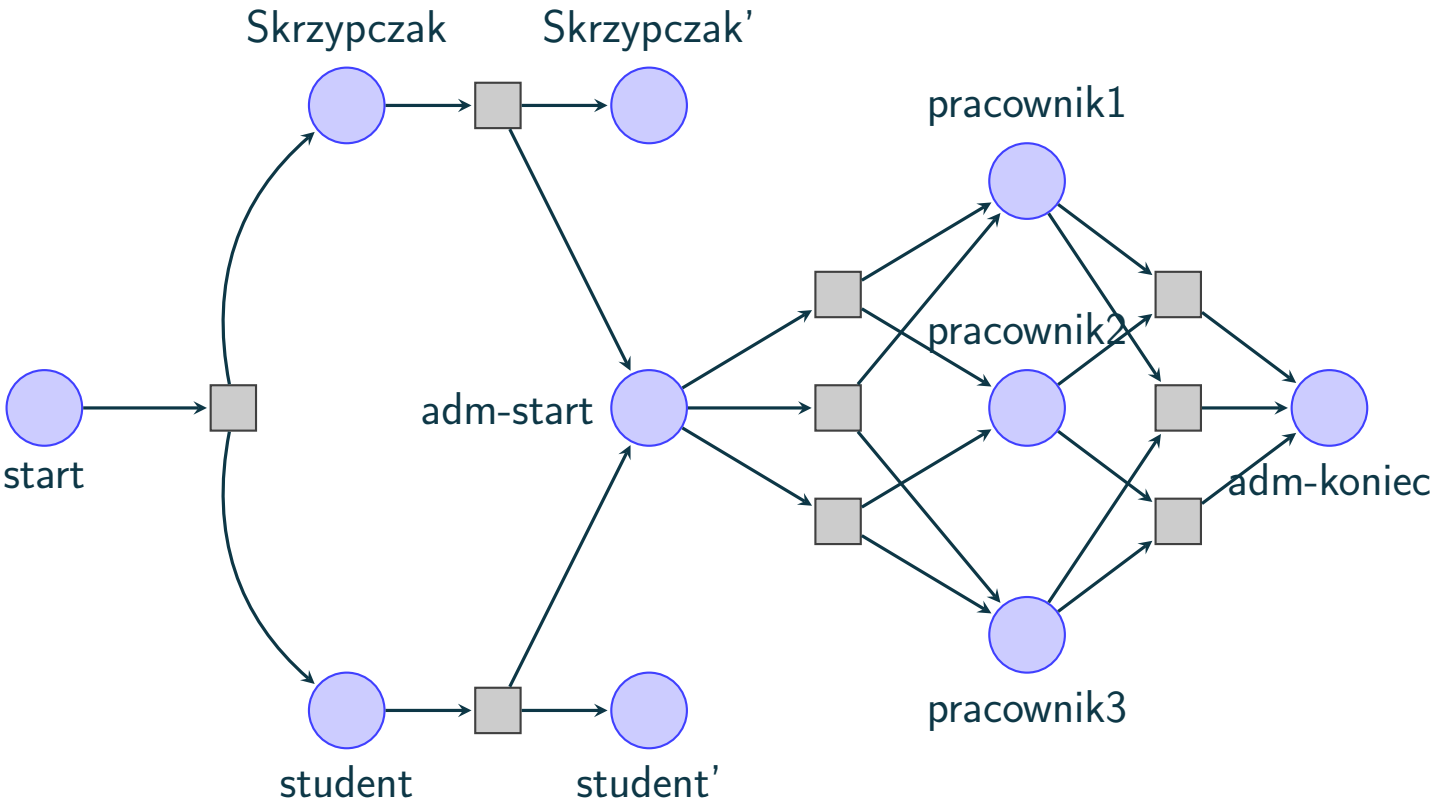
Połączmy te przykłady



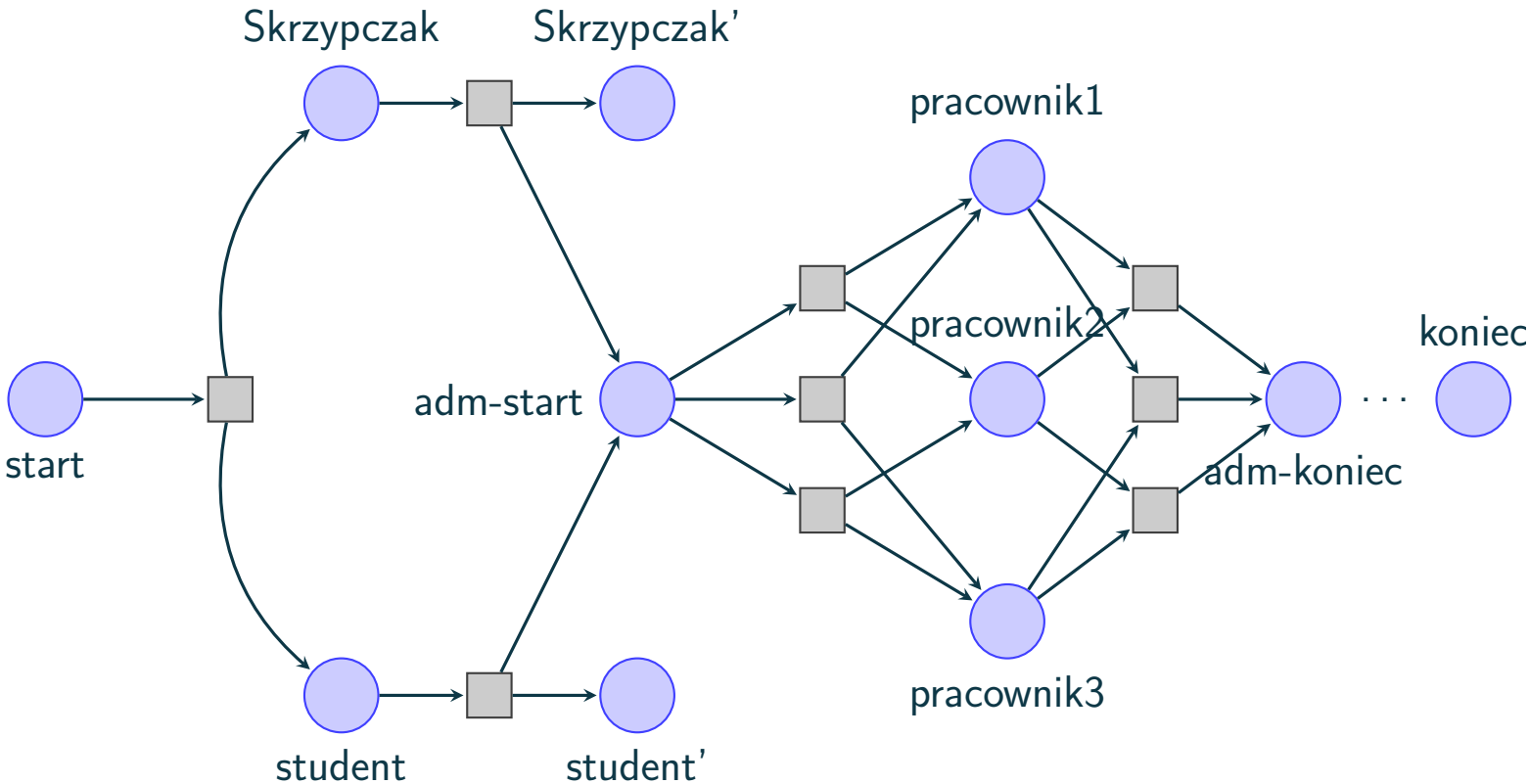
Połączmy te przykłady



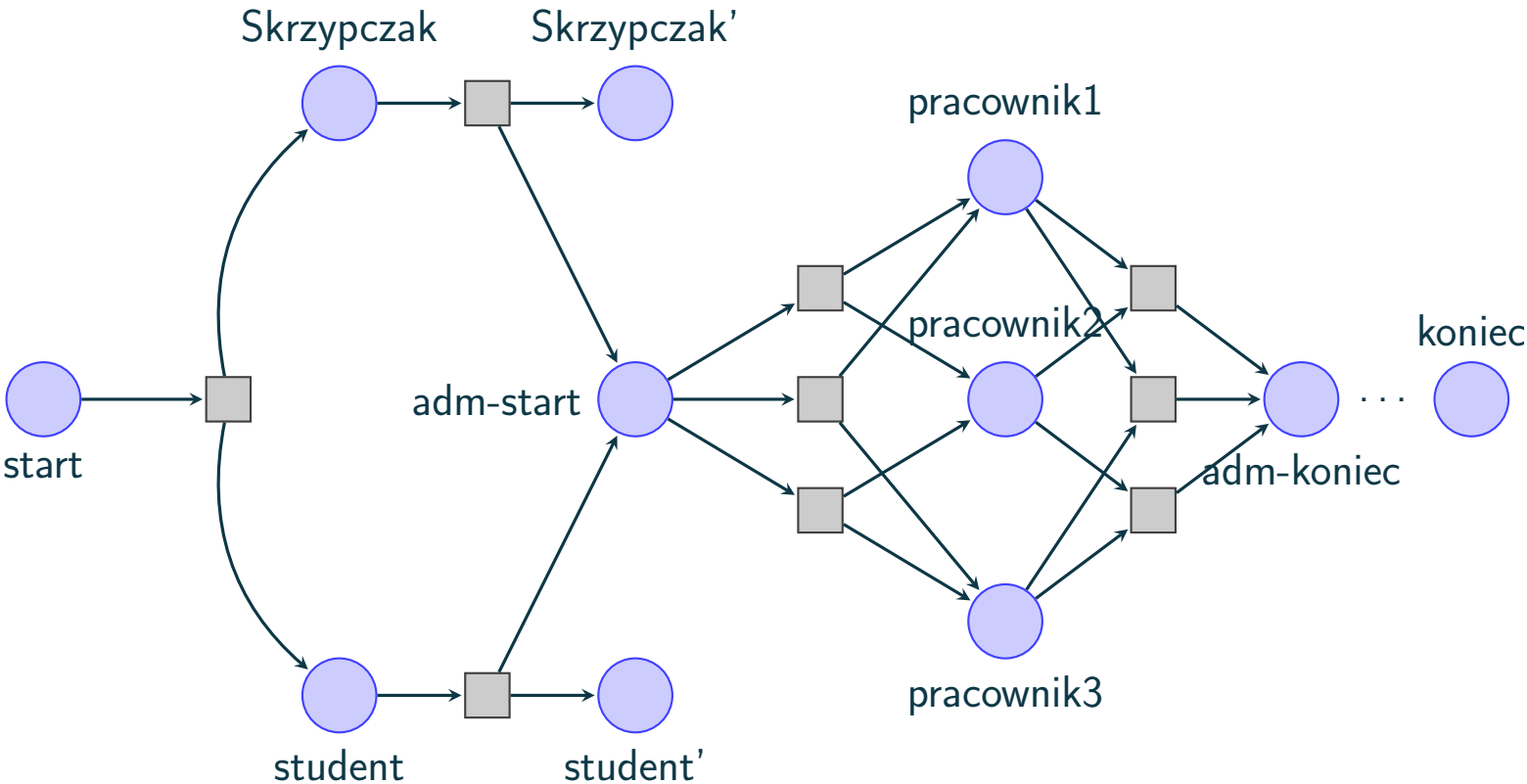
Połączmy te przykłady



Połączmy te przykłady

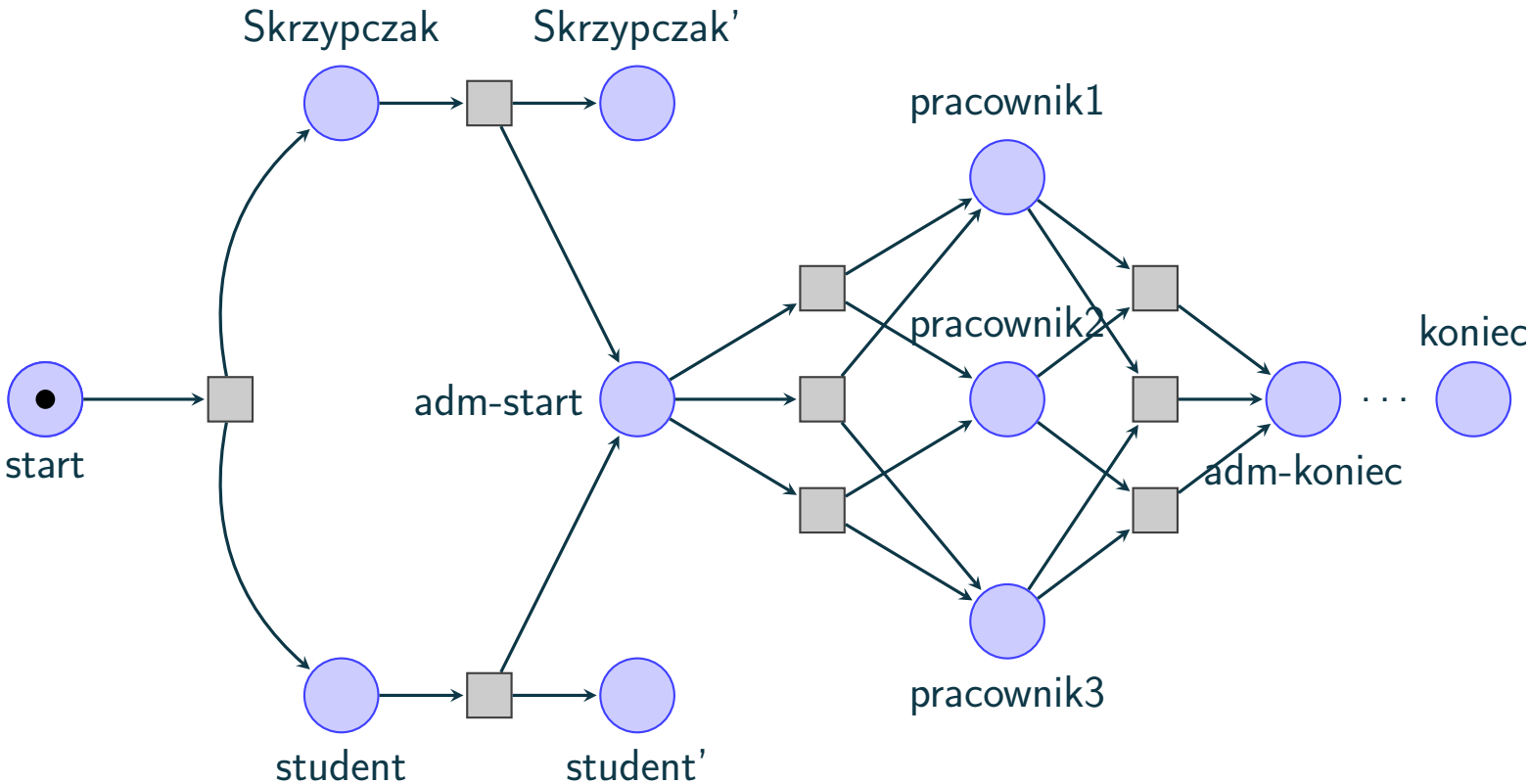


Połączmy te przykłady



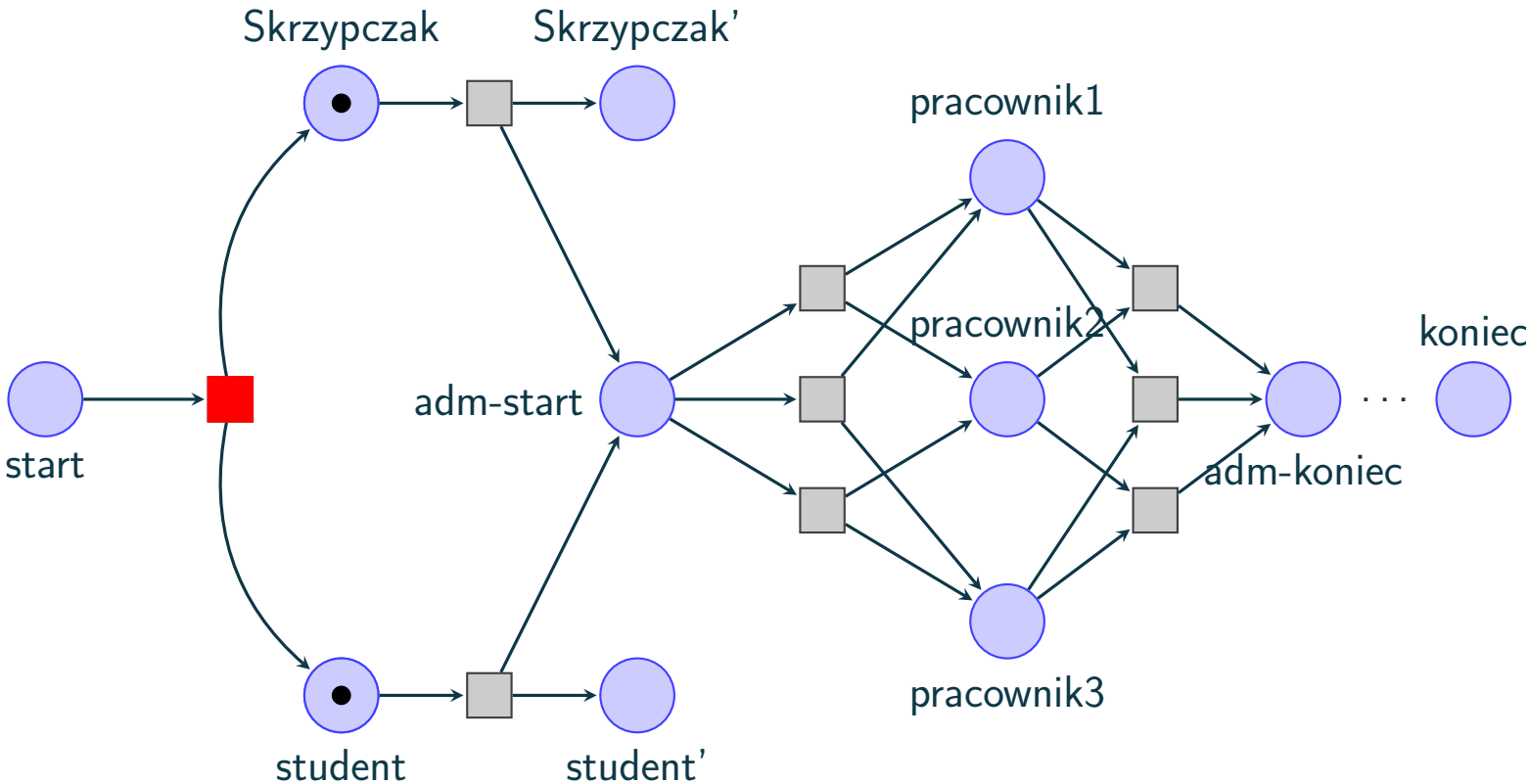
- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



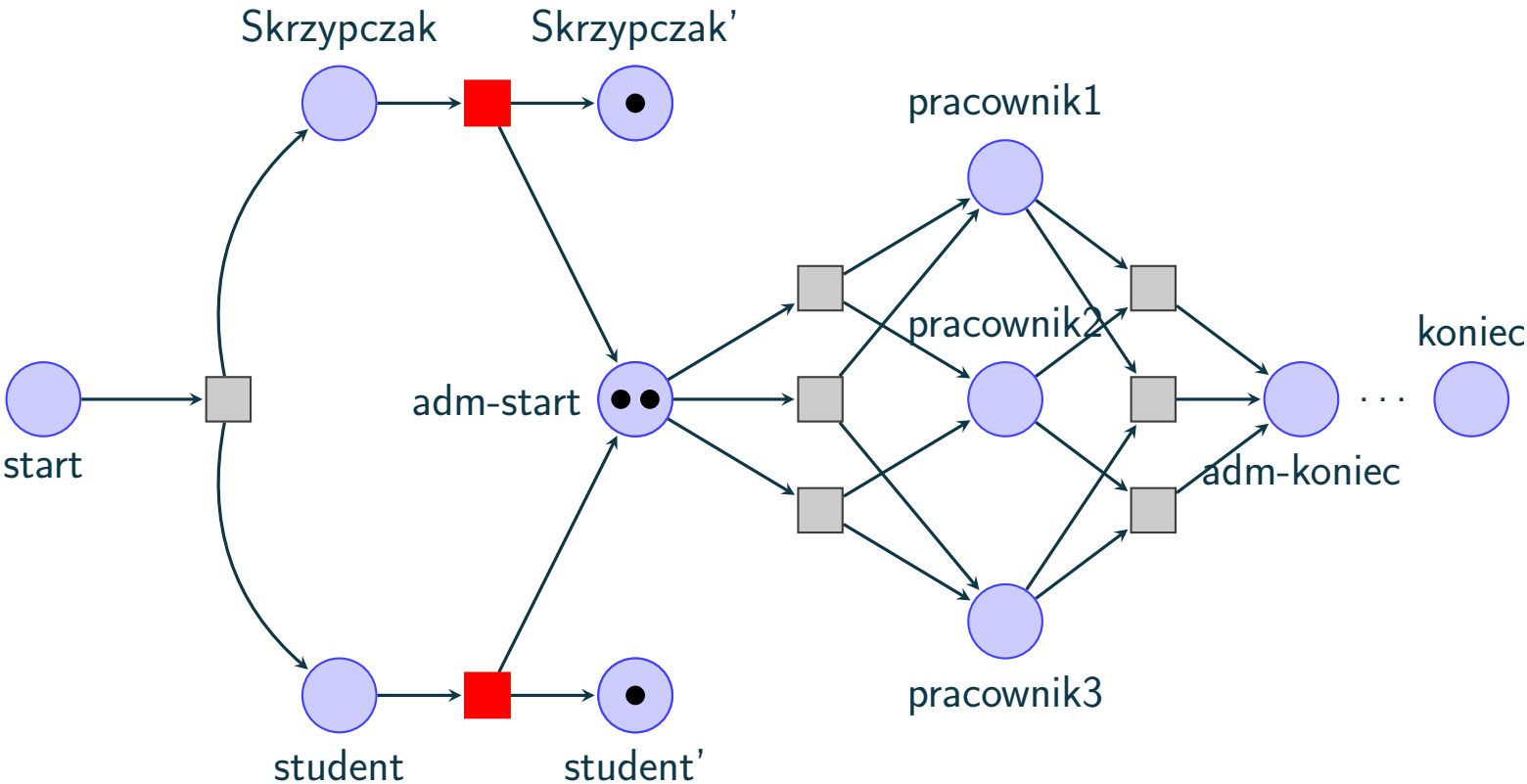
- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



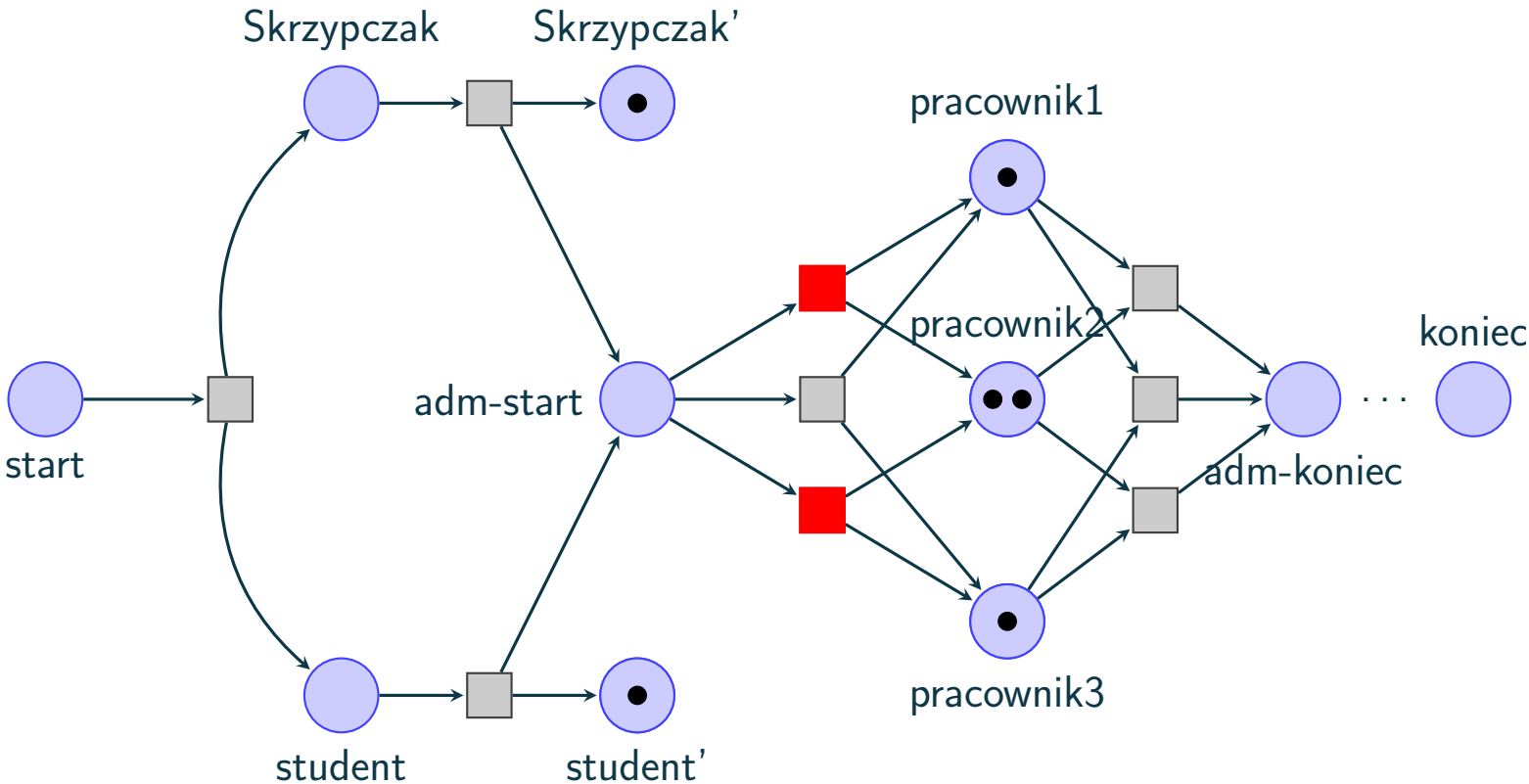
- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



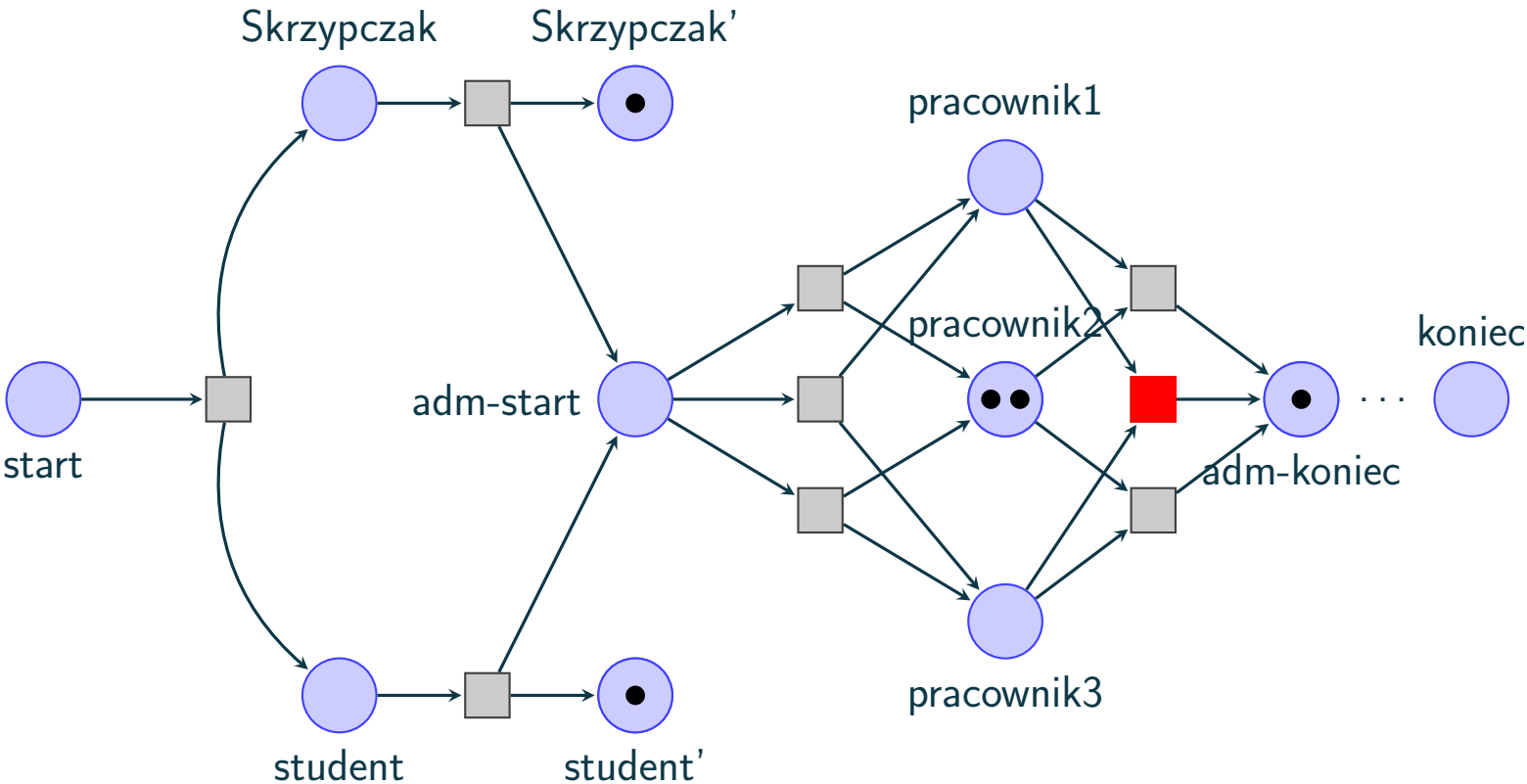
- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



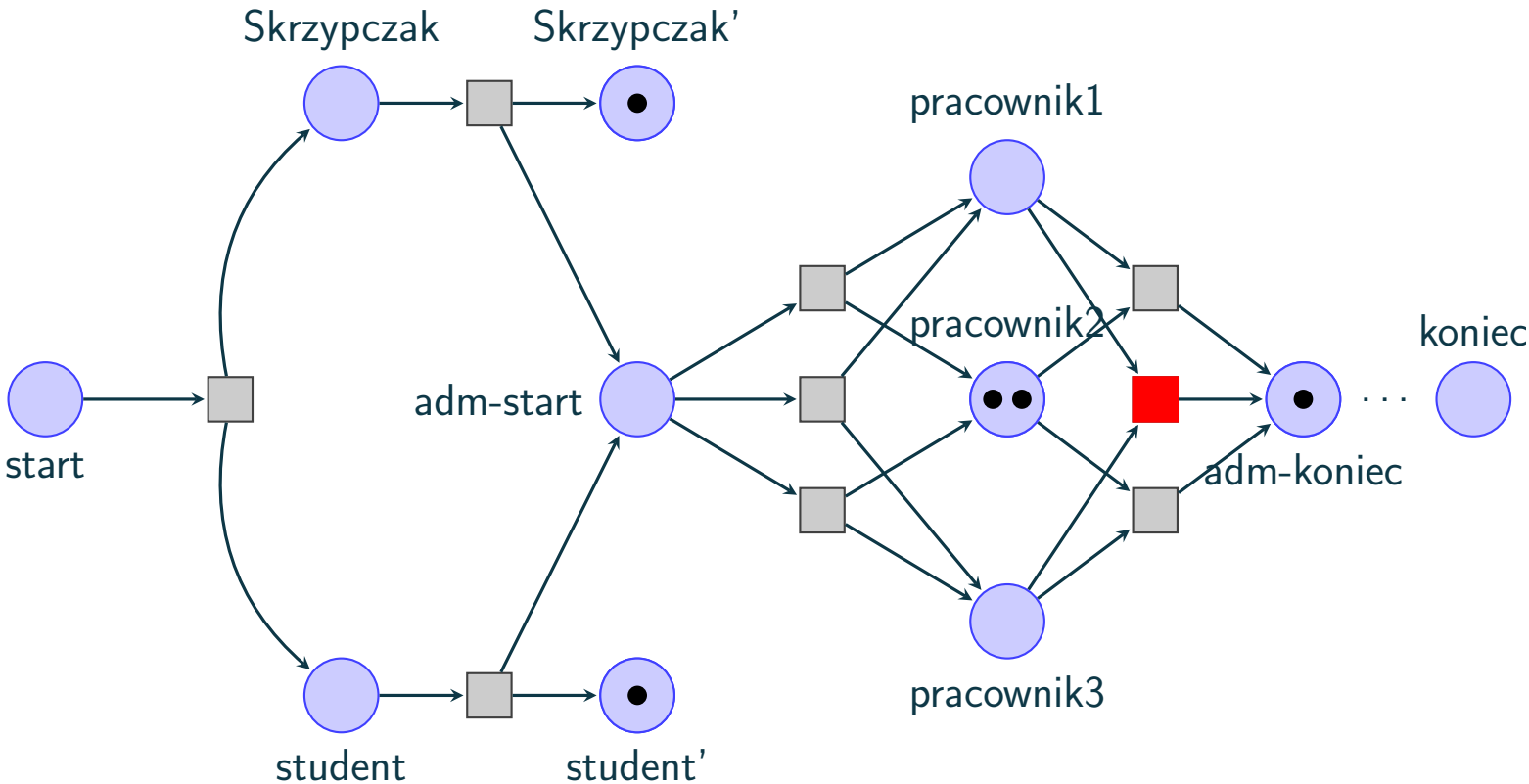
- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



- Przypomnijmy: obie części były poprawne

Połączmy te przykłady



- Przypomnijmy: obie części były poprawne
- A teraz to nie jest poprawne

Plan

1. Wstęp i przykłady

2. Coś o zastosowaniach

3. Jakiś dowód

Czy są algorytmy które rozwiązują te problemy?

Tak!

Czy są algorytmy które rozwiązują te problemy?

Tak!

Czy są to szybkie algorytmy?

Czy są algorytmy które rozwiązują te problemy?

Tak!

Czy są to szybkie algorytmy?

Nie!

Czy są algorytmy które rozwiązują te problemy?

Tak!

Czy są to szybkie algorytmy?

Nie!

O intuicji dlaczego nie jest ostatnia część

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

W postaci zwartej $x_n = 2^n$

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

W postaci zwartej $x_n = 2^n$

Jak napisać krótki (wielomianowy od n) program który oblicza x_n i

- może tylko dodawać stałe (np. x $+=$ 1)
- ma dostęp do n

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

W postaci zwartej $x_n = 2^n$

Jak napisać krótki (wielomianowy od n) program który oblicza x_n i

- może tylko dodawać stałe (np. x $+=$ 1)
- ma dostęp do n

loop
    x_n $-=$ 1 x_{n+1} $+=$ 2
until $x_n = 0$

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

W postaci zwartej $x_n = 2^n$

Jak napisać krótki (wielomianowy od n) program który oblicza x_n i

- może tylko dodawać stałe (np. x $+=$ 1)
- ma dostęp do n

loop
    x_n $-=$ 1 x_{n+1} $+=$ 2
until $x_n = 0$

Czyli potrzeba n zmiennych i $3n$ linijek kodu

Ciągi i proste języki programowania

Rozważmy $x_0 = 1$, $x_{n+1} = 2 \cdot x_n$

W postaci zwartej $x_n = 2^n$

Jak napisać krótki (wielomianowy od n) program który oblicza x_n i

- może tylko dodawać stałe (np. x $+=$ 1)
- ma dostęp do n

loop
$\quad x_n \mathrel{-}= 1 \quad x_{n+1} \mathrel{+}= 2$
until $x_n = 0$

Czyli potrzeba n zmiennych i $3n$ linijek kodu

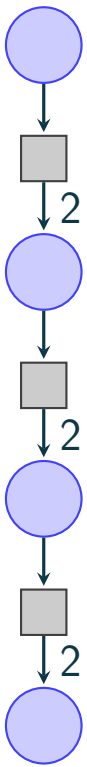
O sieciach Petriego można myśleć jako języku programowania bez **until**

Sieci Petriego jako język programowania

Na tym slajdzie $n = 3$

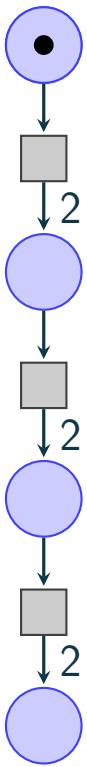
Sieci Petriego jako język programowania

Na tym slajdzie $n = 3$



Sieci Petriego jako język programowania

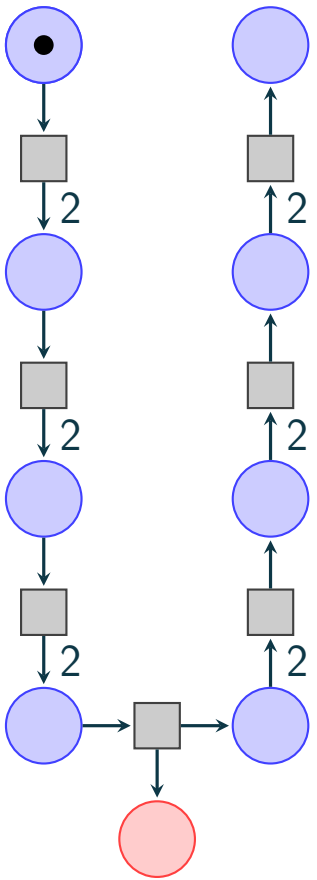
Na tym slajdzie $n = 3$



co najwyżej 2^n na dole

Sieci Petriego jako język programowania

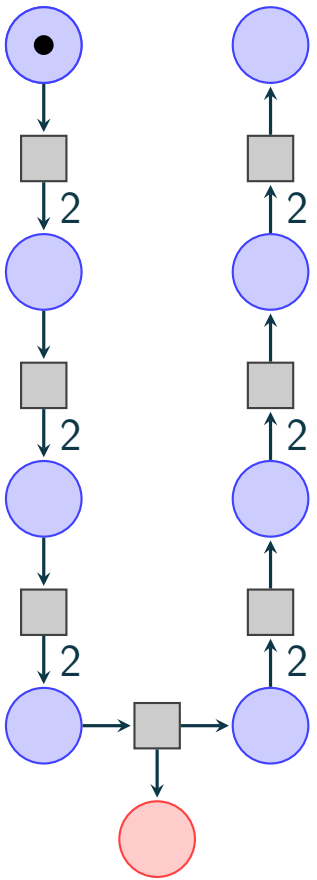
Na tym slajdzie $n = 3$



co najwyżej 2^n na dole

Sieci Petriego jako język programowania

Na tym slajdzie $n = 3$



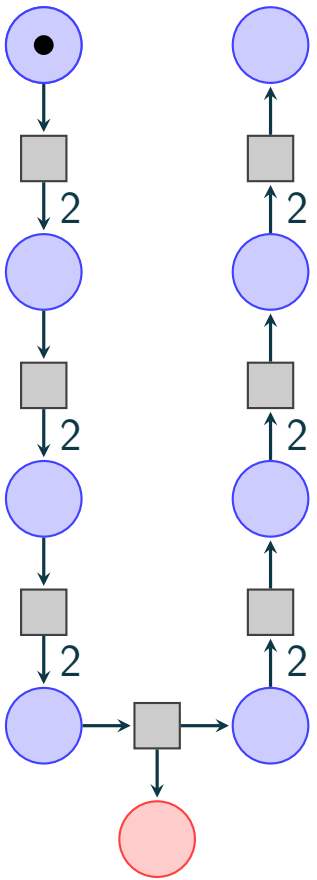
dojdzie do góry tylko jeśli 2^n na dole

(a w czerwonym zostanie 2^n)

co najwyżej 2^n na dole

Sieci Petriego jako język programowania

Na tym slajdzie $n = 3$



dojdzie do góry tylko jeśli 2^n na dole

(a w czerwonym zostanie 2^n)

$x_0 \mathrel{+}= 1$

loop

$x_0 \mathrel{-}= 1 \quad x_2 \mathrel{+}= 2$

$\vdots$

loop

$x_{n-1} \mathrel{-}= 1 \quad x_n \mathrel{+}= 2$

loop

$x_n \mathrel{-}= 1 \quad y_n \mathrel{+}= 1 \quad z \mathrel{+}= 1$

loop

$y_n \mathrel{-}= 2 \quad y_{n-1} \mathrel{+}= 1$

$\vdots$

loop

$y_1 \mathrel{-}= 2 \quad y_0 \mathrel{+}= 1$

co najwyżej 2^n na dole

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Ale mógł dojść do końca tylko jeśli w czerwonym miejscu było dokładnie 2^n.

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Ale mógł dojść do końca tylko jeśli w czerwonym miejscu było dokładnie 2^n.

$\approx$ każdy algorytm na pokrywalność będzie wolny (co najmniej 2^n kroków).

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Ale mógł dojść do końca tylko jeśli w czerwonym miejscu było dokładnie 2^n.

$\approx$ każdy algorytm na pokrywalność będzie wolny (co najmniej 2^n kroków).

Jakie inne ciągi można "zaimplementować"?

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Ale mógł dojść do końca tylko jeśli w czerwonym miejscu było dokładnie 2^n.

$\approx$ każdy algorytm na pokrywalność będzie wolny (co najmniej 2^n kroków).

Jakie inne ciągi można "zaimplementować"?

Twierdzenie (Lipton 1976)

$x_0 = 1$, $x_{n+1} = x_n^2$, czyli $x_n = 2^{2^n}$

Sieci Petriego jako język programowania (2)

Poprzedni program mógł się skończyć w wielu miejscach

Ale mógł dojść do końca tylko jeśli w czerwonym miejscu było dokładnie 2^n.

$\approx$ każdy algorytm na pokrywalność będzie wolny (co najmniej 2^n kroków).

Jakie inne ciągi można "zaimplementować"?

Twierdzenie (Lipton 1976)

$x_0 = 1$, $x_{n+1} = x_n^2$, czyli $x_n = 2^{2^n}$

Kluczowe jest symulowanie $x_{n+1} = x_n^2$

(tak jak poprzednio $x_{n+1} = 2 \cdot x_n$)

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Dla $i = 0$: zainicjalizować $x_0 = 0$ i $x'_0 = 1$ $\quad$ jak $x_0 \mathrel{+}= 1$ to $x_0 \mathrel{-}= 1$ itp

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Dla $i = 0$: zainicjalizować $x_0 = 0$ i $x'_0 = 1$ $\quad$ jak $x_0 \mathrel{+}= 1$ to $x_0 \mathrel{-}= 1$ itp

Dec$_0$ x_0 to $\quad x_0 \mathrel{-}= 2 \quad x'_0 \mathrel{+}= 2$ i podobnie **Inc**$_0$ x_0

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Dla $i = 0$: zainicjalizować $x_0 = 0$ i $x'_0 = 1$ $\quad$ jak $x_0 \mathrel{+}= 1$ to $x_0 \mathrel{-}= 1$ itp

Dec$_0$ x_0 to $\quad x_0 \mathrel{-}= 2 \quad x'_0 \mathrel{+}= 2$ i podobnie **Inc**$_0$ x_0

Krok indukcyjny

- Inicjalizowanie

loop
  $x_n \mathrel{+}= 1 \quad x_n \mathrel{-}= 1$
  loop
    $y_n \mathrel{+}= 1 \quad y'_n \mathrel{+}= 1$
    $x'_{n+1} \mathrel{+}= 1 \quad y'_{n+1} \mathrel{+}= 1$
  Dec$_n$ y_n
Dec$_n$ x_n

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Dla $i = 0$: zainicjalizować $x_0 = 0$ i $x'_0 = 1$ $\quad$ jak $x_0 \mathrel{+}= 1$ to $x_0 \mathrel{-}= 1$ itp

Dec$_0$ x_0 to $\quad x_0 \mathrel{-}= 2 \quad x'_0 \mathrel{+}= 2$ i podobnie **Inc**$_0$ x_0

Krok indukcyjny

- Inicjalizowanie
- dla **Dec**$_{n+1}$ x_{n+1}

loop
  $x_n \mathrel{+}= 1 \quad x_n \mathrel{-}= 1$
  loop
    $y_n \mathrel{+}= 1 \quad y'_n \mathrel{+}= 1$
    $x_{n+1} \mathrel{-}= 1 \quad x'_{n+1} \mathrel{+}= 1$
  Dec$_n$ y_n
Dec$_n$ x_n

Symulowanie $x_{n+1} = x_n^2$

Pomysł, dla każdego $0 \leqslant i \leqslant n$ indukcyjnie mieć:

- zmienne x_i, y_i oraz x'_i, y'_i t. że $x_i + x'_i = y_i + y'_i = 2^{2^i}$ (domyślnie $x_i = 0$)

- **Dec**$_i$ x_i równoważne $x_i \mathrel{-}= 2^{2^i}$ $\quad x'_i \mathrel{+}= 2^{2^i}$

  Inc$_i$ x_i równoważne $x_i \mathrel{+}= 2^{2^i}$ $\quad x'_i \mathrel{-}= 2^{2^i}$

Dla $i = 0$: zainicjalizować $x_0 = 0$ i $x'_0 = 1$ $\quad$ jak $x_0 \mathrel{+}= 1$ to $x_0 \mathrel{-}= 1$ itp

Dec$_0$ x_0 to $\quad x_0 \mathrel{-}= 2 \quad x'_0 \mathrel{+}= 2$ i podobnie **Inc**$_0$ x_0

Krok indukcyjny

- Inicjalizowanie
- dla **Dec**$_{n+1}$ x_{n+1}
- dla **Inc**$_{n+1}$ x_{n+1}

loop
  $x_n \mathrel{+}= 1 \quad x_n \mathrel{-}= 1$
  loop
    $y_n \mathrel{+}= 1 \quad y'_n \mathrel{+}= 1$
    $x_{n+1} \mathrel{+}= 1 \quad x'_{n+1} \mathrel{-}= 1$
  Dec$_n$ y_n
Dec$_n$ x_n

Konkluzja

- Nie da się trudniej niż podwójnie wykładnicze

  dla pokrywalności [Rackoff 1978] i poprawności [Blondin et al. 2022]

Konkluzja

- Nie da się trudniej niż podwójnie wykładnicze

  dla pokrywalności [Rackoff 1978] i poprawności [Blondin et al. 2022]

- Dla osiągalności można trudniej

  $x_0 = 3$, $x_{n+1} = x_n!$ [Czerwiński et al. 2019]

  $x_n \approx A_n$ (funkcja Ackermanna) [Czerwiński i Orlikowski 2021], [Leroux 2021]

Konkluzja

- Nie da się trudniej niż podwójnie wykładnicze

  dla pokrywalności [Rackoff 1978] i poprawności [Blondin et al. 2022]

- Dla osiągalności można trudniej

  $x_0 = 3$, $x_{n+1} = x_n!$ [Czerwiński et al. 2019]

  $x_n \approx A_n$ (funkcja Ackermanna) [Czerwiński i Orlikowski 2021], [Leroux 2021]

- Mimo to istnieją sensowne implementacje tych problemów

Konkluzja

- Nie da się trudniej niż podwójnie wykładnicze

  dla pokrywalności [Rackoff 1978] i poprawności [Blondin et al. 2022]

- Dla osiągalności można trudniej

  $x_0 = 3$, $x_{n+1} = x_n!$ [Czerwiński et al. 2019]

  $x_n \approx A_n$ (funkcja Ackermanna) [Czerwiński i Orlikowski 2021], [Leroux 2021]

- Mimo to istnieją sensowne implementacje tych problemów

  Intuicyjnie można uprościć problem

  np. pozwolić na tranzycje które powodują że liczba żetonów jest ujemna

Konkluzja

- Nie da się trudniej niż podwójnie wykładnicze

  dla pokrywalności [Rackoff 1978] i poprawności [Blondin et al. 2022]

- Dla osiągalności można trudniej

  $x_0 = 3$, $x_{n+1} = x_n!$ [Czerwiński et al. 2019]

  $x_n \approx A_n$ (funkcja Ackermanna) [Czerwiński i Orlikowski 2021], [Leroux 2021]

- Mimo to istnieją sensowne implementacje tych problemów

  Intuicyjnie można uprościć problem

  np. pozwolić na tranzycje które powodują że liczba żetonów jest ujemna

  Obserwacja: jeśli dla uproszczonej pokrywalności otrzymamy odpowiedź NIE

  to znaczy że dla zwykłej pokrywalności też jest NIE