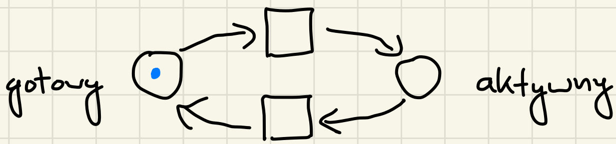
Ćwiczenia 1 2.10

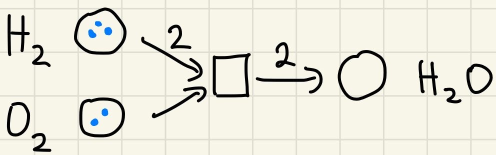
Sieci Petriego – krótkie wprowadzenie

* przykład: stan procesu



* przykład: reakcje chemiczne

$$2H_2 + O_2 \to 2H_2O$$



Elementarna sieć Petriego z konfiguracją:

* zbiór **miejsc** P i **tranzycji** T, $P \cap T = \emptyset$
* zbiór **łuków** $F \subseteq (P \times T) \cup (T \times P)$
* konfiguracja $M \subseteq P$

Ogólna sieć:

* F to łuki z przypisanymi wagami ($\in \mathbb{N}$)
* $M : P \to \mathbb{N}$ jako uogólnienie konfiguracji

Kiedy można odpalić tranzycję?

* w elementarnych sieciach, gdy mamy po żetonie na miejscach wejściowych (ozn. $^\bullet t$), a miejsca wyjściowe ($t^\bullet$) są puste
* w ogólnych sieciach, gdy na miejscach wejściowych liczby żetonów są co najmniej takie jak wagi łuków

od teraz mamy na myśli sieci ogólne

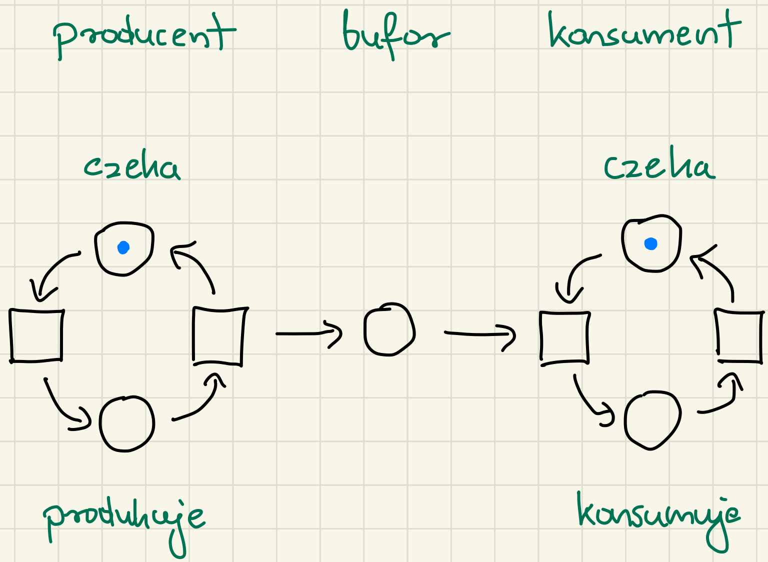
Ważne własności (mówimy o sieci z konfiguracją)

* osiągalność – czy z konfiguracji początkowej można na skutek odpalenia pewnej sekwencji tranzycji przejść do innej?
* żywotność – każda tranzycja jest żywa; tranzycja t jest żywa, gdy z każdej osiągalnej konfiguracji można odpalić pewną sekwencję tranzycji, po których możliwe jest odpalenie tranzycji t

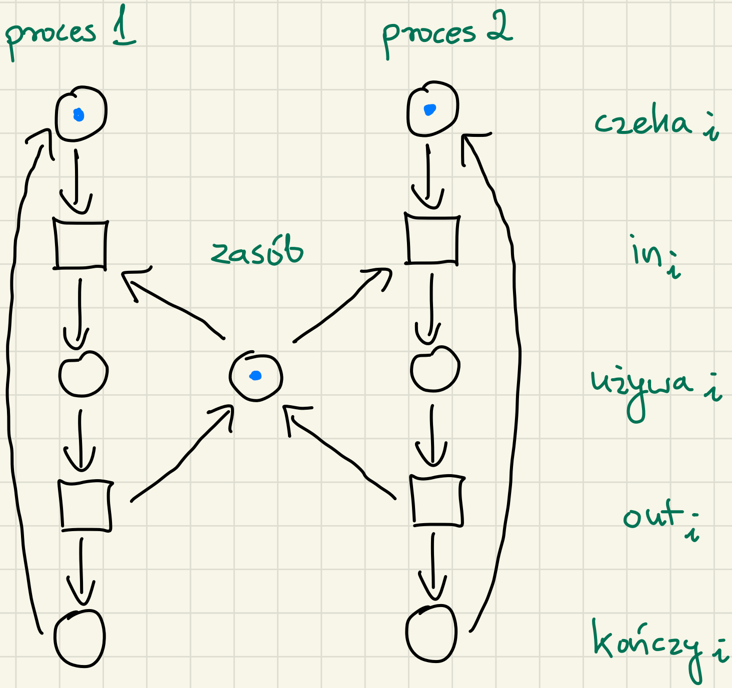
* **K-ograniczoność** – każde miejsce jest K-ograniczone; miejsce jest k-ograniczone, jeśli w każdej osiągalnej konfiguracji zawiera nie więcej niż k żetonów
* **ograniczoność** – sieć z konfiguracją jest ograniczona, jeśli istnieje $K \in \mathbb{N}$, takie że mamy K-ograniczoność
* **brak blokady** – nie można osiągnąć konfiguracji w której wszystkie tranzycje są martwe (nie można ich odpalić)

Możemy też mówić o strukturalnych własnościach sieci, które nie zależą od wybranej konfiguracji. Przykładowo, sieć jest **strukturalnie ograniczona**, jeśli dla każdej możliwej konfiguracji początkowej M sieć z konfiguracją M jest ograniczona.

1. Zamodeluj za pomocą sieci elementarnych klasyczne problemy współbieżności: producenta / konsumenta i wzajemne wykluczanie



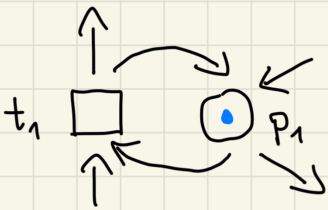
* producent i konsument nie komunikują się bezpośrednio
* jedyna komunikacja zachodzi przez bufor



* procesy 1 i 2 nigdy nie komunikują się ze sobą bezpośrednio, tzn. nie wykonują wspólnych akcji (tranzycji)
* Łatwo uogólnić na więcej procesów
* nie pilnujemy tutaj, aby każdy proces, który chce skorzystać z zasobu, mógł tego w końcu dokonać (problem zagłodzenia)

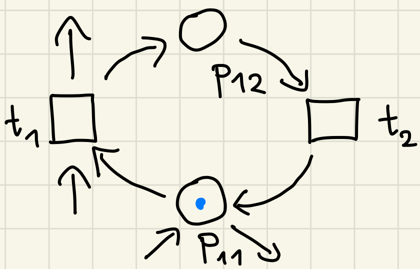
2. Eliminacja ciasnych pętli, eliminacja wag > 1

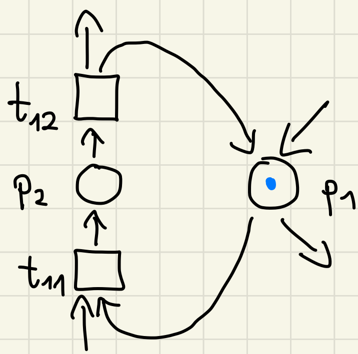
ciasna pętla = miejsce i tranzycja są połączone w obie strony



I sposób — powielenie miejsca

II sposób — powielenie tranzycji



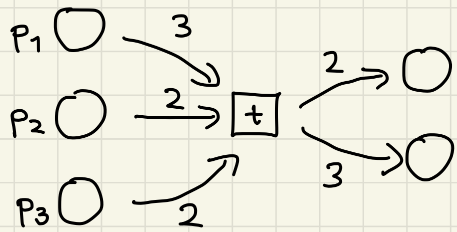


po odpaleniu t_1 tranzycja t_2 może w dowolnym momencie przenieść żeton z p_{12} na p_{11}

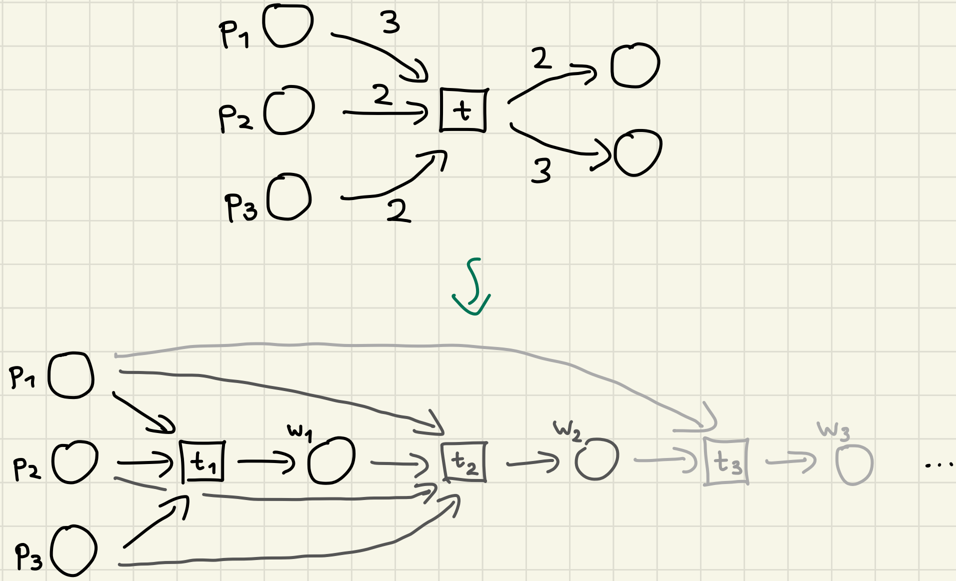
przy tej modyfikacji żeton na miejscu p_2 oznacza coś w stylu „wykonuje się tranzycja t_1"

* założenie $^\bullet t \cap t^\bullet = \emptyset$ upraszcza analizę własności sieci

Eliminacja wag > 1



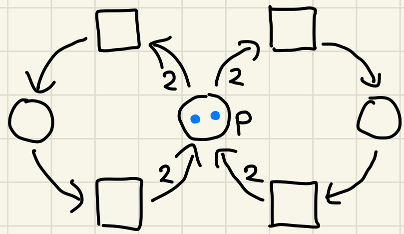
* skoro mamy dostępne tylko łuki o wagach 1, pomysł jest taki, aby stopniowo pobierać po jednym żetonie z każdego miejsca
* w przykładzie tranzycja t pobiera 3 żetony z p_1 oraz po 2 żetony z p_2 i p_3, dlatego „rozbijemy" „część pobierającą" żetony na 3 nowe tranzycje: t_1, t_2, t_3
* pierwsza z nich, t_1, pobierze po 1 żetonie z każdego miejsca, druga zrobi to samo, a trzecia zabierze już tylko jeden żeton z p_1
* stąd łącznie t_1, t_2 i t_3 zabiorą 3 żetony z p_1 i po 2 żetony z p_2, p_3, czyli dokładnie tyle, co tranzycja t



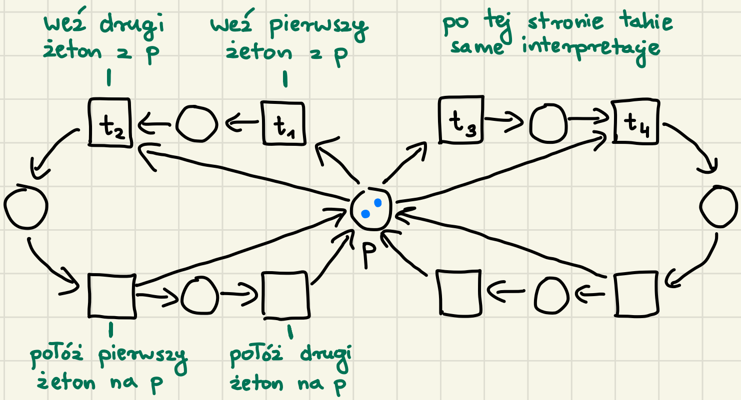
* w powyższej konstrukcji dodaliśmy też miejsca wyjściowe w_1, w_2, w_3 odpowiednio dla t_1, t_2, t_3, które oznaczają: wzięto po 1, 2, 3 żetony z miejsc, z których musieliśmy tyle wziąć

* problem: w oryginalnej sieci może być niemożliwe osiągnięcie dead-locka, a po tej transformacji sieć może się zakleszczać

oryginalna sieć:

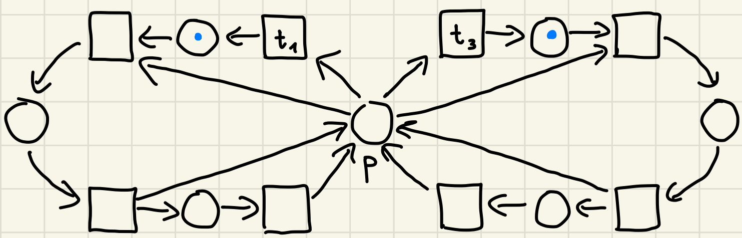


sieć po transformacji :



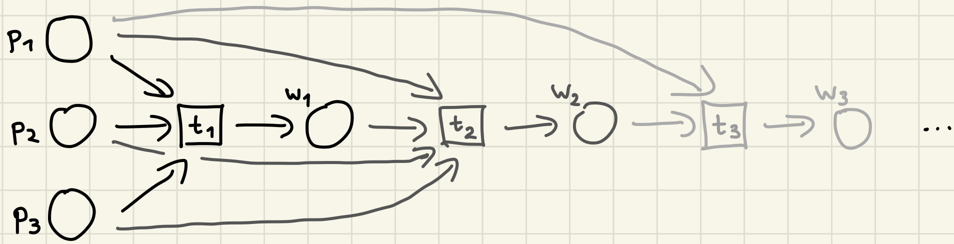
w oryginalnej sieci nie było zakleszczenia,

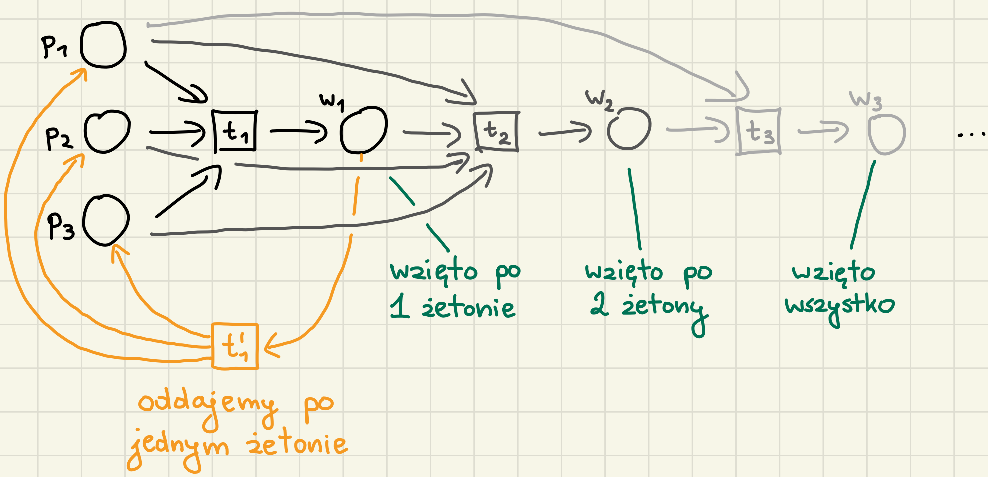
w nowej po odpaleniu t_1 i t_3 mamy dead-lock



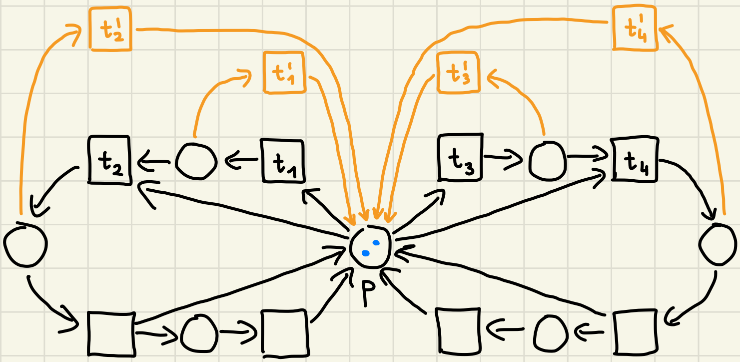
jak to naprawić ?

* wystarczy dodać tranzycje, które będą odkładać żetony z powrotem na początkowe miejsce
* w pierwszym przykładzie dla tranzycji t_1 tworzymy tranzycję t_1', która pobiera żeton z miejsca w_1 (oznaczającego „pobrano po jednym żetonie z p_1, p_2 i p_3") i kładzie po jednym żetonie na p_1, p_2 i p_3





* w drugim przykładzie cała nowa sieć będzie wyglądać tak:



* w takiej sieci nie wystąpi już zakleszczenie
* Łatwo też dla niej zdefiniować, które jej konfiguracje odpowiadają którym w oryginalnej sieci

* uwaga: jeśli zmienimy początkową konfigurację na taką z jednym żetonem na miejscu p, to oryginalna sieć będzie od razu w dead-locku, a ta powyżej nie (może odpalać t_1 i t_1' na zmianę)

* wniosek: trzeba uważać, jakie własności sieci się zachowują przy jej zmienianiu

3. Równoważność sieci elementarnych oraz 1-ograniczonych sieci ogólnych

$\Leftarrow$

* możemy założyć, że nie ma wag > 1, wpp tranzycje z takimi łukami można usunąć
* otrzymana sieć jest siecią elementarną, bo początkowa konfiguracja kładzie na pola pojedyncze żetony, a w dowolnej osiągalnej konfiguracji M każda żywa tranzycja t spełnia $t^{\bullet} \cap M = \phi$

$\Rightarrow$

Uwaga: nie możemy po prostu użyć danej sieci elementarnej i powiedzieć, że traktujemy ją jako ogólną sieć – np. sieć producenta-konsumenta staje się wtedy nieograniczona (na buforze może się wygenerować dowolna liczba żetonów)

Podpowiedź: w sieci elementarnej każde miejsce albo zawiera żeton albo jest puste, czyli tak jakby ma dwa stany