Równoważność systemów współbieżnych – bisymulacja

Marcin Poturalski
m.poturalski@students.mimuw.edu.pl

30 listopada 2004

KONTEKST

Weryfikacja systemów współbieżnych.

- SWP: „weryfikacja nie ma sensu . . . za wyjątkiem małych ważnych części systemu" – takich jak protokoły komunikacyjne.

- Dzisiaj istnieje szereg narzędzi komputerowych wspomagających weryfikację: CWB, SPIN, UPPAAL.

- Szczególnie systemy współbieżne warto weryfikować wspomagając się komputerem, a to dlatego, że są one bardzo "nieludzkie".

W ramach tego referatu opowiem trochę o teorii, która za takimi narzędziami stoi. A tak naprawdę o teorii za tą teorią.

KONTEKST

Jak to sie robi:

1. Modelowanie

2. Właściwa weryfikacja

   - **model checking** - sprawdzanie, czy model spełnia formułę w jakiejś logice (logika modalna, rachunek μ); pozwala sprawdzać takie własności jak bezpieczeństwo, czy żywotność
   - **equivalence checking** - sprawdzanie, czy dwa systemy są w jakimś sensie równoważne; o różnych rodzajach równoważności (ze szczególnym naciskiem na bisymulację) traktuje ten referat

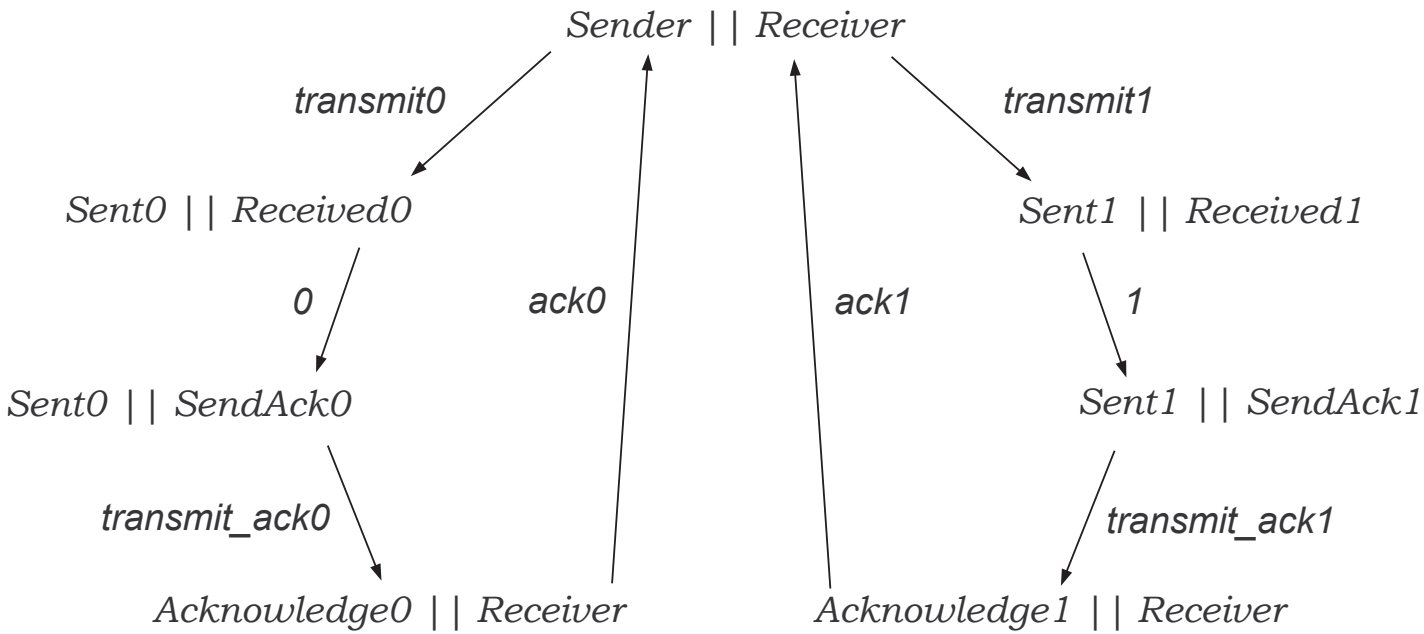
Przykład 1 (prawie CCS)

Prosty protokół:

- **Nadawca** nadaje bit 0 lub 1
- **Odbiorca** potwierdza otrzymanie tego bitu

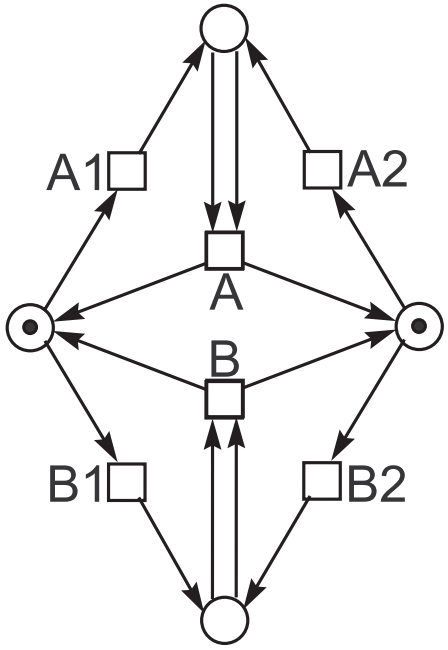
$$
\begin{aligned}
Sender &= transmit0.Sent0 + transmit1.Sent1 \\
SentB &= \overline{transmit\_ackB}.AcknowledgeB \\
AcknowledgeB &= \mathsf{ackB}.Sender \\
\\
Receiver &= \overline{transmit0}.Received0 + \overline{transmit1}.Received1 \\
ReceivedB &= \mathsf{B}.SendAckB \\
SendAckB &= transmit\_ackB.Receiver \\
\\
System &= Sender||Receiver
\end{aligned}
$$

Graf stanów osiągalnych:

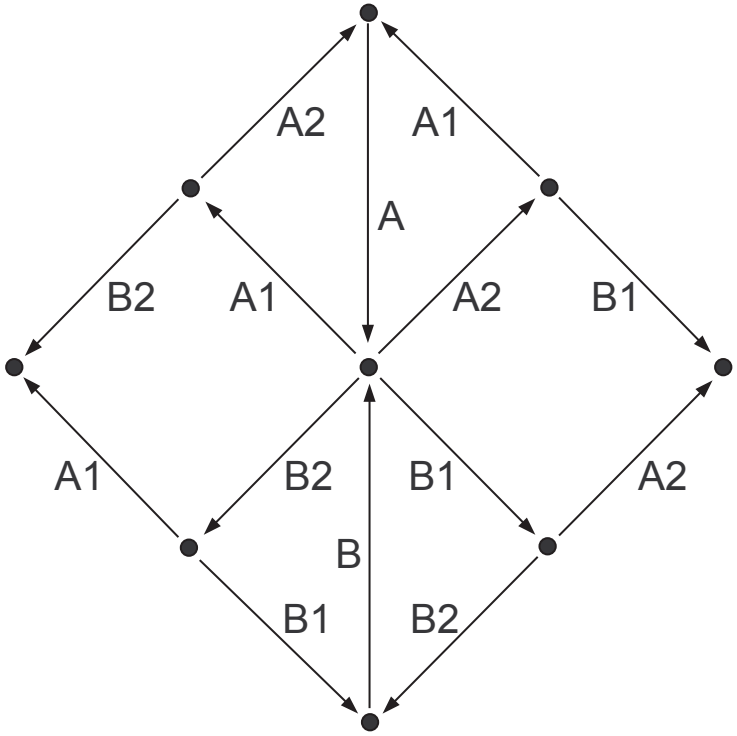


Przykład 2 (sieć Petriego)

Problem dwóch filozofów



Graf stanów osiągalnych:



RÓWNOŚĆ JĘZYKÓW GENEROWANYCH

Na graf osiągalnych stanów można spojrzeć jak na znany z JAiO automat:

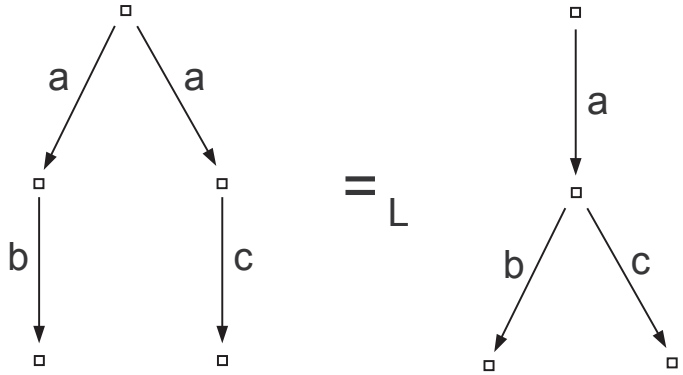
- wierzchołki to stany
- etykietowane krawędzie wyznaczają relację przejścia
- stan początkowy jest wyróżniony
- za stany końcowe możemy przyjąć wszystkie stany

RÓWNOŚĆ JĘZYKÓW GENEROWANYCH

Z automatem związane jest pojęcie rozpoznawanego (generowanego) przez niego języka.

Prowadzi nas to do pierwszej równoważności procesów. Procesy P i Q są równoważne, jeśli odpowiadające im automaty generują (rozpoznają) ten sam język:

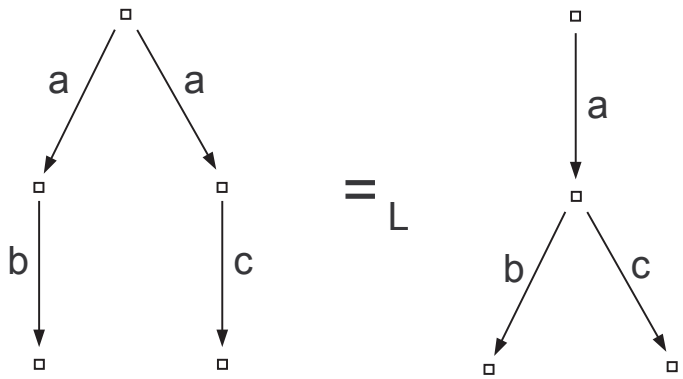
$$P =_L Q \quad \equiv_{def} \quad L(P) = L(Q)$$



RÓWNOŚĆ JĘZYKÓW GENEROWANYCH

Równoważność ta, zwana **trace equivalence** dobrze oddaje ideę równoważności obserwacyjnych, którymi się zajmiemy – porównujemy jedynie zdarzenia, jakie procesy mogą wykonać, niejako obserwujemy ich działanie. Nie wchodzimy w strukturę wewnętrzną procesów.

W kontekście procesów współbieżnych równoważność ta nie jest jednak bardzo użyteczna. Utożsamia ona zbyt wiele procesów, ponieważ bierze pod uwagę tylko zbiór możliwych przebiegów.



BISYMULACJA

Podstawowa równoważność dla teorii procesów współbieżnych to **bisymulacja**.

Zdefiniujemy ją poprzez grę. Gra toczy się w turach. "Planszą" są dwa procesy **P** i **Q**.

Gracz I

TURA

- wybiera jeden z procesów
- w wybranym procesie wykonuje dowolną tranzycję – proces zmienia stan

KONIEC TURY

Gracz II

- w drugim procesie wykonuje tranzycję o etykiecie takiej, jak tranzycja wybrana przez **Gracza I** – proces zmienia stan

BISYMULACJA

Gracz I | **Gracz II**

TURA

- wybiera jeden z procesów
- w wybranym procesie wykonuje dowolną tranzycję – proces zmienia stan

- w drugim procesie wykonuje tranzycję o etykiecie takiej, jak tranzycja wybrana przez **Gracza I** – proces zmienia stan

KONIEC TURY

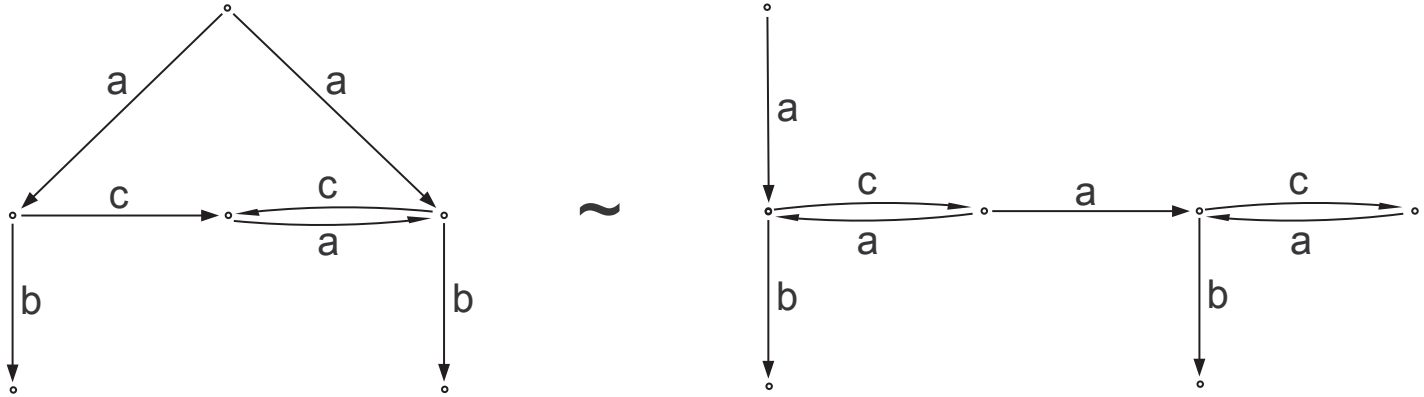
Gracz I wygrywa, gdy **Gracz II** nie może wykonać ruchu.

Gracz II wygrywa w przeciwnym przypadku, czyli gdy **Gracz I** nie może wykonać ruchu lub gdy gra toczy się w nieskończoność.

BISYMULACJA

Mówimy, że procesy P i Q są **bisymulacyjnie równoważne**, gdy **Gracz II** ma strategię wygrywającą. Oznaczamy to $P \sim Q$.

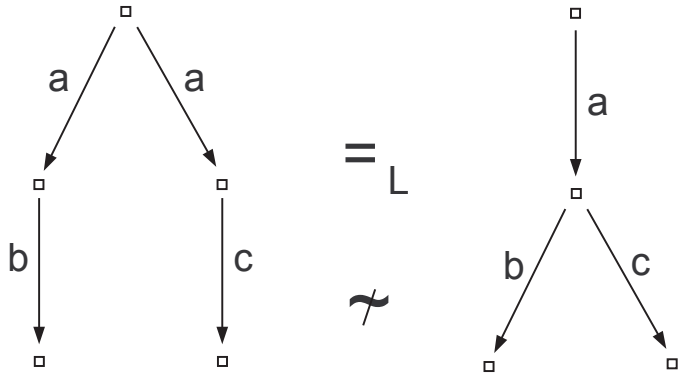
W przeciwnym przypadku, czyli gdy strategię wygrywającą ma **Gracz I**, procesy P i Q nie są w bisymulacji.



Bisymulacja jest oczywiście relacją równoważności.

BISYMULACJA a TRACE EQUIVALENCE

Można się zastanawiać, jak się ma **trace equivalence** (równość języków generowanych) do bisymulacji. Spójrzmy ponownie na przykład:



Jak widać procesy równoważne z punktu widzenia **trace equivalence** nie muszą być w bisymulacji.

BISYMULACJA a TRACE EQUIVALENCE

Implikacja w drugą stronę zachodzi: procesy, które są w bisymulacji, generują ten sam zestaw śladów:

$$P \sim Q \Rightarrow P =_L Q \quad (\sim \subseteq =_L)$$

Czy przy jakiś dodatkowych założeniach **trace equivalence** implikuje równoważność bisymulacyjną?

BISYMULACJA a TRACE EQUIVALENCE

Czy przy jakiś dodatkowych założeniach **trace equivalence** implikuje równoważność bisymulacyjną?

Tak. Dla procesów deterministycznych zachodzi:

$$P =_L Q \Rightarrow P \sim Q \quad (=_L \subseteq \sim)$$

Zatem dla procesów deterministycznych:

$$P =_L Q \iff P \sim Q \quad (=_L = \sim)$$

SYMULACJA

Zmodyfikujmy nieco grę, której użyliśmy do definiowania bisymulacji:

Gracz I

- wybiera jeden z procesów

TURA

- w wybranym procesie wykonuje dowolną tranzycję – proces zmienia stan

Gracz II

- w drugim procesie wykonuje tranzycję o etykiecie takiej, jak tranzycja wybrana przez **Gracza I** – proces zmienia stan

KONIEC TURY

Równoważność tę nazywamy **symulacją**. Oznaczamy ją przez $\sim_S$.

SYMULACJA

Symulacja, podobnie jak bisymulacja, implikuje **trace equivalence**.

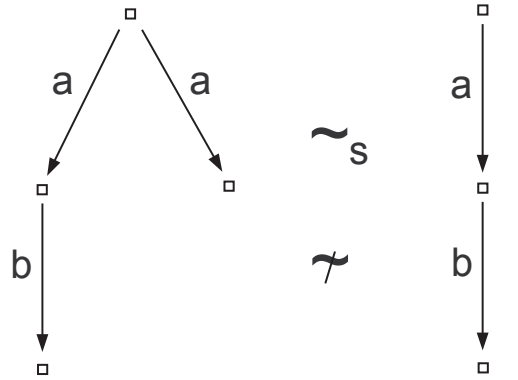
Oczywiście bisymulacja implikuje symulację.

Pytanie: czy symulacja implikuje bisymulację? Czyli czy symulacja jest równa bisymulacji?

SYMULACJA

Pytanie: czy symulacja implikuje bisymulację? Czyli czy symulacja jest równa bisymulacji?

Nie:



SYMULACJA

Mamy zatem następujące implikacje:

$$\sim \subseteq \sim_S \subseteq =_L$$

A dla systemów deterministycznych:

$$\sim = \sim_S = =_L$$

n-BISYMULACJA

Kolejną wariacją na temat bisymulacji jest n-bisymulacja (gdzie n to dowolna liczba naturalna).

Gra, której używamy do jej definiowania jest identyczna jak w wypadku zwykłej bisymulacji, ale liczba tur jest ograniczona przez n. Jeśli gra nie skończy się przed upływem n tur, wygrywa **Gracz II**.

Równoważność tę oznaczamy $\sim_n$.

n-BISYMULACJA

Zachodzą następujace, oczywiste fakty:

- $\sim_0$ to relacja totalna
- $\sim_{n+1} \subseteq \sim_n$
- $=_L \subseteq \sim_1$
- $\sim_n$ i $=_L$ są niezależne dla $n > 1$
- $\sim \subseteq \sim_n$

n-BISYMULACJA

Ciekawsza jest relacja $\sim_\omega =_{def} \bigcap \sim_i$, która alternatywnie można zdefiniować poprzez następującą grę:

Gracz I

- wybiera n – liczbę tur

TURA

- wybiera jeden z procesów
- w wybranym procesie wykonuje dowolną tranzycję – proces zmienia stan

KONIEC TURY

Gracz II

- w drugim procesie wykonuje tranzycję o etykiecie takiej, jak tranzycja wybrana przez **Gracza I** – proces zmienia stan

n-BISYMULACJA

Oczywiście $\sim_\omega \subseteq =_l$. Implikacja w drugą stronę nie zachodzi.

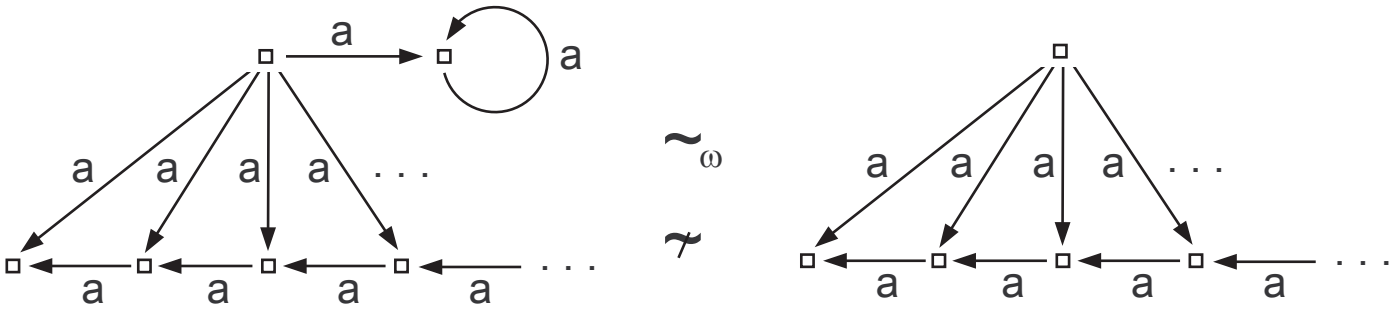
Prawdą jest też, że $\sim \subseteq \sim_\omega$.

Ciekawe jest pytanie, czy zachodzi implikacja odwrotna? Czy $\sim_\omega \subseteq \sim$?

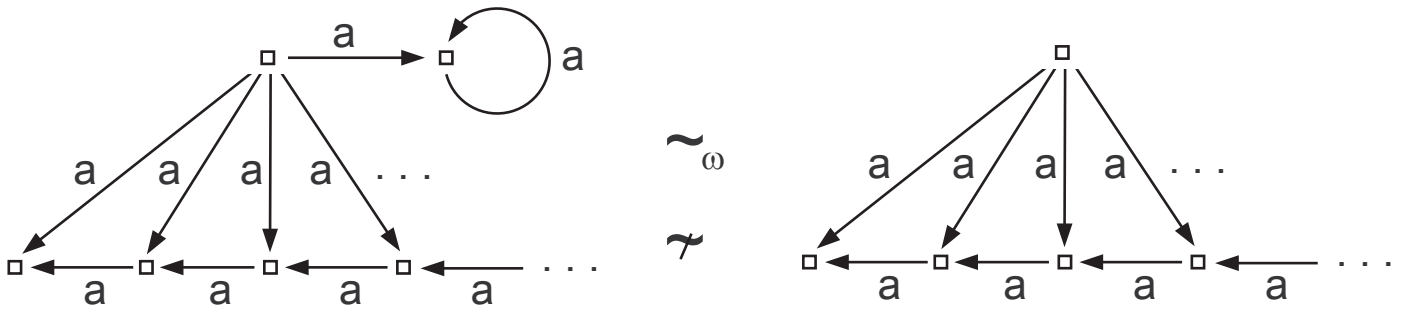
n-BISYMULACJA

Czy $\sim_\omega \subseteq \sim$?

W przypadku ogólnym okazuje się, że nie, co pokazuje następujący przykład:



n-BISYMULACJA



Jeśli dodamy niezbyt restrykcyjne z praktycznego punktu widzenia założenie, aby z dowolnego stanu procesu wychodziło jedynie skończenie wiele tranzycji o danej etykiecie (jest tzw. **własność skończonego obrazu**) to zawieranie zachodzi, a wobec tego:

$$\sim = \sim_\omega$$

DŁUGIE KROKI

Następną wariacją na temat bisymulacji, jaką rozpatrzymy, będzie zmiana definicji pojedynczego ruchu graczy.

W oryginalnej grze **Gracz I** wykonuje pojedyncza tranzycję, na co **Gracz II** odpowiada pojedynczą tranzycją o tej samej etykiecie. Zmiana polega na tym, że **Gracz I** wykonuje dowolny ciąg tranzycji, na co **Gracz II** musi odpowiedzieć ciągiem tranzycji (o odpowiednich etykietach).

Jak ma się tak zdefiniowana relacja do bisymulacji?

DŁUGIE KROKI

Jak ma się tak zdefiniowana relacja do bisymulacji?

Okazuje się, że jest to dokładnie ta sama relacja.

DŁUGIE KROKI

Podobnie jak dla krótkich kroków, możemy dla kroków długich zdefiniować relacje $\sim_n$ i $\sim_\omega$. Zachodzą dla nich następujące, oczywiste fakty:

- $\sim_0$ to relacja totalna
- $\sim_{n+1} \subseteq \sim_n$
- $\sim_1 = =_L$
- $\sim_n \subseteq =_L \quad (n > 1)$
- $\sim \subseteq \sim_n$
- $\sim = \sim_\omega$ dla procesów z własnością skończonego obrazu

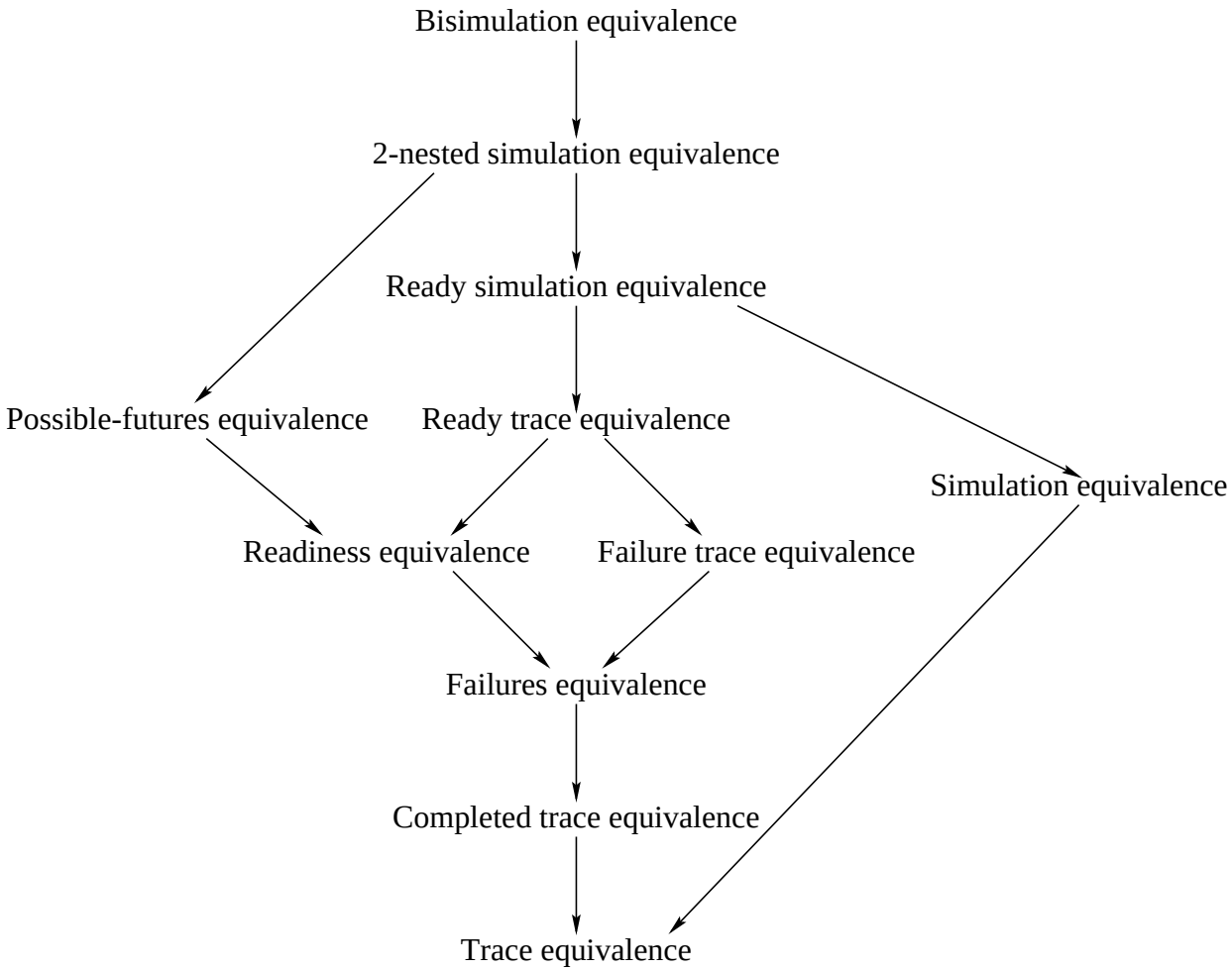
ZŁOŻONOŚĆ OBLICZENIOWA

To, co odróżnia bisymulację od $\sim_i$ to złożność obliczeniowa.

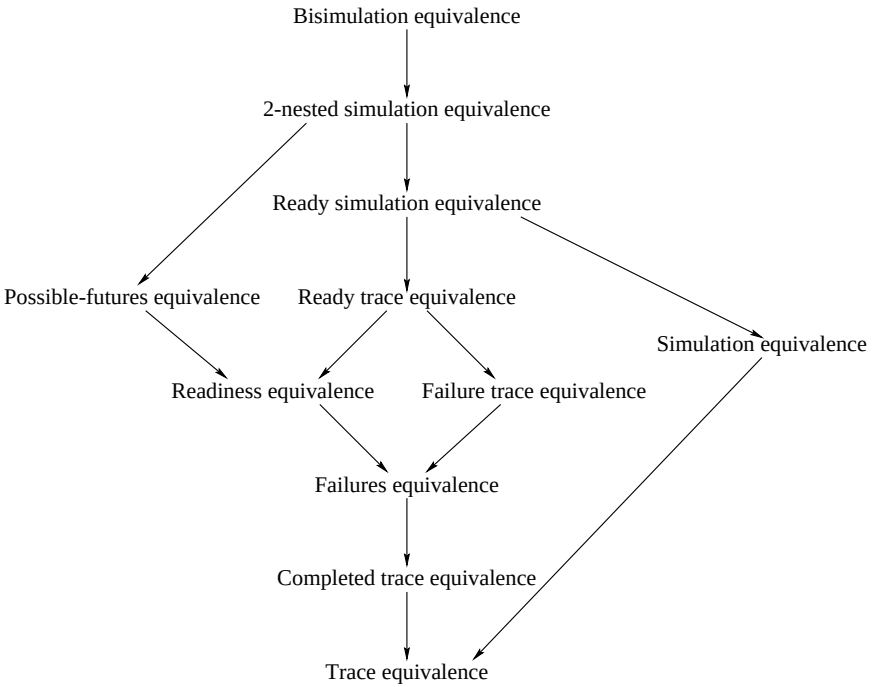
Niech n oznacza sumaryczną liczbę stanów procesów **P** i **Q** zaś m ich sumaryczną liczbę tranzycji. Wówczas:

równoważność	**złożoność**
bisymulacja	$O(m \log n)$
symulacja	$O(mn)$
$\sim_i \ \ (i > 0)$	PSPACE-zupełne

SPEKTRUM VAN GLABBEEK'A

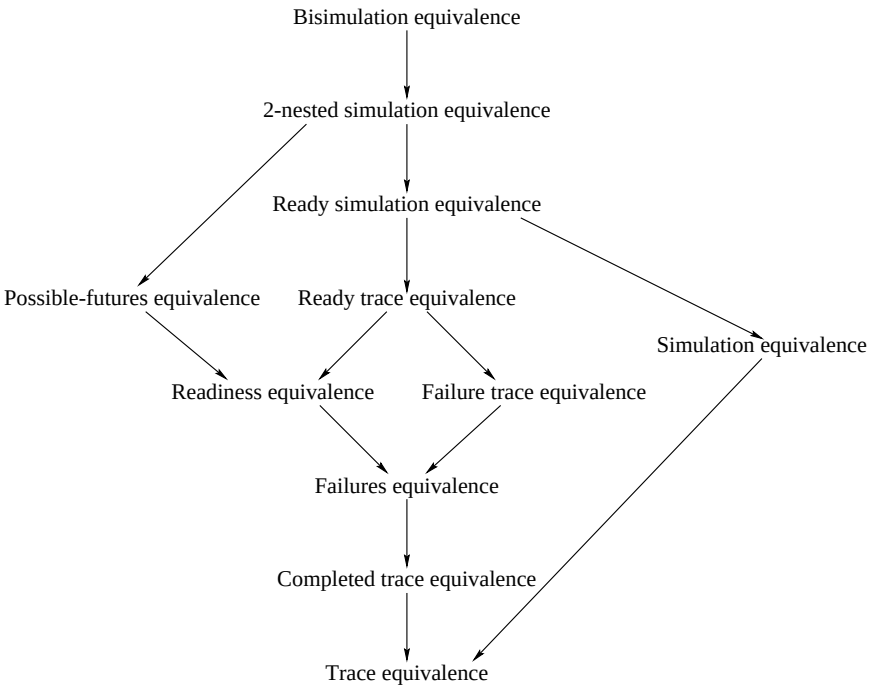


SPEKTRUM VAN GLABBEEK'A



Wiemy, że dla systemów deterministycznych $\sim = =_L$. Widać z tego, że dla systemów deterministycznych wszystkie powyższe równoważności są równe.

SPEKTRUM VAN GLABBEEK'A



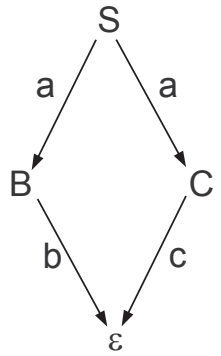
Dla procesów skończenie stanowych równoważności symulacyjne są obliczalne w czasie wielomianowym, zaś niesymulacyjne są PSPACE-zupełne.

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

Zajmiemy się teraz klasami procesów nieskończenie stanowych. Oczywiście będą nas interesowały jedynie takie procesy, których przestrzeń stanów jest skończenie generowana.

Wróćmy do analogi z JAiO i obserwacji, że procesy, które rozważamy, to automaty.

Każdemu automatowi skończonemu odpowiada gramatyka prawostronna, generująca ten sam język regularny co on.



$$\begin{array}{lcl} S \to aB & & S \xrightarrow{a} B \\ S \to aC & \Rightarrow & S \xrightarrow{a} B \\ B \to b & & B \xrightarrow{b} \varepsilon \\ C \to c & & C \xrightarrow{c} \varepsilon \end{array}$$

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

W hierarchii Chomskiego oczko wyżej od języków regularnych znajdują sie języki bezkontekstowe, generowane przez gramatyki bezkontekstowe. W gramatykach tych produkcje mają postać:

$$V \rightarrow (a + V)^*$$

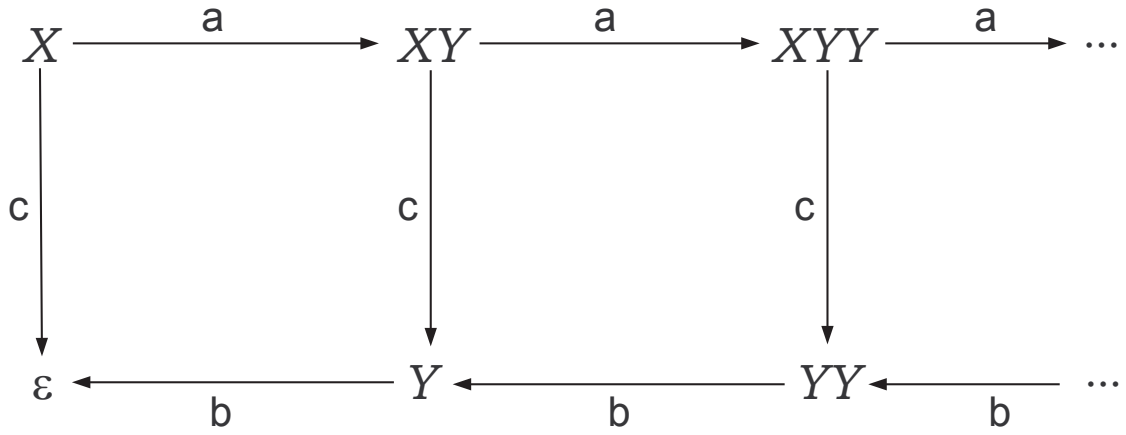
Każdą* gramatykę kontekstową można jednak przekształcić do **postaci normalnej Greibach**:

$$V \rightarrow aV^* \qquad \Rightarrow \qquad V \stackrel{a}{\rightarrow} V^*$$

Przypomnijmy dodatkowo, że każde wyprowadzalne słowo można wyprowadzać w sposób **standardowy**, tzn. rozwijając zawsze skrajnie lewy nieterminal.

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

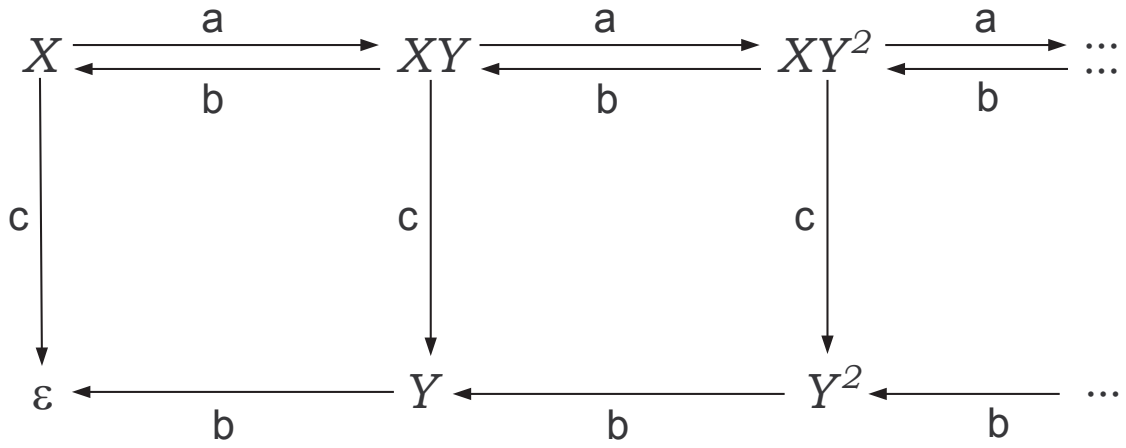
Biorąc zbiór reguł (produkcji) w postaci normalnej Greibach i rozwijąc zawsze skrajnie prawą zmienną procesową (nieterminal), dostajemy proces z klasy **procesów bezkontekstowych**, zwanej także **BPA** (**Basic Process Algebra**). Stanami są słowa nad V, czyli zbiorem zmiennych procesowych.



$$X \xrightarrow{a} XY$$
$$X \xrightarrow{c} \varepsilon$$
$$Y \xrightarrow{b} \varepsilon$$

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

Powyższą konstrukcję można powtórzyć z dokładnością do jednej zmiany:



$$X \xrightarrow{a} XY$$
$$X \xrightarrow{c} \varepsilon$$
$$Y \xrightarrow{b} \varepsilon$$

Jeśli dopuścimy przemienność stałych procesowych, a więc zamienimy stany ze słów na multipodzbiory V, dostaniemy klasę procesów zwaną **BPP** (**Basic Parallel Processes**).

Pierwszy, nieprzemienny wariant odpowiada złożeniu sekwencyjnemu procesów, zaś drugi, przemienny – złożeniu równoległemu.

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

Tę metodę generowania procesów, polegającą z grubsza na przepisywaniu zmiennych procesowych według reguł postaci $\alpha \stackrel{a}{\rightarrow} \beta$, nazywa się **rewrite systems**. Nakładając różne ograniczenia na zbiór reguł dostaniemy szereg ważnych klas procesów:

ograniczenia	**sekwencyjne**	**równoległe**
$\alpha \in V^*,\ \beta \in V^*$	PDA	PN
$\alpha \in QS,\ \beta \in QS^*$	PDA	MSA
$\alpha \in V,\ \beta \in V^*$	BPA	BPP
$\alpha \in V,\ \beta \in V \cup \{\varepsilon\}$	FSA	FSA

gdzie $V = Q \uplus S$, FSA = Finite state automata = automaty skończone, PDA = Pushdown automata = automaty (skończone) ze stosem, MSA = Multiset automata, a PN = Petri nets = sieci Petriego.

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

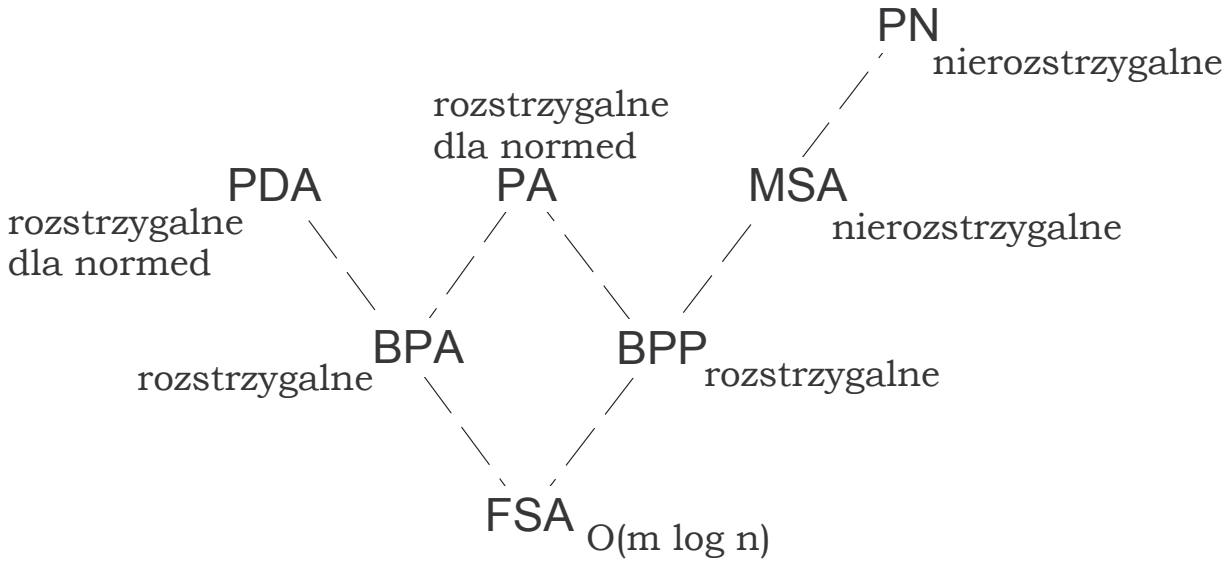
W przypadku sekwencyjnym daje się zauważyć dwie ciekawe rzeczy:

- Najogólniejsza klasach reguł daje, tak samo jak mniej ogólna, PDA
- Chociaż gramatyki bezkontekstowe i automaty ze stosem rozpoznają tę samą klasę języków, to już z dokładnością do izomorfizmu procesów ta druga jest ogólniejsza. Można pokazać więcej: klasy te są różne z dokładnością do bisymulacji.

Ogólnie można pokazać, że wszystkie wymienione klasy procesów są różne z dokładnością do bisymulacji.

PROCESY NIESKOŃCZENIE STANOWE - REWRITE SYSTEMS

Mamy teraz do czynienia z nieskończoną przestrzenią stanów, więc nie tyle złożoność obliczeniowa jest problemem, co sama rozstrzygalność. Dla bisymulacji wygląda to tak:



Zaś wszystkie pozostałe równoważności ze spektrum van Glabbeek'a są nierozstrzygalne dla każdej klasy poza FSA.

SŁABA BISYMULACJA

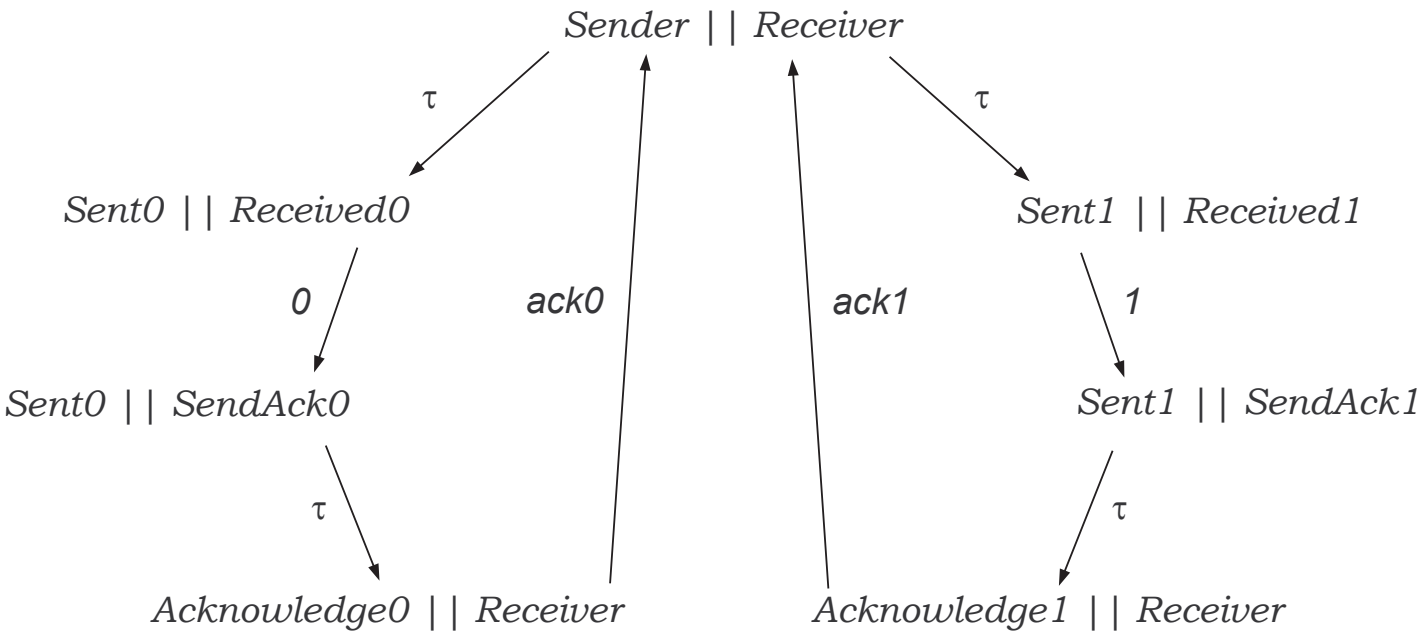
Wróćmy na chwilę do zastosowań praktycznych. Jak można zauważyć bisymulacja nie jest aż tak bardzo użyteczna do weryfikacji – rozróżnia ona zbyt wiele procesów. Dlatego wprowadzono jej odmianę zwaną **słabą bisymulacją**.

Wyróżnia się jedno specjalne zdarzenie, zwyczajowo oznaczane τ. Będzie ono odpowiadać każdemu zdarzeniu wewnętrznemu procesu.

$$
\begin{aligned}
Sender &= transmit0.Sent0 + transmit1.Sent1 \\
SentB &= \overline{transmit\_ackB}.AcknowledgeB \\
AcknowledgeB &= \mathsf{ackB}.Sender \\
\\
Receiver &= \overline{transmit0}.Received0 + \overline{transmit1}.Received1 \\
ReceivedB &= \mathsf{B}.SendAckB \\
SendAckB &= transmit\_ackB.Receiver \\
\\
System &= (Sender||Receiver)\backslash\{transmitB, transmit\_ackB\}
\end{aligned}
$$

SŁABA BISYMULACJA

Graf stanów osiągalnych:



SŁABA BISYMULACJA

Aby zdefiniować bisymulację słabą stosujemy tę samą grę co dla zwykłej bisymulacji (zwanej silną), ale zmieniamy krok.

Krok z etykietą $a \neq \tau$ to dowolna liczba tranzycji τ następnie dokładnie jedna tranzycja a, a potem znów dowolna liczba tranzycji τ.

Krok z etykietą τ to dowolna liczba tranzycji τ (także 0).

PERFORMANCE EQUIVALENCE

Rozszerzamy definicję procesu w ten sposób, że każda równoległa składowa ma swój lokalny zegar. Wykonanie tranzycji w tej składowej powoduje aktualizację tego zegara.

Dla ustalenia rozszerzmy sobie znaną klasę: BPP do TBPP. Stan procesu będzie multipodzbiorem iloczynu kartezjańskiego N i V:

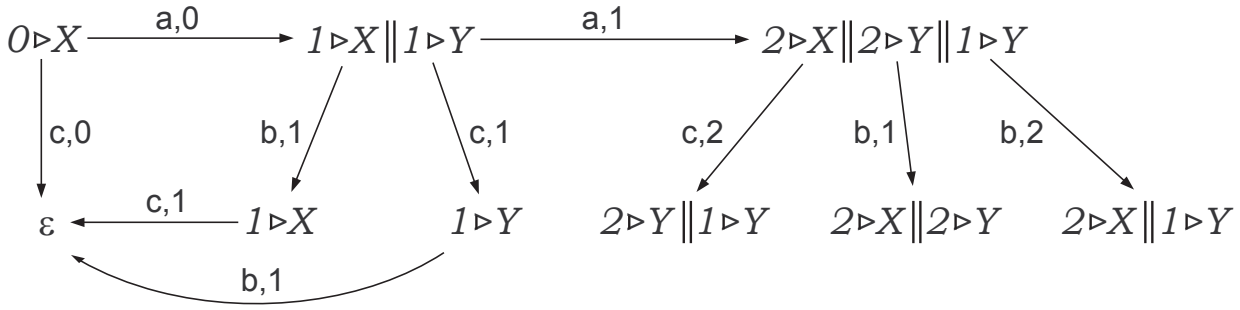
$$t_1 \rhd X_1 || t_2 \rhd X_2 || \ldots || t_n \rhd X_n$$

Załóżmy, że w opisie procesu mamy regułę $X_1 \xrightarrow{a} Y_1 Y_2 \ldots Y_k$. Wówczas wykonanie tranzycji wygenerowanej z tej reguły spowoduje zmianę powyższego stanu na:

$$(t_1 + t(a)) \rhd Y_1 || \ldots || (t_1 + t(a)) \rhd Y_k || t_2 \rhd X_2 || \ldots || t_n \rhd X_n$$

PERFORMANCE EQUIVALENCE

Tranzycje będą etykietowane nie tylko samym zdarzeniem, a parą: zdarzenie i liczba. Liczba to wartość lokalnego zegara przed wykonaniem tranzycji.



$$X \xrightarrow{a} XY$$
$$X \xrightarrow{c} \varepsilon$$
$$Y \xrightarrow{b} \varepsilon$$

Bisymulacja dla TBB nazywa się **performance equivalence**. Jest ona obliczalna w czasie wielomianowym.

Dziękuję

Pytania?