Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Etyka procesów sieci Petriego w świetle teorii śladów

rozprawa doktorska

Joanna Jółkowska

Uniwersytet Mikołaja Kopernika w Toruniu
Wydział Matematyki i Informatyki

Promotor rozprawy
dr hab. Edward Ochmański, prof. UMK
Uniwersytet Mikołaja Kopernika w Toruniu
Wydział Matematyki i Informatyki

Toruń, czerwiec 2008

Oświadczenie autora pracy:

Świadoma odpowiedzialności prawnej oświadczam, że niniejsza rozprawa doktorska została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

....
data podpis autora rozprawy

Oświadczenie promotora rozprawy:

Potwierdzam, że niniejsza rozprawa została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej do oceny przez recenzentów.

....
data podpis promotora rozprawy

Etyka procesów sieci Petriego w świetle teorii śladów*

Słowa kluczowe:
Sieci Petriego, współbieżność, języki śladów, uczciwość

AMS Mathematical Subject Classification 2000:
68Q85 Models and methods for concurrent and distributed computing

Streszczenie

W pracy badane są etyczne aspekty zachowań (sprawiedliwość, uczciwość, bezkonfliktowość) różnych typów sieci Petriego. Podstawowym narzędziem stosowanym w pracy jest teoria śladów, przede wszystkim śladów nieskończonych.

Definiujemy relację niezależności indukowaną przez dowolny system tranzycyjny, tworząc w ten sposób współbieżny system tranzycyjny. Wprowadzamy pojęcie systemów o zachowaniu śladowym i badamy pewne problemy decyzyjne, związane z obliczaniem relacji zależności i rozstrzyganiem, czy dana sieć ma zachowanie śladowe. Pokazujemy, że oba te problemy są rozstrzygalne dla sieci elementarnych i markowanych, a nierozstrzygalne w rozszerzeniach sieci markowanych.

Ponieważ rozszerzenia sieci powodują występowanie zjawisk nie spotykanych w sieciach klasycznych (elementarnych i markowanych), np. nie mają one własności diamentu, zaproponowana została również bardziej precyzyjna definicja konfliktu. Następnie badamy wystąpienia konfliktów oraz istnienie bezkonfliktowych obliczeń w rozszerzeniach sieci elementarnych. Okazuje się, że każde sprawiedliwe obliczenie, zaczynające się od stanu konfliktowego, musi zawierać konfliktowy krok. Pozwala to na skonstruowanie algorytmu, wybierającego tylko bezkonfliktowe obliczenia sprawiedliwe spośród wszystkich obliczeń sieci.

Główna część pracy to przeniesienie hierarchii uczciwości dla obliczeń sekwencyjnych na obliczenia współbieżne (procesy). W pracy zostaje dokładnie przebadana ta hierarchia, najpierw ogólnie, potem konkretnie dla podstawowych klas sieci Petriego – elementarnych i markowanych. Sformułowane zostają też nieprzeplotowe definicje uczciwości procesów i porównane z wcześniejszymi. Ponieważ egzystencjalna uczciwość nie zawsze oznacza uniwersalną, określone też zostają kryteria, które muszą być spełnione, aby w danym systemie skończenie stanowym wszystkie procesy były stabilne ze względu na uczciwość.

* Praca współfinansowana przez Ministerstwo Nauki i Szkolnictwa Wyższego z grantu promotorskiego N N206 2149 33.

Ethics of Petri net processes in the light of trace theory[*]

Keywords:
Petri nets, concurrency, trace languages, fairness

AMS Mathematical Subject Classification 2000:
68Q85 Models and methods for concurrent and distributed computing

Abstract

This thesis deals with ethical aspects of computations (justice, fairness, conflict-freeness) of various kinds of Petri nets. The basic tool used in the thesis is trace theory, especially infinite traces.

We define the independency relation induced by arbitrary transition system, forming this way an asynchronous transition system. We introduce the notion of traceability of transition systems and study some decision problems, related to computing independency and deciding traceability for basic classes of Petri nets. We show that both problems are decidable for elementary and place/transition nets and undecidable in broader classes of nets – inhibitor, reset and transfer nets.

Since extensions of nets admit phenomena unknown in traditional nets (elementary and place/transition), for instance they have not diamond property, we propose a more precise definition of conflict. We study occurrences of conflicts and existence of conflict-free computations in extensions of elementary nets. We show that any just computation starting from a conflict state contains a conflict step. This result allows to construct an algorithm, selecting only conflict-free just computations from among all computations of a given net.

The main part of the thesis generalizes the well-known fairness hierarchy for sequential computations to that of traces (concurrent processes). The fairness hierarchy for traces is similar, but more involved than for sequences. We study this hierarchy, first in general, abstracting from concrete concurrent system, then for basic classes of Petri nets – elementary and place/transition nets. We define also the fairness notions in a non-interleaving way and compare them with the former ones. Since existential fairness is not always equal to universal, we formulate conditions that have to be met by a transition system (with finite number of states) to ensure that all processes of a system are stable as regards fairness.

[*] This PhD thesis has been partially supported by Ministry of Science and Higher Education of Poland, grant N N206 2149 33.

Spis treści

1 Wstęp

Tematyka rozprawy

Sieci Petriego zaproponowane przez C. A. Petriego [40] w roku 1962 (tłumaczenie angielskie w 1966) stanowią obecnie jedno z najbardziej uniwersalnych graficznych narzędzi matematycznych modelujących działania systemów dynamicznych. Stanowią niejako pomost pomiędzy praktyką a teorią. Wykorzystuje się je w różnych dziedzinach – sieć może modelować zarówno programy komputerowe, jak i reakcje chemiczne, ruch uliczny czy życie komórki. Szczególnie przydatne są w badaniu systemów współbieżnych, gdzie potrzebne jest określenie wzajemnych relacji pomiędzy poszczególnymi elementami systemu, przewidywanie przyszłego zachowania, wykrywanie niepożądanych ścieżek wykonania itp.

Każda sieć Petriego składa się z akcji (tranzycji) oraz miejsc, w których przechowywane są zasoby umożliwiające wykonanie poszczególnych akcji, jak również powstające w wyniku wykonania tychże akcji. Miejsca mogą być też rozumiane jako warunki, które muszą być spełnione, aby akcja mogła się wykonać. Wykonanie akcji powoduje zmianę stanu sieci: pewne zasoby zostają skonsumowane, nowe – wyprodukowane (zmieniają się warunki). Zachowanie sieci nie jest z góry określone; w każdym stanie sieci może być umożliwionych wiele akcji i w związku z tym jest wiele możliwych dróg działania (obliczeń) sieci.

Sieci są analizowane pod różnymi względami – bada się zarówno jej zachowanie (własności dynamiczne), jak i strukturę (własności statyczne). Do własności dynamicznych należy między innymi rozstrzyganie, czy pewien z góry zadany stan jest osiągany przez jakieś obliczenie sieci. Jest to tzw. *problem osiągalności*, jeden z najtrudniejszych problemów związanych z sieciami Petriego. Problem ten pozostawał otwarty przez kilkanaście lat, rozstrzygnięty został pozytywnie przez Mayra [31] w 1981 roku i Kosaraju [26] w 1982 roku. Blisko związany z tym problemem jest problem osiągalności stanu pustego (równoważny osiągalności stanu dowolnego) i problem osiągalności stanu z ustalonym warunkiem pustym. Innym znanym problemem, aczkolwiek dużo łatwiejszym, jest *problem pokrywalności* (tzn. osiągalności stanu pokrywającego zadany). Jego rozstrzygalność pokazana została już w pierwszych latach rozwoju teorii sieci przez Karpa/Millera [25] w 1969 roku za pomocą tzw. grafu pokrywalności.

Powyższe problemy dotyczą sieci jako całości, ale rozważa się też własności związane z poszczególnymi obliczeniami sieci. Ma to duże znaczenie praktyczne, ponieważ dobry projekt systemu często wymaga, aby obliczenia modelującej go sieci spełniały pewne założenia. Zbiór takich dobrych własności (norm działania) nazywamy etyką obliczeń. Do tych własności należy przede wszystkim uczciwość, która ma zapobiegać „zagłodzeniu” (tzn. wyłączeniu z pracy) jednej z akcji (bądź większego fragmentu) systemu. Badania uczciwości działania systemów współbieznych zapoczątkował słynny przykład pięciu filozofów Dijkstry [14] z 1971 roku. Pojęcie

etyki obliczeniowej sformalizowali (i tak nazwali) Lehman/Pnueli/Stavi [29] w 1981 roku. W zależności od potrzeb (stopnia umożliwienia zagłodzonej akcji) definiuje się różne poziomy uczciwości; najważniejsze to sprawiedliwość, uczciwość i superuczciwość. Taką klasyfikację stopni uczciwości (a nawet bardziej rozbudowaną, nieskończoną) zaproponował Best [3] w 1984 roku.

Jedną z sytuacji, które mogą prowadzić do nieuczciwości, jest konflikt. Jest to taki stan sieci, w którym każda z dwóch (lub więcej) akcji ma wystarczającą ilość zasobów, aby się wykonać, ale zasoby te są wystarczające tylko dla jednej z nich. Jeśli obliczenie ma być uczciwe, powinna być wykonana ta akcja, która dotąd wykonywała się rzadziej. Innym rozwiązaniem jest ominięcie konfliktu – takie projektowanie systemów, aby sytuacje konfliktowe nie występowały. Dlatego etykę obliczeń będziemy rozumieć szerzej – to nie tylko różne rodzaje uczciwości, ale też i bezkonfliktowość.

W rozprawie badać będziemy jednak nie tyle obliczenia sieci, co procesy, czyli zbiory obliczeń równoważnych w sensie zamienności niektórych akcji (niezależnych, czyli mogących się wykonywać współbieżnie). Do badania procesów sieci użyjemy teorii śladów.

Pojęcie śladu (*trace*) zaproponowane zostało przez Mazurkiewicza [32] w 1977 roku do badania zachowania systemów współbieżnych. Ślady to elementy monoidu ilorazowego – zwanego monoidem śladów – otrzymanego jako iloraz monoidu wolnego przez kongruencję wyznaczoną przez relację niezależności (współbieżności) akcji. Monoidy śladów okazały się niezwykle ciekawym obiektem matematycznym, o często zaskakujących właściwościach. Teoria śladów, dobrze już opisana i ustabilizowana, jest nadal aktualna i stale rozwijana. Wszechstronną monografią teorii śladów jest książka [13].

Systemy komputerowe, a również wiele systemów rzeczywistych, oparte są na pracy w czasie nieograniczonym. Modelowanie działania takich systemów prowadzi do pojęcia obliczeń nieskończonych; w przypadku systemów współbieżnych – nieskończonych procesów współbieżnych. Modelem takich procesów, zastosowanym w niniejszej rozprawie, są ślady nieskończone (Mazurkiewicz [33], obszerna monografia Gastin/Petit [20]).

Zawartość rozprawy

Rozprawa składa się z sześciu rozdziałów, z których pierwszy jest niniejszym wstępem, zawierającym ogólne streszczenie pracy i jej wyników.

W rozdziale 2 przedstawiam podstawowe pojęcia używane w dalszych rozdziałach, w szczególności opis różnych typów sieci Petriego (2.2), podstawowe pojęcia i fakty teorii śladów (2.3) oraz definicje uczciwości obliczeń (2.5).

Ponieważ chcemy użyć teorii śladów do badania procesów sieci, pojawia się zagadnienie określenia, które akcje są niezależne w sieciach Petriego. Mazurkiewicz [32] zaproponował definicję strukturalną dla sieci elementarnych (dwie akcje są zależne, jeśli mają wspólne warunki). W podrozdziale 2.4 przeprowadzona jest dyskusja na temat możliwości zaadaptowania tej definicji do pozostałych klas sieci, w wyniku

której proponujemy inną definicję, opartą na zachowaniu sieci, dającą się zastosować do dowolnych systemów tranzycyjnych. Otóż dwie akcje są niezależne, jeśli zamiana kolejności ich wykonań nie wyprowadza nas poza język rozważanej sieci:

Definicja 2.25. *Niezależność indukowana przez język*

$$a\mathrm{I}_L b \Leftrightarrow (\forall u,v \in A^*)\ (uabv \in L \Leftrightarrow ubav \in L)$$

Dalsza część pracy opiera się na tej właśnie definicji.

W rozdziale 2.2.4 w podobny zachowaniowy sposób definiujemy konflikt (dwie akcje są w konflikcie w pewnym stanie, jeśli są w nim umożliwione, a po wykonaniu jednej z nich druga już nie jest umożliwiona lub stan osiągany w wyniku obu zależy od kolejności ich wykonania), a następnie w rozdziale 3 badamy możliwość unikania konfliktów w sieciach elementarnych i ich rozszerzeniach, tzw. sieciach bezpiecznych (*safe nets* – Badouel/Darondeau [2]). Okazuje się, że:

> **Stwierdzenie 3.8.** W sieciach bezpiecznych każde sprawiedliwe obliczenie rozpoczynające się w stanie konfliktowym zawiera krok konfliktowy.

Wynik ten umożliwił opracowanie algorytmu, który ze zbioru wszystkich obliczeń sieci wybiera zbiór obliczeń bezkonfliktowych.

Ponieważ definicja niezależności akcji zaproponowana w rozdziale 2.4 ma charakter dynamiczny, pojawia się pytanie, czy da się ją wyznaczyć dla dowolnej sieci. W rozdziale 4 pokazuję, że problem wyznaczenia zachowaniowej relacji niezależności jest rozstrzygalny w sieciach elementarnych i markowanych (*place/transition nets*), a nierozstrzygalny w pewnych rozszerzeniach sieci markowanych.

> **Twierdzenie 4.6.** Problem zależności „Czy dane akcje *a* i *b* są zależne?" jest rozstrzygalny w klasie sieci markowanych.

> **Twierdzenie 4.15.** Problem zależności jest nierozstrzygalny dla sieci inhibitorowych, czyszczących i przerzucających.

Oba dowody zbudowane są w oparciu o problem osiągalności. W pierwszym przypadku bezpośrednio korzystamy z faktu rozstrzygalności problemu osiągalności w sieciach markowanych, a w drugim wystarcza nierozstrzygalność problemu pustości warunku w sieciach rozszerzonych. Okazuje się przy okazji, że nie ma bezpośredniego dowodu równoważności problemu osiągalności z problemem pustości warunku dla sieci rozszerzonych, tak więc nierozstrzygalność osiągalności niekoniecznie musi implikować nierozstrzygalność pustości warunku.

Innym zagadnieniem decyzyjnym, którym zajmuję się w tym rozdziale, jest problem rozstrzygania, czy zachowanie danej sieci może być w pełni opisywane śladami. Okazuje się bowiem, że ślady mogą gubić pewne informacje o (lokalnej) współbieżności niektórych akcji – wystarczy jeden stan, w którym akcje są zależne, aby

już były globalnie zależne, pomija się wtedy wykonania współbieżne w innych stanach. Wprowadzam więc pojęcie systemu o zachowaniu śladowym: jest to system, w którym akcje zależne globalnie nie mogą się wykonywać współbieżnie, lub równoważnie, jeśli choć raz akcje mogą się wykonać współbieżnie, to są globalnie niezależne:

> **Definicja 4.17.** *Zachowanie śladowe*
> Zachowanie systemu tranzycyjnego $S=(A,Q,q_0)$ jest *śladowe* wtedy i tylko wtedy, gdy $(\forall a,b\in A)\ ((\exists q\in Q)\ a||_q b\ ) \Rightarrow a\mathrm{I}_S b$.

Rozstrzyganie, czy dana sieć ma zachowanie śladowe, opiera się na rozstrzyganiu, czy istnieje para akcji zależnych i lokalnie współbieżnych. Pokazuję, że problem ten jest rozstrzygalny w klasie sieci markowanych (dowód wykorzystuje rozstrzygalność problemu osiągalności).

> **Twierdzenie 4.19.** Problem śladowości zachowania „Czy zachowanie danej sieci markowanej jest śladowe?" jest rozstrzygalny.

W rozszerzeniach sieci markowanych problem śladowości okazuje się nierozstrzygalny. Do pokazania tego faktu wykorzystujemy nierozstrzygalność problemu pustości warunku.

> **Twierdzenie 4.21.** Problem śladowości zachowania jest nie-rozstrzygalny dla sieci inhibitorowych, czyszczących i przerzucających.

Wreszcie w rozdziale 5 przedstawiam dokładną hierarchię procesów sieci Petriego ze względu na ich właściwości etyczne, przy czym procesy sieci traktowane są jako ślady, dla których relacja niezależności generująca odpowiednią kongruencję jest niezależnością zdefiniowaną w podrozdziale 2.4. Pojawia się pytanie, jak przenieść definicję uczciwych obliczeń na ślady? Jeśli potraktujemy ślady jako zbiory swoich linearyzacji (podejście przeplotowe), nasuwa się naturalne rozróżnienie na własności egzystencjalne (jakieś obliczenie procesu ma rozważaną własność) i uniwersalne (wszystkie obliczenia procesu mają tę własność). W pracy badam tak zdefiniowane klasy procesów ze względu na sprawiedliwość, uczciwość i superuczciwość dla sieci elementarnych i markowanych. Okazuje się, że każdy równoważnik obliczenia superuczciwego jest superuczciwy (niezależnie od rodzaju sieci, a nawet niezależnie od urządzenia generującego język), oraz że każdy równoważnik obliczenia sprawiedliwego jest sprawiedliwy w sieciach elementarnych i markowanych bezpętelkowych.

> **Stwierdzenie 5.2.** W dowolnym systemie tranzycyjnym każdy proces egzystencjalnie superuczciwy jest uniwersalnie super-uczciwy.

Stwierdzenie 5.6. W elementarnych sieciach Petriego każdy proces egzystencjalnie sprawiedliwy jest uniwersalnie sprawiedliwy: uJUST = eJUST.

Stwierdzenie 5.8. W bezpętelkowych sieciach markowanych uJUST = eJUST.

Możliwe jest także inne podejście do zdefiniowania pewnych własności obliczeń w wersji śladowej – podejście nieprzeplotowe, w którym definicje odpowiednich pojęć opierają się na własnościach skończonych prefiksów śladu. Okazuje się jednak, że klasy procesów zdefiniowane zgodnie z tym podejściem pokrywają się z pewnymi klasami zdefiniowanymi przeplotowo:

Stwierdzenie 5.15. Nieprzeplotowa superuczciwość to dokładnie przeplotowa superuczciwość.

Stwierdzenie 5.16. Nieprzeplotowa sprawiedliwość to dokładnie przeplotowa sprawiedliwość egzystencjalna.

Stwierdzenie 5.17. Nieprzeplotowa uczciwość to dokładnie przeplotowa sprawiedliwość uniwersalna.

Przypadek, gdy pewna własność ma charakter egzystencjalny, ale nie uniwersalny, może być niepożądany z punktu widzenia praktycznego konstruowania systemu współbieżnego o zadanych własnościach. W podrozdziale 5.4 badam wpływ konfuzji na tego rodzaju niestabilność procesów. Rozdział kończą efektywne kryteria charakteryzujące równość pewnych klas procesów w systemach skończenie stanowych:

- Twierdzenie 5.21 – charakteryzacja równości klas procesów egzystencjalnie uczciwych i uniwersalnie uczciwych oraz
- Twierdzenie 5.22 – charakteryzacja równości klas procesów uniwersalnie uczciwych i superuczciwych.

W rozdziale 6 podsumowuję uzyskane wyniki i sygnalizuję kilka problemów otwartych, związanych z tematyką rozprawy.

Większość wyników niniejszej rozprawy została opublikowana w pracach [24, 37, 38].

Podziękowania

Serdecznie dziękuję mojemu promotorowi Edwardowi Ochmańskiemu, za naukową opiekę, pomoc i poświęcony czas oraz koleżankom i kolegom z Torunia.

2 Pojęcia podstawowe

2.1 Słowa, języki, systemy tranzycyjne

Niech A będzie skończonym zbiorem. Monoidem wolnym generowanym przez zbiór A nazywamy monoid $(A^*,\cdot)$, którego elementami są wszystkie skończone ciągi elementów zbioru A, jedynką jest ciąg 0-elementowy $\varepsilon=()$, zwany ciągiem pustym, a operacja złożenia (konkatenacji) jest określona następująco: jeśli $x=(x_1, x_2,\ldots, x_n)$, $y=(y_1, y_2,\ldots, y_m)$ $(n, m \geq 0)$, to $x\cdot y=(x_1, x_2,\ldots, x_n, y_1, y_2,\ldots, y_m)$. Nawiasy, przecinki i znak operacji złożenia będą zawsze pomijane, tak więc powyższe ciągi x, y i xy będą zapisywane jako $x=x_1x_2\ldots x_n$, $y=y_1y_2...y_m$ i $xy=x_1x_2\ldots x_ny_1y_2\ldots y_m$. Zbiór A nazywamy *alfabetem*, elementy A – *literami*, elementy A^* – *słowami*, a podzbiory A^* – *językami*.

W pracy będziemy też rozważać słowa nieskończone nad alfabetem A, zbiór takich nieskończonych słów oznaczamy przez A^ω, natomiast $A^\infty = A^*\cup A^\omega$ jest zbiorem wszystkich (skończonych i nieskończonych) słów nad A. Słowo $u\in A^*$ jest *(skończonym) prefiksem* słowa $w\in A^\infty$ (oznaczenie: $u \leq^{\text{fin}} w$) wtedy i tylko wtedy, gdy istnieje słowo $v\in A^\infty$ takie, że $w=uv$. Dla skończonego słowa $u\in A^*$ jego nieskończone powtórzenie $uuu\ldots$ będzie oznaczane przez u^ω. Operacja złożenia w A^∞ jest operacją częściową: uv jest określone, gdy $u\in A^*$, $v\in A^\infty$.

Długość słowa $w\in A^*$ oznaczamy symbolem $|w|$, gdzie $|w|=n \Leftrightarrow w\in A^n$, dla $n\in\mathbb{N}$. Jeśli $w\in A^\omega$ (jest słowem nieskończonym), to piszemy $|w|=\omega$. Liczbę wystąpień litery $a\in A$ w słowie $w\in A^\infty$ oznaczamy symbolem $|w|_a$ i piszemy $|w|_a=\omega$, jeśli litera a występuje w słowie w nieskończenie wiele razy. Alfabetem słowa $w\in A^\infty$ nazywamy zbiór $\text{Alph}(w)=\{a\in A \mid w\in A^*aA^\infty\}$, czyli zbiór liter występujących w słowie w.

Przez $\text{Pr}(w)$ oznaczać będziemy zbiór wszystkich skończonych prefiksów słowa $w\in A^\infty$, czyli $\text{Pr}(w)=\{u\in A^* \mid u \leq^{\text{fin}} w\}$. Przez $\text{Pr}(L)$ dla $L\subseteq A^\infty$ będziemy oznaczać zbiór wszystkich skończonych prefiksów słów języka L, czyli $\text{Pr}(L)=\bigcup\{\text{Pr}(w) \mid w\in L\}$. Język $L\subseteq A^\infty$ nazywamy *prefiksowo domkniętym* wtedy i tylko wtedy, gdy $\text{Pr}(L)=L$.

Dla $L\subseteq A^*$ oznaczamy: $L^\omega = \{w\in A^\omega \mid \text{Pr}(w)\subseteq\text{Pr}(L)\}$ oraz $L^\infty = \text{Pr}(L)\cup L^\omega$.

Słowo $u\in L$ jest *rozszerzalne* (w prefiksowo domkniętym języku L) wtedy i tylko wtedy, gdy $(\exists a\in A)$ $ua\in L$, a *nierozszerzalne* w przeciwnym przypadku. Lewostronnym ilorazem języka $L\subseteq A^*$ przez słowo $w\in A^*$ nazywamy zbiór $L/w=\{u\in A^*;\ wu\in L\}$.

W pracy będziemy zajmować się głównie językami prefiksowo domkniętymi generowanymi przez pewne systemy współbieżne, w szczególności rozważane będą języki sieci Petriego. Niektóre wyniki jednak będą prawdziwe dla języków generowanych przez dowolne systemy tranzycyjne. Przypomnijmy więc definicję systemu tranzycyjnego:

Definicja 2.1. *System tranzycyjny (deterministyczny)*

System tranzycyjny to trójka (A, Q, q_0), gdzie:

- A jest skończonym zbiorem *akcji*,
- Q jest przeliczalnym zbiorem *stanów*,
  – stan $q\in Q$ to częściowa funkcja $q : A\to Q$,
- $q_0\in Q$ jest *stanem początkowym*.

Można rozważać niedeterministyczne systemy tranzycyjne, w których stany są relacjami w $A \times Q$. Deterministyczne systemy tranzycyjne będą jednak całkowicie wystarczające dla potrzeb tej pracy, gdyż wszystkie sieci Petriego mają deterministyczną naturę (jeśli wykonanie akcji a w stanie M prowadzi do stanu M', to stan M' jest zawsze jednoznacznie określony).

Systemy tranzycyjne przedstawiane będą w postaci grafów skierowanych, których wierzchołkami są elementy Q, krawędzie zaś są etykietowane literami alfabetu A. Ze stanu q_1 istnieje krawędź o etykiecie a do stanu q_2 wtedy i tylko wtedy, gdy $q_1(a)$ jest określone i $q_1(a)=q_2$.

W definicji stany są funkcjami określonymi na A, w naturalny jednak sposób możemy je rozszerzyć do funkcji określonych na A^*. Wtedy $q : A^* \to Q$, gdzie: $q(\varepsilon)=q$ dla każdego $q \in Q$, i $(\forall u,v \in A^*)$ $q(uv)=q(u)(v)$, jeśli $q(u)$ i $q(u)(v)$ są określone, a w przeciwnym przypadku $q(uv)$ jest nieokreślone. Od tego momentu będziemy patrzeć na stany w tym szerszym znaczeniu.

Zapis $q(u)=q(v)$ oznacza, że $q(u)$ i $q(v)$ są określone i równe, lub że oba są nieokreślone. System tranzycyjny (A, Q, q_0) ma *własność diamentu* wtedy i tylko wtedy, gdy dla każdego $q \in Q$ i dowolnych $a,b \in A$ jeśli $q(ab)$ i $q(ba)$ są określone, to $q(ab)=q(ba)$.

Stany osiągalne. Stan q' jest *osiągalny ze stanu* q wtedy i tylko wtedy, gdy istnieje $w \in A^*$ takie, że $q(w)$ jest określone i $q(w)=q'$. Stan q' jest *osiągalny*, jeśli jest osiągalny ze stanu początkowego q_0. Zakładamy od tego momentu, że wszystkie stany z Q są osiągalne.

Obliczenia w systemach tranzycyjnych. Niech $S = (A, Q, q_0)$ będzie systemem tranzycyjnym. Mówimy, że ciąg $w \in A^*$ jest *umożliwiony* w stanie $q \in Q$ wtedy i tylko wtedy, gdy $q(w)$ jest określone. *Obliczeniem skończonym* w S jest każde słowo skończone $w \in A^*$, które jest umożliwione w stanie q_0. *Obliczeniem nieskończonym* w S jest każde nieskończone słowo $a_1a_2\ldots \in A^\omega$, którego każdy skończony prefiks jest obliczeniem w S. Zbiór $L(S)$ wszystkich skończonych obliczeń w S nazywamy *sekwencyjnym zachowaniem* (lub *językiem*) systemu S (lub generowanym przez S). Z definicji wynika, że $L(S)$ jest prefiksowo domknięty.

Zauważmy, że ponieważ stany są funkcjami, to systemy tranzycyjne (w niniejszej pracy) są deterministyczne i każde obliczenie skończone ma jednoznacznie wyznaczony stan końcowy, jak i ciąg stanów pośrednich.

Systemy kanoniczne dla języka. Dla każdego niepustego prefiksowo domkniętego języka $L \subseteq A^*$ możemy określić jego *minimalny system tranzycyjny* $S_L = (A, Q, q_0)$, gdzie stany są utożsamione z niepustymi lewymi ilorazami języka L:

$$Q = \{L/w;\ w \in A^*\} \qquad q_0 = L/\varepsilon$$

Dla dowolnego $q=L/w \in Q$ i dowolnego $u \in A^*$ stan $q(u)$ jest określony, jeśli $L/wu \neq \emptyset$ i wtedy $q(u)=L/wu$.

System S_L będziemy nazywać *kanonicznym systemem tranzycyjnym* dla języka L. Dowolny język możemy traktować jako zachowanie sekwencyjne jego systemu kanonicznego.

Przez $\mathbb{N}$ będziemy oznaczać zbiór liczb naturalnych $\mathbb{N}=\{0,1,\ldots\}$. *Wielozbiorem* nad zbiorem X nazywamy funkcję $f: X \to \mathbb{N}$. Na wielozbiorach nad tym samym zbiorem X określamy działania mnogościowe w następujący sposób:

- suma dwóch wielozbiorów f i g to taki wielozbiór $h=f\cup g$, że $h(x)=f(x)+g(x)$ dla dowolnego $x\in X$;
- różnica dwóch wielozbiorów f i g to taki wielozbiór $h=f-g$, dla którego $h(x)=\max(f(x)-g(x),0)$ dla dowolnego $x\in X$;
- część wspólna dwóch wielozbiorów f i g to taki wielozbiór $h=f\cap g$, dla którego $h(x)=\min(f(x),g(x))$ dla dowolnego $x\in X$;
- relacja inkluzji: $f\subseteq g$ wtedy i tylko wtedy, gdy $f(x)\leq g(x)$ dla dowolnego $x\in X$.

2.2 Sieci Petriego

Sieci Petriego zostały zaproponowane przez C. A. Petriego w 1962 roku jako narzędzie opisu dyskretnych systemów rozproszonych. Pozwalają one modelować systemy współbieżne, stanowią też wygodny aparat matematyczny do opisu i badania specyficznych aspektów zachowania tych systemów.

Podstawowe, z teoretycznego punktu widzenia, klasy sieci Petriego to sieci elementarne i sieci markowane. Na ich bazie konstruuje się bardziej złożone sieci, takie jak sieci kolorowane, obiektowe, priorytetowe, czyszczące, inhibitorowe i inne.

2.2.1 Sieci elementarne i markowane

Definicja 2.2. *Sieć, sieć elementarna, sieć markowana*

- Siecią nazywamy trójkę $N = (A, P, F)$, gdzie
  – A i P są skończonymi zbiorami rozłącznymi; elementy zbioru A nazywamy *akcjami*, elementy zbioru P – *warunkami*;
  – $F\subseteq(A\times P)\cup(P\times A)$ jest relacją binarną, zwaną *relacją przepływu*.
- Siecią elementarną nazywamy parę (N,M_0), gdzie N jest siecią, $M_0\subseteq P$ jest *stanem początkowym*;
- Siecią markowaną nazywamy parę (N,M_0), gdzie N jest siecią, *stan początkowy* M_0 jest wielozbiorem nad P.

Oznaczenia. Dla dowolnej akcji $a\in A$ zbiór $\{p\in P \mid pFa\}$ oznaczać będziemy przez $^\bullet a$ (lub in(a)) i nazywać zbiorem *warunków wejściowych* akcji a, zaś zbiór $\{p\in P \mid aFp\}$ oznaczać będziemy przez $a^\bullet$ (out(a)) i nazywać zbiorem *warunków wyjściowych* akcji a. Przez $^\bullet a^\bullet$ oznaczać będziemy zbiór wszystkich warunków (wejściowych i wyjściowych) związanych z akcją $a\in A$, czyli $^\bullet a^\bullet = {}^\bullet a\cup a^\bullet$. Sieć nazywamy *bezpętelkową*, jeśli nie zawiera warunków, które są jednocześnie wejściowe i wyjściowe dla pewnej akcji, tzn.

$(\forall a \in A)\ {}^\bullet a \cap a^\bullet = \emptyset$. Zgodnie z przyjętym w literaturze założeniem przyjmujemy, że sieci elementarne są bezpętelkowe.

Sieci jako grafy. Sieci będą przedstawiane w postaci skierowanych grafów dwudzielnych, których wierzchołkami będą elementy $A \cup P$, krawędziami zaś elementy relacji F. Akcje będą oznaczane kwadratami, a warunki kółkami. Stany będą oznaczane żetonami w warunkach – w sieciach elementarnych może być co najwyżej jeden żeton w danym warunku, w sieciach markowanych – więcej.

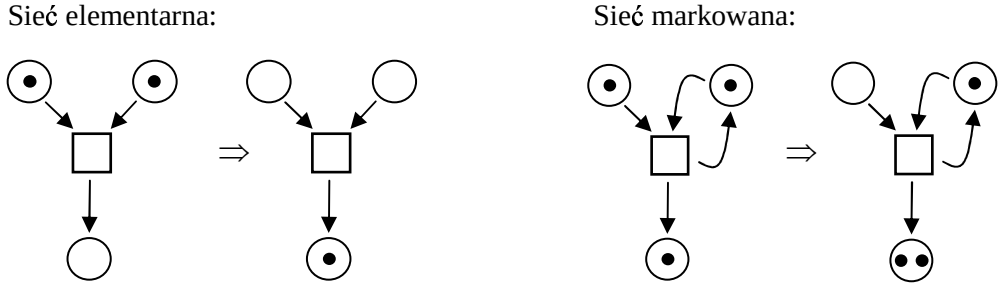
Stany przechowują informację o rozkładzie żetonów w każdym warunku w danym momencie działania sieci. W sieciach elementarnych stan M jest zbiorem warunków posiadających żeton, w sieciach markowanych $M(p)$ jest ilością żetonów w warunku p w stanie M. Stany sieci elementarnych możemy także traktować jako funkcje – o wartościach w zbiorze $\{0,1\}$, tzn. $M : P \to \{0,1\}$. Czasami wygodniej będzie zapisać cały stan w postaci wektora $[M(p_1), M(p_2), \ldots, M(p_k)]$, gdzie $p_1, \ldots, p_k$ są wszystkimi warunkami danej sieci.

Działanie sieci elementarnych. Niech $N=(A,P,F,M_0)$ będzie siecią elementarną. Mówimy, że akcja $a \in A$ jest *umożliwiona* w stanie $M \subseteq P$ wtedy i tylko wtedy, gdy $(\forall p \in {}^\bullet a)\ M(p)=1$ oraz $(\forall p \in a^\bullet)\ M(p)=0$. Akcja umożliwiona może być *wykonana*, a jej wykonanie zmienia stan globalny sieci; nowym stanem jest $M' = (M - {}^\bullet a) \cup a^\bullet$.

Działanie sieci markowanych. Niech teraz $N=(A,P,F,M_0)$ będzie siecią markowaną. Mówimy, że akcja $a \in A$ jest *umożliwiona* w stanie $M \in \mathbb{N}^P$ wtedy i tylko wtedy, gdy $(\forall p \in {}^\bullet a)\ M(p) \geq 1$. *Wykonanie* w stanie M akcji umożliwionej prowadzi do stanu $M' = (M - {}^\bullet a) \cup a^\bullet$ – i tym razem są to operacje na wielozbiorach.

Zapis MaM' oznaczać będzie (w dowolnej klasie sieci), że akcja a jest umożliwiona w stanie M, a jej wykonanie prowadzi do stanu M'. Takie pojedyncze wykonanie akcji umożliwionej będzie nazywane *krokiem sekwencyjnym*. Zapis Ma będzie oznaczał fakt, że akcja a jest umożliwiona w stanie M.

Wykonanie akcji w sieciach Petriego – przykład



Rys. 1. Przykłady działania sieci elementarnych i markowanych

Pojęcie umożliwienia pojedynczej akcji możemy w naturalny sposób rozszerzyć na umożliwienie ciągu akcji $w \in A^*$. Mówimy, że ciąg $w=w_1 \ldots w_n \in A^*$ ($w_i \in A$) jest *umożliwiony w stanie M* i jego wykonanie prowadzi do stanu M' (piszemy MwM'), jeśli Mw_1M_1, $M_1w_2M_2$, …, $M_{n-1}w_nM'$ dla pewnych (jednoznacznie określonych) stanów pośrednich $M_1, M_2, \ldots, M_{n-1}$.

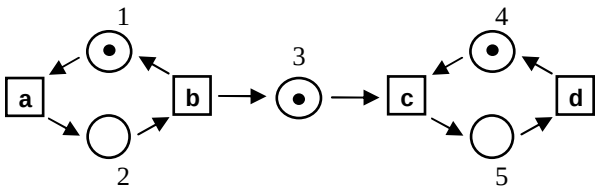
Mówimy, że ciąg akcji $w=w_1 \ldots w_n \in A^*$ jest *umożliwiony w sieci* N=(A,P,F,M_0), jeśli jest on umożliwiony w stanie początkowym M_0. Takie ciągi umożliwione w danej sieci będziemy nazywać *obliczeniami* tej sieci. Czasami zamiast zapisu $w_1 \ldots w_n$ wygodnie będzie wymienić wszystkie stany pośrednie i zapisać ciąg $w_1 \ldots w_n$ w postaci $Mw_1M_1w_2M_2 \ldots M_{n-1}w_nM'$. Stany M, M_1, M_2, …, M_{n-1}, M' będą nazywane *stanami obliczenia* $w \in A^*$. Język L(N) sieci N to zbiór wszystkich obliczeń tej sieci, czyli $L(N)=\{ w \in A^* \mid (\exists M)\ M_0wM \}$.

Stany osiągalne. Niech N=(A,P,F,M_0) będzie (elementarną lub markowaną) siecią Petriego. Mówimy, że stan M' jest *osiągalny ze stanu M* wtedy i tylko wtedy, gdy $(\exists w \in A^*)\ MwM'$. Stan M nazywamy *osiągalnym w sieci* N wtedy i tylko wtedy, gdy M jest osiągalny ze stanu początkowego M_0. Zbiór wszystkich stanów osiągalnych zapisywać będziemy jako RS(N), lub krótko RS przy ustalonej sieci N (z ang. *reachable states*).

Graf osiągalności. *Grafem osiągalności* sieci N=(A,P,F,M_0) nazywamy parę RG=(G,M_0), gdzie $RS \times A \times RS \supseteq G = \{(M,a,M') \mid M \in RS \wedge MaM'\}$.

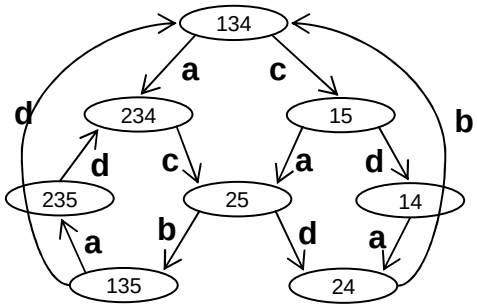
Wierzchołkami grafu osiągalności są stany osiągalne RS, krawędzie są etykietowane akcjami sieci. Dokładniej: z wierzchołka M prowadzi krawędź do wierzchołka M' etykietowana akcją a wtedy i tylko wtedy, gdy M jest stanem osiągalnym z M_0, akcja $a \in A$ jest umożliwiona w stanie M, a po jej wykonaniu sieć przechodzi do stanu M'.

Przykład 2.3 *Sieć i jej graf osiągalności*



Rys. 2. Sieć elementarna N

Zbiorem akcji tej sieci jest $A=\{a,b,c,d\}$; zbiorem warunków jest $P=\{1,2,3,4,5\}$. Stany sieci zapisywać będziemy w postaci ciągów warunków (np. stan początkowy {1,3,4} będzie pisany jako 134). Zbiorem stanów osiągalnych sieci N jest RS={134, 234, 15, 25, 14, 135, 24, 235}. Graf osiągalności RG tej sieci wygląda następująco:



Rys. 3. Graf osiągalności sieci N

Dla sieci markowanych stany będziemy przedstawiać również w postaci ciągów warunków, w których niektóre warunki mogą się powtórzyć, np. na Rys. 40 (Przykład 5.9) wielozbiór {2,4,5,5} jest przedstawiony na grafie osiągalności jako 2455. Czasami będzie wygodniej przedstawić stan sieci markowanej w postaci wektora – wtedy będziemy używać notacji z ukośnikami, np. na Rys. 13 (Przykład 2.32) napis 1/3 oznacza stan, w którym pierwszy warunek ma 1 żeton, a drugi – 3 żetony.

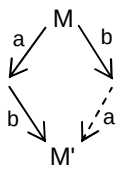
Na stanach, reprezentowanych jako wektory, określamy częściowy porządek: $M \leq M' \Leftrightarrow M \subseteq M'$ (jako wielozbiory) $\Leftrightarrow (\forall p \in P)\ M(p) \leq M'(p)$. Zapis $M < M'$ oznaczać będzie, że $M \leq M'$ oraz $M \neq M'$.

Graf osiągalności jako automat. W sieciach elementarnych liczba stanów osiągalnych nie może być większa niż liczba $2^{|P|}$ wszystkich podzbiorów zbioru P, jest więc zawsze skończona. Graf osiągalności jest zatem zawsze grafem skończonym. Traktując go jako automat skończony nad A^*, ze zbiorem stanów RS, stanem początkowym M_0 i całym RS jako zbiorem stanów końcowych wnioskujemy, że dla każdej sieci elementarnej N zbiór L(N) jej skończonych ciągów wykonań jest regularnym podzbiorem A^*.

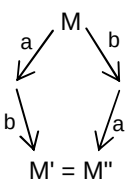
W sieciach markowanych natomiast liczba stanów osiągalnych może być nieskończona, co daje nieskończony graf osiągalności. Grafy osiągalności mogą być wtedy traktowane jako systemy tranzycyjne o przeliczalnej liczbie stanów. Istnieją jednak sieci markowane, których grafy osiągalności są skończone. Dla każdej takiej sieci istnieje pewna stała $c \in \mathbb{N}$ ograniczająca liczbę żetonów w każdym warunku: $(\forall M \in \mathrm{RS})(\forall p \in P)\ M(p) < c$, dlatego nazywane są one *sieciami ograniczonymi*.

„Diamentowość" podstawowych sieci Petriego jest bardzo ważną cechą, ułatwiającą ich badanie i często wykorzystywaną w dowodach innych własności tych sieci. Z kolei „moc" rozszerzonych sieci Petriego jest pośrednio związana z zaburzeniem tej własności. Przypomnijmy najpierw definicje własności diamentu w dwóch podstawowych wersjach – silnej i słabej. Niech a, b będą akcjami, a M, M' – stanami sieci.

Silna własność diamentu:
Jeśli $MabM'$ i Mb, to $MbaM'$.



(Słaba) własność diamentu:
Jeśli $MabM'$ i $MbaM''$, to $M'=M''$.

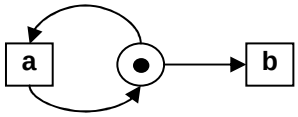


Oczywiście silna własność diamentu implikuje słabą. W dalszej części pracy przez „własność diamentu" będziemy rozumieć słabą własność diamentu.

Fakt 2.4. Sieci elementarne i bezpętelkowe sieci markowane mają silną własność diamentu.

Dowód. Najpierw pokażemy, że jeśli $Ma \wedge Mb \wedge Mab$, to Mba: Jeśli $Ma \wedge Mb \wedge \neg Mba$, to istnieje warunek $p \in {}^\bullet a$, taki, że $M(p)=1$ i $Mb(p)=0$. Zatem $p \in {}^\bullet a \cap {}^\bullet b$, ponieważ $Ma(p)=0$ (bo sieć jest bezpętelkowa), zatem $\neg Mab$. Sprzeczność.
Teraz pokażemy, że $Mab=Mba$. Z definicji $Mab = ((M - {}^\bullet a) \cup a^\bullet) - {}^\bullet b) \cup b^\bullet = ((M - {}^\bullet b) \cup b^\bullet) - {}^\bullet a) \cup a^\bullet = Mba$. ¤

Przykład 2.5. *Fakt 2.4 nie zachodzi dla sieci markowanych z pętelkami*

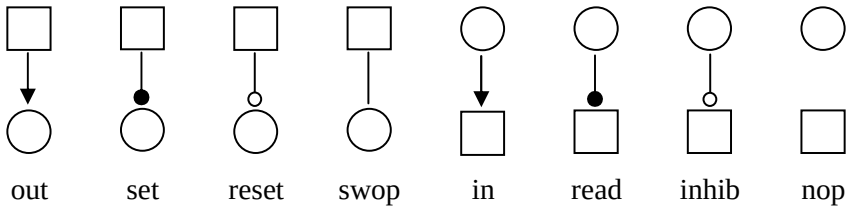


Rys. 4. Zachodzi *Ma*, *Mb* i *Mab*, ale nie *Mba*

Fakt 2.4 (nawet w wersji ze słabą własnością diamentu) nie zachodzi też dla większości rozszerzeń sieci podstawowych (zob. Przykład 2.7).

2.2.2 Rozszerzenia sieci elementarnych

Sieci bezpieczne (*safe nets*) są uogólnieniem sieci elementarnych, wprowadzonym przez Badouela/Darondeau w [2]. W sieciach tych oprócz zwykłych łuków od akcji do warunku (nazwijmy je „out") i z warunku do akcji („in"), dopuszczane są łuki innych typów. Łuki „set" („reset") wstawiają (usuwają) żeton z warunku niezależnie od jego poprzedniej zawartości. Łuki „read" („inhib") sprawdzają, czy jest żeton (czy nie ma żetonu) w warunku. Łuki „swop" zmieniają zawartość placu na odwrotną do zastanej – jeśli był żeton, to czyszczą plac, jeśli nie było, to wstawiają żeton. Notacja i terminologia jest oparta na [2].



Rys. 5. Graficzna reprezentacja łuków w sieciach bezpiecznych

Definicja 2.6. *Sieć bezpieczna*

Siecią bezpieczną nazywamy czwórkę N = (A, P, F, M_0), gdzie
- A i P są skończonymi zbiorami *akcji* i *warunków*;
- $F: A \times P \rightarrow$ {in, out, set, reset, read, inhib, swop, nop} jest *funkcją przepływu*, opisującą rodzaj łuku pomiędzy daną akcją a warunkiem;
- $M_0 \subseteq P$ jest *stanem początkowym*.

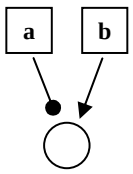
Pojemność warunków jest ograniczona do 1, jak w sieciach elementarnych. Dla sieci bezpiecznych będziemy używać oznaczenia in(a) dla zbioru wszystkich warunków wejściowych akcji a: in(a)={$p \in P \mid F(a,p)$=in} i analogicznie out(a), set(a), reset(a), read(a), inhib(a) i swop(a).

Działanie sieci bezpiecznych. Niech N=(A,P,F,M_0) będzie siecią bezpieczną. Mówimy, że akcja $a \in A$ jest *umożliwiona* w stanie $M \subseteq P$ wtedy i tylko wtedy, gdy in(a)$\cup$read(a)$\subseteq M$ i (out(a)$\cup$inhib(a))$\cap M=\emptyset$. *Wykonanie* w stanie M umożliwionej akcji a zmienia stan sieci; nowym stanem jest stan $M'=M \cup$out(a)$\cup$set(a)–in(a)–reset(a) $\cup$swop^{0,M}(a) – swop^{1,M}(a), gdzie swop^{0,M}(a) oznacza te warunki połączone łukiem typu „swop” z akcją a, które są puste w stanie M: swop^{0,M}(a)={$p \in P \mid F(a,p)$=swop $\wedge$ $M(p)$=0} i analogicznie swop^{1,M}(a)={$p \in P \mid F(a,p)$=swop $\wedge$ $M(p)$=1}.

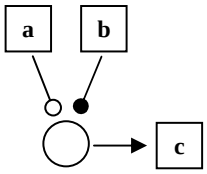
Umożliwienie ciągu akcji, język sieci, zbiór stanów osiągalnych, graf osiągalności – są zdefiniowane tak samo jak dla sieci elementarnych. Podobnie jak dla sieci elementarnych, grafy osiągalności sieci bezpiecznych są zawsze skończone.

Sieci bezpieczne pozwalają na modelowanie zjawisk, które nie mogą wystąpić w sieciach elementarnych. Na przykład sieci elementarne mają własność diamentu, podczas gdy w sieciach bezpiecznych osiągalny jest stan umożliwiający zarówno ciąg *ab*, jak i *ba*, jednak prowadzą one do różnych stanów.

Przykład 2.7. *Brak własności diamentu w sieciach bezpiecznych*



Rys. 6. *Mba* i *Ma*, ale nie *Mab*



Rys. 7. *Mab* i *Mba*, ale {p}=*Mab*$\neq$*Mba*=$\emptyset$

Zauważmy ponadto, że języki powyższych sieci nie mogą być wygenerowane przez sieci elementarne. Jeśli jednak dopuścimy pętelki w sieciach elementarnych, to istnieje sieć elementarna generująca język sieci z Rys. 6, ale już język L=(($a \cup b$)*bc)*($a \cup b$)* sieci z Rys. 7 – nie, nawet jeśli dopuścimy pętelki. Więcej nawet – nie istnieje sieć markowana generująca taki język, ponieważ $abc \in L$, $ba \in L$, ale $bac \notin L$.

2.2.3 Rozszerzenia sieci markowanych

Rozszerzeniem klasy sieci markowanych nazywać będziemy każdą klasę sieci, w których dopuszczalne są klasyczne łuki wejściowe i wyjściowe (jak w sieciach markowanych) i ponadto jakieś dodatkowe typy łuków, specyficzne dla danej klasy.

Jednym z takich rozszerzeń są sieci z wagami.

Definicja 2.8. *Sieć markowana z wagami*

Siecią markowaną z wagami nazywamy czwórkę $N = (A, P, F, M_0)$, gdzie

- A i P są skończonymi zbiorami *akcji* i *warunków*;
- F: $A \times P \cup P \times A \to \mathbb{N}$ jest *funkcją przepływu*, opisującą wagę: ilość żetonów potrzebnych do wykonania danej akcji lub ilość żetonów umieszczanych w warunku po wykonaniu danej akcji;
- *stan początkowy* M_0 jest wielozbiorem nad P.

Oznaczenia. Wagi będą oznaczane na rysunkach liczbami naturalnymi obok odpowiednich łuków, brak liczby obok łuku oznacza wagę 1, brak łuku oznacza wagę 0. Podobnie jak w sieciach bez wag przyjmujemy oznaczenia zbiorów warunków wejściowych i wyjściowych dla ustalonej akcji $a \in A$:

$$\bullet a = \{p \in P \mid F(p,a) > 0\} \qquad a^\bullet = \{p \in P \mid F(a,p) > 0\} \qquad \bullet a^\bullet = \bullet a \cup a^\bullet$$

Działanie sieci markowanych z wagami. Niech $N = (A,P,F,M_0)$ będzie siecią markowaną z wagami. Mówimy, że akcja $a \in A$ jest *umożliwiona* w stanie $M \in \mathbb{N}^P$ wtedy i tylko wtedy, gdy $(\forall p \in \bullet a)\ M(p) \geq F(p,a)$. *Wykonanie* w stanie M akcji umożliwionej zmienia stan sieci; nowym stanem jest stan M' określony następująco:

$$(\forall p \in P)\ M'(p) = M(p) - F(p,a) + F(a,p).$$

Dla każdej sieci z wagami można skonstruować sieć bez wag symulującą jej zachowanie (Starke [44]). To rozszerzenie jest jednak użyteczne i wygodne dla zastosowań.

Klasami sieci istotnie poszerzającymi możliwości sieci markowanych są między innymi sieci czyszczące (*reset nets*), inhibitorowe (*inhibitor nets*) i przerzucające (*transfer nets*).

Definicja 2.9. *Sieci czyszczące, inhibitorowe, przerzucające*

Siecią czyszczącą (odpowiednio *inhibitorową, przerzucającą*) nazywamy czwórkę $N = (A, P, F, M_0)$, gdzie

- A i P są skończonymi zbiorami *akcji* i *warunków*;
- *stan początkowy* M_0 jest wielozbiorem nad P;
- F jest funkcją przepływu:
  - dla sieci czyszczących
    F: $A \times P \cup P \times A \to \mathbb{N} \cup \{\text{reset}\}$ oraz $F(P,A) \subseteq \mathbb{N}$;
  - dla sieci inhibitorowych
    F: $A \times P \cup P \times A \to \mathbb{N} \cup \{\text{inhib}\}$ oraz $F(A,P) \subseteq \mathbb{N}$;

– dla sieci przerzucających
$F: A\times P\cup P\times A \to \mathbb{N} \cup P$ oraz $(\forall p\in P)(\forall a\in A)\ F(p,a) \subseteq \mathbb{N}\cup\{p\}$ i
$(\forall p\in P)(\forall a\in A)\ (\ F(p,a)=p \Leftrightarrow (\exists q\in P)\ F(a,q)=p\ )$.

Z przyczyn technicznych przyjmujemy, że każde dwa węzły sieci są połączone najwyżej jednym łukiem, czyli $(\forall p\in P)(\forall a\in A)\ F(p,a)=0 \vee F(a,p)=0$.

Oznaczenia:

$\bullet a=\{p\in P \mid F(p,a)\in \mathbb{N} \text{ i } F(p,a)>0\}$ $\qquad$ $a\bullet=\{p\in P \mid F(a,p)\in \mathbb{N} \text{ i } F(a,p)>0\}$

$\text{reset}(a)=\{p\in P \mid \text{F}(a,p)=\text{reset}\}$ $\qquad$ $\text{inhib}(a)=\{p\in P \mid \text{F}(p,a)=\text{inhib}\}$

Działanie sieci czyszczących. Definicja umożliwienia akcji a w stanie M jest taka sama jak dla sieci markowanych z wagami: $(\forall p\in \bullet a)\ M(p) \geq F(p,a)$. Działanie sieci jest nieco inne – łuk typu „reset" czyści warunek niezależnie od jego poprzedniej zawartości, zatem stan M' po wykonaniu akcji a w stanie M jest następujący:

$$M'(p) = \begin{cases} 0 & \text{jeśli } p\in \text{reset}(a) \\ M(p)-F(p,a)+F(a,p) & \text{w przeciwnym przypadku} \end{cases}$$

Działanie sieci inhibitorowych. Akcja a jest umożliwiona w stanie M, jeśli warunki wejściowe mają odpowiednią ilość żetonów, a inhibitorowe są puste: $(\forall p\in \bullet a)\ M(p) \geq F(p,a)$ i $(\forall p\in \text{inhib}(a))\ M(p)=0$. Działanie sieci jest takie samo jak dla sieci markowanych z wagami: $M'(p) = M(p)-F(p,a)+F(a,p)$.

Działanie sieci przerzucających. Akcja a jest umożliwiona w stanie M, jeśli $(\forall p\in \bullet a)\ M(p) \geq F(p,a)$. Działanie sieci jest nieco inne niż w sieciach markowanych – łuk przerzucający przenosi zawartość jednego warunku do innego, zatem stan M' po wykonaniu akcji a w stanie M jest następujący:

$$M'(p) = \begin{cases} 0 & \text{jeśli } F(p,a)=p \\ M(p) + M(q) & \text{jeśli } F(a,p)=q \\ M(p)-F(p,a)+F(a,p) & \text{w przeciwnym przypadku} \end{cases}$$

Sieci markowane i wymienione wyżej rozszerzenia można wyrazić w ramach ogólnego modelu tzw. sieci samomodyfikujących (*self-modifying nets*), zaproponowanych przez Valka [45], badanych i uogólnionych w [15]. Są to sieci ze zmiennymi wagami, zależnymi od aktualnych zawartości wszystkich warunków. Nie przytaczamy ich formalnych definicji, gdyż nie będą one rozważane w rozprawie.

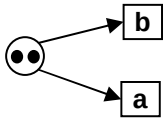
W dalszej części pracy ważną rolę odgrywać będzie własność diamentu. Wiemy, że sieci elementarne i bezpętelkowe markowane mają silną własność diamentu (Fakt 2.4), a więc też słabą. Łatwo pokazać, że dowolne sieci markowane i sieci inhibitorowe mają słabą własność diamentu. Pozostałe rozważane w rozprawie rozszerzenia (sieci bezpieczne, czyszczące, przerzucające) nie mają własności diamentu.

2.2.4 Konflikty w sieciach Petriego

Badając sieci Petriego, rozważa się zarówno ich własności statyczne (związane ze strukturą sieci), jak i własności dynamiczne (związane z zachowaniem sieci). Te ostatnie będą nas szczególnie interesować.

Jedną z wielu sytuacji dynamicznych, występujących w trakcie działania sieci, jest konflikt. W sieciach elementarnych [42] definiuje się konflikt jako stan, w którym dwie akcje są umożliwione i ubiegają się o wspólny zasób, tzn. mają wspólny warunek wejściowy lub wyjściowy. W sieciach elementarnych jeśli akcje mają wspólny warunek (obojętnie – wejściowy lub wyjściowy), to wykonanie jednej powoduje uniemożliwienie drugiej, ta definicja jest więc zupełnie odpowiednia. W sieciach markowanych sytuacja jest już nieco bardziej złożona. Rozważmy następującą sieć:

Przykład 2.10. *Wspólny warunek a niezależność*



Rys. 8. Sieć markowana, w której dwie akcje mają wspólny warunek, jednak mogą się wykonać niezależnie

Rzeczywiście, obie akcje ubiegają się o ten sam zasób, jednak ten zasób jest na tyle duży, że akcje mogą się wykonać bezkonfliktowo: dla stanu M z Rys. 8 zachodzi Mab i także Mba oraz oba stany wynikowe są równe. Taką sytuację uznajemy za bezkonfliktową.

Widzieliśmy już (Przykład 2.7), że w sieciach bezpiecznych zachodzą zjawiska nie spotykane w sieciach elementarnych i markowanych. Oba przypadki (Rys. 6 i 7) uznajemy za konfliktowe, gdyż kolejność wykonania akcji wpływa na stan całej sieci.

Wymienione wyżej warunki składają się na następującą ogólną definicję konfliktu, obowiązującą w dowolnej klasie sieci Petriego (i ogólniej – w dowolnym systemie tranzycyjnym):

Definicja 2.11. *Konflikt, stan konfliktowy, konfliktowe obliczenie*

Niech (A,P,F,M_0) będzie siecią (dowolnego typu).

- Akcje $a,b \in A$ są w *konflikcie* w stanie M (piszemy M[a#b]) jeśli
  (1) $a \neq b$ są różne,
  i (2) $Ma \wedge Mb$ obie umożliwione w stanie M,
  i (3) $\neg Mab \vee \neg Mba \vee Mab \neq Mba$ jeden z ciągów ab, ba nie jest umożliwiony w M lub oba prowadzą do różnych stanów.
- Stan M jest *stanem konfliktowym* jeśli $(\exists a,b \in A)$ M[a#b].
- Krok MaM' jest *krokiem konfliktowym* jeśli $(\exists b \in A)$ M[a#b].
- Obliczenie $M_0a_1M_1a_2M_2\ldots$ jest *konfliktowe* jeśli $(\exists i \geq 1)$ $(\exists b \in A)$ $M_{i-1}(a_i\#b)$, tzn. jeśli zawiera konfliktowy krok.

Stan (krok, obliczenie) nazywamy *bezkonfliktowym*, jeśli nie jest konfliktowe.

W rozdziale 3 zajmiemy się problemem zapewniania bezkonfliktowego działania sieci.

2.3 Ślady i języki śladów

Użytecznym narzędziem matematycznym do badania zachowania sieci Petriego z uwzględnieniem przemienności niektórych akcji (akcji niezależnych – zob. następny podrozdział) są monoidy śladów.

Monoidy wolne częściowo przemienne (*free partially commutative monoids*) po raz pierwszy pojawiły się w literaturze w roku 1969 jako narzędzie do badania przekształceń słów (Cartier/Foata [6]). Osiem lat później (Mazurkiewicz [32]) zostały zaproponowane jako matematyczne narzędzie do badania i opisu działania systemów współbieżnych, pod dziś używaną nazwą monoidy śladów (*trace monoids*).

Definicja 2.12. *Alfabet współbieżny*

Alfabet współbieżny (A,I) to alfabet skończony A z symetryczną i antyzwrotną *relacją niezależności* $\mathrm{I}\subseteq A\times A$. Uzupełnieniem relacji I jest symetryczna i zwrotna relacja $\mathrm{D}=A\times A-\mathrm{I}$, zwana *relacją zależności*. Relację tę rozszerzamy na słowa w następujący sposób: $(\forall u,w\in A^*)\ u\mathrm{I}w \Leftrightarrow \mathrm{Alph}(u)\times\mathrm{Alph}(v)\subseteq\mathrm{I}$, stąd $u\mathrm{D}w \Leftrightarrow \mathrm{Alph}(u)\times\mathrm{Alph}(v)\cap\mathrm{D}\neq\emptyset$.

Alfabet współbieżny przedstawiamy graficznie w postaci grafu zależności (A,D), w którym wierzchołkami są litery alfabetu, a krawędzie łączą litery zależne.

Definicja 2.13. *Relacja wyprowadzenia*

Niech $x, y\in A^*$. *Wyprowadzeniem* z x do y według relacji I (oznaczenie: $x \dashrightarrow y$) nazywamy ciąg $w_0,w_1,...,w_n$ słów z A^* taki, że $w_0=x$, $w_n=y$ oraz $(\forall i=1,...,n)\ (w_{i-1}=w_i \vee (\exists(a,b)\in \mathrm{I})\ (\exists u,v\in A^*)\ w_{i-1}=uabv \wedge w_i=ubav)$.

Zauważmy, że jeśli $x \dashrightarrow y$, to także $y \dashrightarrow x$, ponieważ relacja niezależności I jest symetryczna. Zatem możemy określić relację równoważności słów następująco:

Definicja 2.14. *Relacja równoważności*

Mówimy, że dwa słowa $x, y\in A^*$ są równoważne, jeśli jedno można wyprowadzić z drugiego: $x\approx y \ \Leftrightarrow\ x \dashrightarrow y \ \Leftrightarrow\ y \dashrightarrow x$.

Relacja $\approx$ jest kongruencją w monoidzie wolnym A^*.

Definicja 2.15. *Monoid śladów*

Monoidem śladów M(A,I) (lub A*/I) nazywamy monoid ilorazowy A*/≈. Elementy monoidu śladów nazywamy *śladami*, a podzbiory – *językami śladów*.

Zauważmy, że M(A,∅)=A*, czyli monoid wolny A* jest szczególnym przypadkiem monoidu śladów (w którym każde dwie litery z A są zależne).

Zgodnie z powyższą definicją, ślady to klasy abstrakcji relacji ≈; każda taka klasa zawiera skończoną liczbę elementów równej długości (dwa słowa należą do tej samej klasy abstrakcji, jeśli jedno powstaje z drugiego przez skończoną ilość przestawień sąsiadujących liter niezależnych). Można więc określić *długość śladu* $\alpha \in$ M(A,I) jako wspólną długość jego elementów: jeśli $\alpha=[w]$ dla pewnego $w \in A^*$, to $|\alpha|=|w|$. Elementy śladu będziemy nazywać *linearyzacjami* tego śladu.

W naturalny sposób możemy zdefiniować operacje, wiążące języki śladów z językami słów:

- Morfizm kanoniczny $[\cdot]: A^* \to$ M(A,I)
  przypisuje każdemu słowu $w \in A^*$ ślad $[w] \in$ M(A,I).
- Obraz kanoniczny $[\cdot]: 2^{A^*} \to 2^{\mathrm{M}(A,\mathrm{I})}$
  przypisuje każdemu językowi $L \subseteq A^*$ język śladów $[L]=\{[w] \mid w \in L\} \subseteq$ M(A,I).
- Spłaszczenie $\bigcup: 2^{\mathrm{M}(A,\mathrm{I})} \to 2^{A^*}$
  przypisuje każdemu językowi śladów $T \subseteq$ M(A,I) sumę mnogościową jego śladów $\bigcup T=\{w \in A^* \mid [w] \in T\}$.
- Domknięcie Cl: $2^{A^*} \to 2^{A^*}$
  przypisuje każdemu językowi $L \subseteq A^*$ zbiór wszystkich słów równoważnych jakiemuś słowu z L: $\mathrm{Cl}(L)=\bigcup[L]=\{w \in A^* \mid (\exists v \in L)\ w \approx v\}$.

Następujące związki między wyżej zdefiniowanymi operacjami są w większości oczywiste:

$\bigcup[L]=\mathrm{Cl}(L)$ $\quad [L]=[\mathrm{Cl}(L)]$ $\quad [\bigcup[L]]=[L]$ $\quad [L \cup L']=[L] \cup [L']$

$\mathrm{Cl}(L \cup L')=\mathrm{Cl}(L) \cup \mathrm{CL}(L')$ $\quad \mathrm{Cl}(\mathrm{Cl}(L))=\mathrm{Cl}(L)$ $\quad \mathrm{Cl}(\bigcup[L])=\bigcup[L]$

Ważną rolę w badaniu języków śladów odgrywa także porządek słownikowy. Pozwala on jednoznacznie reprezentować ślady za pomocą słów minimalnych.

Definicja 2.16. *Porządek słownikowy*

Niech A będzie skończonym alfabetem, uporządkowanym liniowo przez relację $<$. Porządek ten rozszerzamy do porządku liniowego $\leq$ na A*, zwanego *porządkiem słownikowym*, w sposób następujący:

$$(\forall u \in A^+)\ \varepsilon < u$$

$$(\forall u,v \in A^*)\ (\forall a,b \in A)\ au < bv \iff a<b \vee (a=b \wedge u<v)$$

$$(\forall u,v \in A^*)\ u \leq v \iff u<v \vee u=v$$

Definicja 2.17. *Minimalna reprezentacja języka*

Niech M=M(*A*,I) i niech $\alpha \in M$. Przez *Lex*(α) oznaczać będziemy najmniejszy (względem $\leq$) element śladu α:

$$Lex(\alpha) = u \;\Leftrightarrow\; u \in \alpha \wedge (\forall w \in \alpha)\; u \leq w$$

Lex(α) nazywać będziemy *minimalnym reprezentantem* śladu α. Jeśli $T \subseteq M(A,I)$ jest językiem śladów, to przez *Lex*(*T*) oznaczać będziemy *minimalną reprezentację* języka *T*, czyli język słów

$$Lex(T) = \{Lex(\alpha) \mid \alpha \in T\} = \bigcup T \cap Lex(M)$$

Definicja 2.18. *Rzutowanie*

Dla $X \subseteq A$ *rzutowaniem* na *X* nazywamy morfizm $\Pi_X : A^* \to X^*$ wycierający litery z $A-X$ i pozostawiający litery z *X*. Formalnie: $\Pi_X(a) = a$, jeśli $a \in X$ i $\Pi_X(a) = \varepsilon$, jeśli $a \notin X$.

Lemat 2.19. *O rzutowaniu*

Niech M=M(*A*,I) i niech $u,v \in A^*$. Słowa *u*,*v* są równoważne, jeśli ich rzuty na pary liter zależnych są jednakowe:

$$[u]=[v] \Leftrightarrow (\forall (a,b) \in D)\; \Pi_{a,b}(u)=\Pi_{a,b}(v)$$

Dowód: ($\Rightarrow$) Indukcja po długości wyprowadzenia. Wystarczy pokazać, że jeśli $u=xcdy$ i $v=xdcy$ dla *c*I*d*, to zachodzi teza lematu. Jeśli *a*D*b*, to możliwe są trzy przypadki:

(1) $c \notin \{a,b\} \wedge d \in \{a,b\}$ wtedy $\Pi_{a,b}(u)=\Pi_{a,b}(x)d\Pi_{a,b}(y)=\Pi_{a,b}(v)$

(2) $c \in \{a,b\} \wedge d \notin \{a,b\}$ wtedy $\Pi_{a,b}(u)=\Pi_{a,b}(x)c\Pi_{a,b}(y)=\Pi_{a,b}(v)$

(3) $c \notin \{a,b\} \wedge d \notin \{a,b\}$ wtedy $\Pi_{a,b}(u)=\Pi_{a,b}(x)\Pi_{a,b}(y)=\Pi_{a,b}(v)$

We wszystkich przypadkach $\Pi_{a,b}(u)=\Pi_{a,b}(v)$.

($\Leftarrow$) Najpierw zauważmy, że $(\forall a \in A)\; |u|_a=|v|_a$, bo inaczej byłoby $\Pi_a(u) \neq \Pi_a(v)$. Przypuśćmy teraz, że $[u] \neq [v]$, więc $u \neq v$ i przyjmijmy, że $u=Lex([u])$ i $v=Lex([v])$ są słowami minimalnymi (słownikowo) śladów [*u*] i [*v*]. Zatem $u=xay$, $v=xbz$, dla pewnych $a \neq b$. Przyjmijmy, że $a<b$ w alfabecie. Jeśli *a*D*b* to $\Pi_{a,b}(u) \neq \Pi_{a,b}(v)$ – sprzeczność, więc *a*I*b*. Jeśli tak, to $v=xby'ay''$. Gdyby *a*Iy' (tzn. każda litera słowa y' jest niezależna z *a*), to $xaby'y'' \approx v$ i $xaby'y'' < v$, co niemożliwe, bo $v=Lex([v])$. Istnieje więc *c* w podsłowie y', takie że *c*D*a*. Ale wtedy $\Pi_{a,c}(u)=\Pi_{a,c}(x)a\Pi_{a,c}(y)$ i $\Pi_{a,c}(v)=\Pi_{a,c}(x)c\Pi_{a,c}(z)$, więc $\Pi_{a,c}(u) \neq \Pi_{a,c}(v)$ – sprzeczność. Zatem $[u]=[v]$. ¤

Lemat 2.19 pokazali Cori/Perrin [8]. Powyższy dowód, wykorzystujący minimalne reprezentacje śladów, jest jednak dużo prostszy.

Graf śladu jest pojęciem pomocniczym, ale bardzo użytecznym i wygodnym. Nawiązując do interpretacji śladów w ramach teorii współbieżności (grafy śladów reprezentują procesy systemów współbieżnych), pomagają one „zobaczyć" ślady jako częściowo uporządkowany zbiory wystąpień akcji, gdzie częściowy porządek reprezentuje związki przyczynowo-skutkowe, jak również następstwo czasowe między poszczególnymi akcjami.

Definicja 2.20. *Graf śladu*

Niech (A,I) będzie alfabetem współbieżnym, i niech $w=a_1...a_n \in A^*$. Grafem słowa w względem relacji niezależności I nazywamy graf $G_I(w)=(V,\rightarrow,\lambda)$ z etykietowanymi wierzchołkami, gdzie:

$V=\{x_1,...,x_n\}$ – zbiór wierzchołków;

$x_i \rightarrow x_j \Leftrightarrow i<j \wedge a_i D a_j$ – zbiór krawędzi;

$\lambda: V \rightarrow A$ – funkcja etykietująca: $\lambda(x_i)=a_i$ dla i=1,...,n.

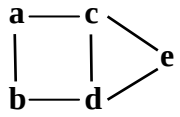
Dwa grafy etykietowane nazywamy izomorficznymi, jeśli istnieje ich izomorfizm zachowujący etykietowanie. Można pokazać, że jeśli $[u]=[v]$, to $G(u)$ i $G(v)$ są izomorficzne. Wobec tego możemy określić (z dokładnością do izomorfizmu) *graf śladu* α jako graf dowolnego jego reprezentanta, czyli dowolnego $w \in \alpha$. Podobnie można pokazać, że jeśli $[u]\neq[v]$, to $G(u)$ i $G(v)$ nie są izomorficzne. Zatem grafy śladów reprezentują ślady jednoznacznie. Grafom śladów poświęcony jest rozdział „Dependence graphs" (Hoogeboom/Rozenberg [22]) w książce [13].

Formalnie, rysując graf śladu, powinniśmy rysować wszystkie jego krawędzie. Wobec faktu, że $\rightarrow^*$ jest częściowym porządkiem, możemy rysować tylko jego szkielet, czyli podgraf $\rightarrow - \rightarrow\rightarrow^+$ jego krawędzi nierozkładalnych.

Przykład 2.21.

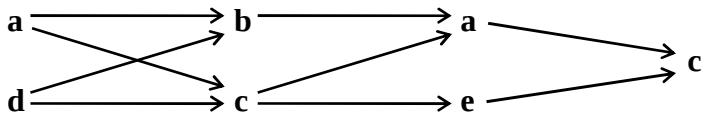
$A=\{a,b,c,d,e\}$ $I=\{(a,e),(a,d),(b,c),(b,e)\}$ $D=\{(a,c),(a,b),(b,d),(c,d),(c,e),(d,e)\}$

Graf zależności:



Rozważmy ślad [*dabcaec*]. Zbiór jego wszystkich linearyzacji to {*dabcaec*, *adbcaec*, *dacbaec*, *adcbaec*, *dabceac*, *adbceac*, *dacbeac*, *adcbeac*, *adcebac*, *dacebac*}.

Graf tego śladu wygląda następująco:



Rys. 9. Graf śladu [*dabcaec*]

Ślady nieskończone

Aby zdefiniować ślady nieskończone analogicznie jak w przypadku skończonym – jako elementy pewnego monoidu ilorazowego – należałoby zdefiniować działanie w takim monoidzie, czyli konkatenację nieskończonych ciągów. Takie próby są podejmowane ([12,19,27,35]), tu jednak chcemy tego uniknąć. Ograniczymy się do zdefiniowania relacji równoważności dwóch ciągów nieskończonych – klasy abstrakcji tej relacji

nazwiemy śladami. Definicja ta będzie zgodna z wcześniejszą definicją śladów skończonych.

Zauważmy, że relacja niezależności indukuje relację równoważności na A^* (Definicja 2.14). Ta relacja na skończonych słowach może być rozszerzona na słowa nieskończone $x, y \in A^\omega$ w następujący sposób (Best/Devillers [4]).

Definicja 2.22. *Ślady nieskończone*
Nieskończone obliczenia $x, y \in A^\omega$ są równoważne wtedy i tylko wtedy, gdy dla każdego skończonego prefiksu u słowa x istnieje skończony prefiks v słowa y taki, że $v \approx uw$ dla pewnego $w \in A^*$, i odwrotnie:

$$x \approx y \;\Leftrightarrow\; (\forall u \leq^{\text{fin}} x)\,(\exists\, v \leq^{\text{fin}} y)\,(\exists w \in A^*)\; v \approx uw \wedge (\forall u \leq^{\text{fin}} y)\,(\exists\, v \leq^{\text{fin}} x)\,(\exists w \in A^*)\; v \approx uw$$

Zauważmy, że jest to relacja równoważności. Możemy więc zdefiniować *nieskończone ślady* jako klasy abstrakcji tej relacji. Zbiór wszystkich skończonych A^*/I i nieskończonych A^ω/I śladów nad alfabetem współbieżnym (A,I) będzie oznaczany przez A^∞/I.

Rozważać będziemy jedynie konkatenację śladu skończonego $\alpha=[u] \in A^*/I$ z dowolnym $\beta=[v] \in A^\infty/I$, zdefiniowaną w naturalny sposób: $\alpha\beta=[uv]$. Dla śladów α, β zapis $\beta \leq^{\text{fin}} \alpha$ będzie oznaczał, że β jest skończony i $\alpha=\beta\gamma$ dla pewnego śladu γ (może być nieskończony), taki ślad β będzie nazywany *prefiksem* śladu α, a γ – *ogonem*. Ilość wystąpień litery $a \in A$ w śladzie $\alpha \in A^\infty/I$ będzie oznaczany przez $|\alpha|_a$. Oczywiście, jeśli $\alpha=[w]$, to $|\alpha|_a=|w|_a$.

Definicję 2.22 (prefiksową) zaproponowali Best/Devillers [4] (warunek (2) w Twierdzeniu 2.23 poniżej jest innym jej sformułowaniem). Ślady nieskończone można też zdefiniować przy pomocy rzutów na kliki liter zależnych (Flé/Roucairol [16], warunek (1) w Twierdzeniu 2.23). Gastin/Petit [20] wzmiankują o równoważności tych dwóch definicji, jednak bez dowodu. Niżej przedstawimy pełny dowód tego faktu.

Przypomnijmy, że *domknięcie śladowe* języka $L \subseteq A^*$ (względem relacji niezależności I) to zbiór słów

$$\mathrm{Cl}(L) = \bigcup[L] = \{w \in A^* \mid (\exists u \in L)\; u \dashrightarrow w\},$$

czyli zbiór wszystkich słów wyprowadzalnych ze słów języka L przez przestawianie sąsiednich liter niezależnych.

Niech $w \in A^\infty$. Przypomnijmy, że $\mathrm{Pr}(w) = \{u \in A^* \mid (\exists v \in A^\infty)\; uv=w\}$ oznacza zbiór wszystkich skończonych prefiksów słowa w, a $\mathrm{Pr}(L) = \bigcup\{\mathrm{Pr}(w) \mid w \in L\}$ jest domknięciem prefiksowym języka $L \subseteq A^*$, czyli zbiorem wszystkich prefiksów elementów języka L.

Niech $u,v \in A^*$. Będziemy pisać $\mathrm{Oc}(u) \subseteq \mathrm{Oc}(v)$, jeśli dla każdego $a \in A^*$ zachodzi nierówność $|u|_a \leq |v|_a$, czyli liczby wystąpień poszczególnych liter w słowie u są nie większe niż w słowie v. Zapis $a \in \mathrm{Oc}(v)$, dla $a \in A$, $v \in A^*$, oznaczać będzie, że $|v|_a \geq 1$, czyli że litera a występuje w słowie v.

Twierdzenie 2.23. *Równoważność dwu definicji śladów nieskończonych*
Niech $u,v\in A^{\omega}$. Następujące zdania są równoważne:

(1) $(\forall a,b\in A,\ a\mathrm{D}b)\ \Pi_{a,b}(u) = \Pi_{a,b}(v)$ *definicja rzutowa*

(2) $\mathrm{Pr}(u) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$ i $\mathrm{Pr}(v) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$ *definicja prefiksowa*

(3) $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u))) = \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$

Dowód.

(1) $\Rightarrow$ (2): Niech u_n oznacza prefiks u długości n. Pokażemy indukcyjnie, że $(\forall n)\ u_n\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$, czyli $\mathrm{Pr}(u) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$.

Oczywiście $u_0=\varepsilon\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$. Założenie indukcyjne: $u_n\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$. Niech $u_{n+1}=u_n a$; weźmy najkrótszy prefiks $w\in \mathrm{Pr}(v)$ taki, że $\mathrm{Oc}(u_{n+1})\subseteq\mathrm{Oc}(w)$. Taki prefiks istnieje, bo z (1) mamy $(\forall a\in A)\ \Pi_a(u) = \Pi_a(v)$, czyli liczby wystąpień poszczególnych liter w słowach u i v są równe. W połączeniu z założeniem indukcyjnym mamy

$$w \dashrightarrow u_n xay;\ \text{dla pewnych } x,y\in A^*,\ a\notin \mathrm{Oc}(x).$$

Zauważmy, że $x\mathrm{I}a$, bo jeśli nie, to $a\mathrm{D}b$ dla jakiegoś $b\in \mathrm{Oc}(x)$, ale wtedy byłoby $\Pi_{a,b}(u) = \Pi_{a,b}(u_n)\,a\ldots \neq \Pi_{a,b}(v) = \Pi_{a,b}(w)\ldots = \Pi_{a,b}(u_n)\,b\ldots$, co przeczy (1). (Korzystamy tu z Lematu 2.19 o rzutowaniu dla słów skończonych.) Zatem $w \dashrightarrow u_n xay \dashrightarrow u_n axy$, więc $u_{n+1}=u_n a\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$, bo $w\in \mathrm{Pr}(v)$.

Analogicznie $(\forall n)\ v_n\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$, czyli $\mathrm{Pr}(v) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$.

(2) $\Rightarrow$ (3): Najpierw pomocniczy Postulat: $(\forall L\subseteq A^*)\ \mathrm{Cl}(\mathrm{Pr}(\mathrm{Cl}(L))) = \mathrm{Pr}(\mathrm{Cl}(L))$.

Dowód Postulatu: $\supseteq$ oczywiste; $\subseteq$: jeśli $w\in \mathrm{Cl}(\mathrm{Pr}(\mathrm{Cl}(L)))$ to

$(\exists x\in L)\ x \dashrightarrow yz \wedge y \dashrightarrow w$, więc $x \dashrightarrow wz$, więc $wz\in \mathrm{Cl}(L)$, więc $w\in \mathrm{Pr}(\mathrm{Cl}(L))$.

Teraz mamy: Jeśli $\mathrm{Pr}(u) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v))$ to (z Postulatu) $\mathrm{Cl}(\mathrm{Pr}(u)) \subseteq \mathrm{Cl}(\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))) = \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$, zatem $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u))) \subseteq \mathrm{Pr}(\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))) = \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$. Analogicznie $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v))) \subseteq \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$. Zatem $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u))) = \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$.

(3) $\Rightarrow$ (1): Zauważmy, że $(\forall a\in A)\ |u|_a = |v|_a$. Gdyby nie, to $|u|_a < |v|_a$ (lub odwrotnie) dla jakiegoś $a\in A$, a wtedy takie $w\in \mathrm{Pr}(v)$, że $|w|_a = |u|_a +1$, należy do $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$ i nie należy do $\mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$. Sprzeczność z (3). Zatem $(\forall a\in A)\ \Pi_a(u)=\Pi_a(v)$.

Jeśli $\Pi_{a,b}(u)\neq\Pi_{a,b}(v)$ dla $a\neq b$, to różnią się one na jakiejś pozycji (bo ich długości są równe), czyli $\Pi_{a,b}(u)=wa\ldots$ i $\Pi_{a,b}(v)=wb\ldots$ dla pewnego $w\in\{a,b\}^*$. Weźmy najkrótszy prefiks $x\in \mathrm{Pr}(u)$, taki że $\Pi_{a,b}(x)=wa$. Mamy $x\in \mathrm{Pr}(u)$, więc $x\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(u)))$, ale $x\notin \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$ [bo $a\mathrm{D}b$, więc dla każdego $y\in \mathrm{Pr}(\mathrm{Cl}(\mathrm{Pr}(v)))$ jego rzut $\Pi_{a,b}(y)$ jest prefiksem $\Pi_{a,b}(v)=wb\ldots$, więc jest $\neq wa=\Pi_{a,b}(x)$, więc $y\neq x$]. Sprzeczność z (3). ¤

2.4 Niezależność akcji w systemach tranzycyjnych

Możemy teraz popatrzeć na sieci Petriego (czy inne systemy współbieżne) jak na systemy generujące języki śladów, w których relacja niezależności nie jest dana z góry, lecz zależy od samej sieci – od jej struktury lub zachowania. W tym podrozdziale omówimy definicję relacji niezależności akcji sieci Petriego (definicję strukturalną), przytoczymy argumenty za jej niewystarczalnością w niektórych sytuacjach i podamy

definicję opartą na zachowaniu sieci (definicję językową), która abstrahuje od rodzaju urządzenia generującego język i będzie używana w następnych rozdziałach pracy. Takie podejście bliskie jest koncepcji Starka [43]. Tu jednak system tranzycyjny indukuje swoją relację niezależności, podczas gdy w [43] relacja taka jest z góry zadana.

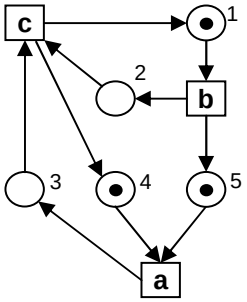
Idea stojąca za uznaniem dwóch akcji za niezależne jest następująca: dwie akcje są niezależne, jeśli mogą się wykonać niezależnie od siebie, współbieżnie, jeśli nie wpływają na siebie, nie blokują się wzajemnie. Dla sieci elementarnych niezależność została zdefiniowana strukturalnie (Mazurkiewicz [32]) – dwie akcje są niezależne, jeśli nie mają wspólnych warunków (wejściowych lub wyjściowych):

$$a\,\mathrm{I}\,b \Leftrightarrow a\neq b \wedge {}^\bullet a^\bullet \cap {}^\bullet b^\bullet = \varnothing$$

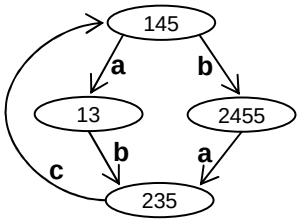
Ta definicja jest zupełnie wystarczająca dla sieci elementarnych. Jednak już przy sieciach markowanych definicja strukturalna gubi pewną informację o możliwości współbieżnego wykonania akcji. Rozważmy następujący przykład:

Przykład 2.24. *Niezależność strukturalna a niezależność zachowaniowa*

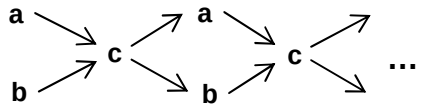
Sieć markowana:



Graf osiągalności:



Graf śladu $[(abc)^\omega]$:



Rys. 10. Akcje *a* i *b* są strukturalnie zależne, ale zachowaniowo niezależne

W powyższej sieci akcje *a* i *b* mają wspólny warunek (5), są więc strukturalnie zależne. Zauważmy jednak, że w obliczeniu $(abc)^\omega$ akcje *a* i *b* mogą się zamieniać miejscami – wszędzie tam, gdzie jest umożliwione *ab*, jest też umożliwione *ba*. Są one więc zachowaniowo niezależne.

Te rozważania prowadzą więc do bardziej ogólnej, zorientowanej zachowaniowo (językowo) definicji relacji niezależności.

Definicja 2.25. *Niezależność indukowana przez język*
Niech $L\subseteq A^*$ będzie językiem prefiksowo domkniętym. Relacja niezależności I_L indukowana przez *L* jest określona następująco:

$$a\,\mathrm{I}_L\,b \text{ wtw } (\forall u,v\in A^*)\ uabv\in L \Leftrightarrow ubav\in L.$$

Niezależność językowa jest maksymalną niezależnością, względem której *L* jest domknięty. Wtedy język *L* może być traktowany jako zbiór śladów – język śladów [*L*].

Niech $L^{\infty}=L\cup L^{\omega}$ będzie zbiorem wszystkich skończonych i nieskończonych obliczeń indukowanych przez L, jak to zdefiniowaliśmy w podrozdziale 2.1. Można pokazać, że także L^{∞} jest domknięty względem I, więc w pełni reprezentuje język $[L^{\infty}]$ – zbiór wszystkich skończonych i nieskończonych współbieżnych procesów.

Do badania zachowania systemów tranzycyjnych wygodnie będzie wprowadzić jeszcze jedną definicję niezależności akcji. Związana ona będzie ze stanami danego systemu i nazywana niezależnością systemową. Pokażemy, że te dwie niezależności – językowa i systemowa – są równe w przypadku sieci elementarnych i sieci markowanych, co pozwoli wygodnie znajdować i badać niezależność językową przy pomocy grafów osiągalności.

Definicja 2.26. *Niezależność indukowana przez system*
Niech $S=(A,Q,q_0)$ będzie dowolnym systemem tranzycyjnym. Przypomnijmy, że zapis $q(ab)=q(ba)$ oznacza, że $q(ab)$ i $q(ba)$ są określone i równe, lub że oba są nieokreślone. Akcje $a,b\in A$ są:

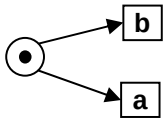
- *lokalnie niezależne* w stanie $q\in Q$ (ozn. $a\mathrm{I}_q b$) wtw $q(ab)=q(ba)$;
- *lokalnie zależne* w stanie $q\in Q$ (ozn. $a\mathrm{D}_q b$) wtw $q(ab)\neq q(ba)$;
- *lokalnie współbieżne* w stanie $q\in Q$ (ozn. $a||_q b$) wtw $(\exists q'\in Q)\ q(ab)=q(ba)=q'$;
- *niezależne* w systemie S (ozn. $a\mathrm{I}_S b$) wtw $(\forall q\in Q)\ a\mathrm{I}_q b$;
- *zależne* w systemie S (ozn. $a\mathrm{D}_S b$) wtw $(\exists q\in Q)\ a\mathrm{D}_q b$.

W przypadku, gdy systemem tranzycyjnym jest graf osiągalności sieci Petriego, otrzymujemy następującą wersję powyższych definicji (M oznacza stan sieci Petriego – podzbiór warunków lub wielozbiór nad zbiorem warunków, zależnie od rodzaju sieci). Akcje $a,b\in A$ są:

- *lokalnie niezależne* w stanie M (ozn. $a\mathrm{I}_M b$) wtw
  $(MabM' \wedge MbaM') \vee (\neg Mab \wedge \neg Mba)$
- *lokalnie zależne* w stanie M (ozn. $a\mathrm{D}_M b$) wtw
  $(Mab \wedge \neg Mba) \vee (\neg Mab \wedge Mba) \vee (MabM' \wedge MbaM'' \wedge M'\neq M'')$
- *lokalnie współbieżne* w stanie M (ozn. $a||_M b$) wtw
  $MabM' \wedge MbaM'$
- *niezależne* w sieci N (ozn. $a\mathrm{I}_N b$) wtw $(\forall M\in \mathrm{RS(N)})\ a\mathrm{I}_M b$
- *zależne* w sieci N (ozn. $a\mathrm{D}_N b$) wtw $(\exists M\in \mathrm{RS(N)})\ a\mathrm{D}_M b$

Zauważmy, że jeśli dwie akcje są zależne i umożliwione w jakimś stanie, to są w konflikcie, natomiast nie odwrotnie: w poniższym przykładzie akcje a, b są w konflikcie, ale nie są zależne.

Przykład 2.27. *Konflikt a zależność*



Rys. 11. Akcje skonfliktowane, ale niezależne

Teraz rozważymy, jakie są związki pomiędzy niezależnością językową a systemową.

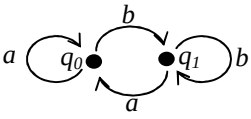
Dla każdego języka prefiksowo domkniętego można skonstruować jego system kanoniczny – minimalny automat generujący go. Wówczas niezależność tego języka i niezależność określona przez jego system kanoniczny są sobie równe:

Fakt 2.28. Niezależność indukowana przez prefiksowo domknięty język L i niezależność indukowana przez system kanoniczny tego języka S_L są równe: $I_L=I_{S_L}$.

Dowód. Uwzględniając, że stanami systemów kanonicznych są ilorazy lewostronne $q_u=L/u$, mamy: $aI_Lb \Leftrightarrow (\forall u)\ L/uab=L/uba \Leftrightarrow (\forall u)\ (L/u)/ab=(L/u)/ba \Leftrightarrow (\forall u)\ q_u(ab)=q_u(ba) \Leftrightarrow (\forall q)\ q(ab)=q(ba)$. ¤

Jednakże równość taka nie zachodzi dla dowolnego systemu S generującego język L, czasami niezależność systemowa jest mniejsza:

Przykład 2.29. *Niezależność systemowa a niezależność językowa*

$L=(a\cup b)^*$ S_L:  S generujący L: 

Rys. 12. Mamy $I_L=I_{S_L}=\{(a,b)\}$ i $I_S=\varnothing$ (bo $q_0(ab)=q_1\neq q_0=q_0(ba)$)

Nie może się jednak zdarzyć, by niezależność systemowa była większa:

Fakt 2.30. Dla dowolnego systemu tranzycyjnego $S=(A,Q,q_0)$ zachodzi inkluzja $I_S\subseteq I_{L(S)}$.
Dowód. Jeśli aI_Sb to $uabv\in L(S) \Leftrightarrow (q_0)u(q)ab(q')v(q'') \Leftrightarrow (q_0)u(q)ba(q')v(q'') \Leftrightarrow ubav\in L(S)$. Zatem $aI_{L(S)}b$. ¤

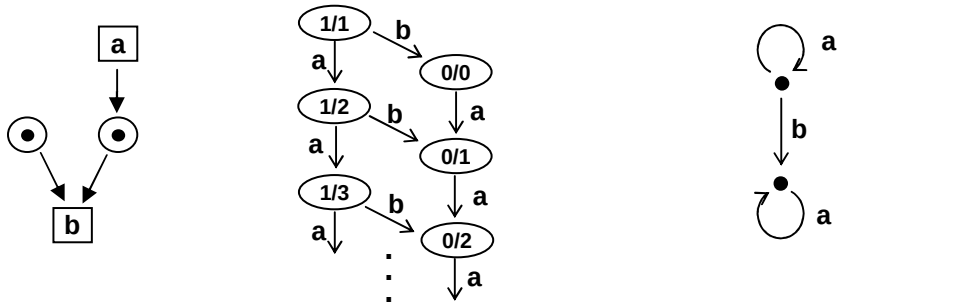
Zauważmy, że system S z Przykładu 2.29 nie miał własności diamentu. Pokażemy, że w deterministycznych systemach z własnością diamentu równość niezależności systemowej i językowej zawsze zachodzi.

Fakt 2.31. Niech $S=(A,Q,q_0)$ będzie deterministycznym systemem z własnością diamentu generującym język $L=L(S)$. Wtedy niezależność indukowana przez system jest równa niezależności indukowanej przez język tego systemu: $I_S=I_{L(S)}$.

Dowód. ($\subseteq$): Fakt 2.30. ($\supseteq$): Przypuśćmy, że aI_Lb i aD_Sb. Zatem istnieje stan $q\in Q$ taki, że $q(ab)$ jest zdefiniowane i $q(ba)$ jest niezdefiniowane (albo odwrotnie), ponieważ S ma własność diamentu. Zatem istnieje $u\in L$ i stan $q'\in Q$ takie, że $(q_0)u(q)ab(q')$. Ale wtedy $uba\in L$, ponieważ aI_Lb, a zatem $(q_0)u(q)ba(q')$, ponieważ S jest deterministyczny i diamentowy. Stąd $q(ba)$ jest zdefiniowane – sprzeczność. ¤

Zauważmy, że graf osiągalności sieci Petriego jest przeważnie inny niż system kanoniczny generujący język tej sieci.

Przykład 2.32. *Graf osiągalności a system kanoniczny języka*



Sieć markowana N Graf osiągalności sieci N System kanoniczny języka L(N)

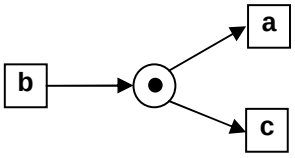
Rys. 13. Graf osiągalności sieci N jest nieskończony,
a system kanoniczny generujący język tej sieci L(N)=Pr(a^*ba^*) – skończony

Tym niemniej, niezależność systemowa (Definicja 2.26 – systemem jest dla sieci graf osiągalności) i językowa (Definicja 2.25 lub 2.26 – w tym przypadku systemem jest dla sieci system kanoniczny generujący język sieci) są równe w przypadku sieci elementarnych, markowanych i inhibitorowych. Wynika to bezpośrednio z Faktu 2.31 i własności diamentu w tych sieciach.

Natomiast sieci bezpieczne, czyszczące i przerzucające nie mają własności diamentu, w związku z czym dwie zdefiniowane wyżej relacje niezależności mogą być różne. W dalszej części pracy, jeśli nie powiedziano wyraźnie, o którą niezależność chodzi, będziemy rozważać niezależność językową.

Wróćmy jeszcze do oryginalnej definicji niezależności Mazurkiewicza dla sieci elementarnych: $aIb \Leftrightarrow (^\bullet a \cup a^\bullet) \cap (^\bullet b \cup b^\bullet) = \varnothing$. Nasza niezależność jest czasami większa (np. aIc w sieci elementarnej z Przykładu 2.33 poniżej), ale nigdy nie zmienia zachowania śladowego. Jest tak dlatego, że jako niezależne określamy też takie pary a, b, które nigdy nie występują obok siebie w obliczeniu (nie wykonują się współbieżnie). Każde sąsiadujące (w obliczeniu) akcje niezależne w naszym sensie są też niezależne w znaczeniu oryginalnym.

Przykład 2.33. *Niezależność językowa a niezależność strukturalna*



Rys. 14. Sieć elementarna, w której aIc, ale ac ani ca nie są nigdy umożliwione

W rozdziale 4 omówimy problemy rozstrzygalności związane z wyznaczaniem niezależności językowej, a następnie w rozdziale 5 będziemy badać etykę procesów, zdefiniowanych jako ślady, opartą również na tej definicji.

2.5 Etyka obliczeń

Pojęcia obliczeń sprawiedliwych i uczciwych zostały zdefiniowane w pracy Lehman/Pnueli/Stavi [29], pojęcie obliczeń superuczciwych zaproponował Best [3] (pod nazwą ∞-fair). Sformułujemy teraz definicje obliczeń sprawiedliwych, uczciwych i superuczciwych w dowolnym systemie tranzycyjnym.

Definicja 2.34. *Obliczenia sprawiedliwe, uczciwe i superuczciwe*

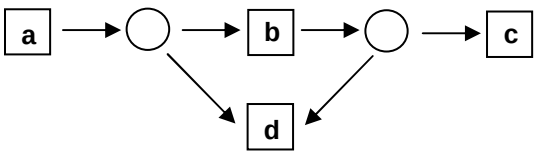
Niech S=(A, Q, q_0) będzie systemem tranzycyjnym, $u=a_1a_2\ldots\in L^{\omega}(S)$ – nieskończonym obliczeniem tego systemu, zaś $a\in A$ – jedną z jego akcji. Wtedy

- akcja a jest
  - *stale umożliwiona* w u $\Leftrightarrow$ ($\exists$i) ($\forall$k≥i) akcja a jest umożliwiona w stanie $q_0(a_1a_2\ldots a_k)$;
  - *nieskończenie często umożliwiona* w u $\Leftrightarrow$ zbiór {i; akcja a jest umożliwiona w stanie $q_0(a_1a_2\ldots a_i)$ } jest nieskończony;
  - *żywa* w u $\Leftrightarrow$ ($\forall$i) ($\exists w\in A^*$) $q_0(a_1a_2\ldots a_i)(w)$ jest określone i akcja a jest umożliwiona w stanie $q_0(a_1a_2\ldots a_iw)$;
- obliczenie u jest
  - *sprawiedliwe* $\Leftrightarrow$ ($\forall a\in A$) jeśli a stale umożliwiona w u, to $|u|_a=\omega$;
  - *uczciwe* $\Leftrightarrow$ ($\forall a\in A$) jeśli a nieskończenie często umożliwiona w u, to $|u|_a=\omega$;
  - *superuczciwe* $\Leftrightarrow$ ($\forall a\in A$) jeśli a żywa w u, to $|u|_a=\omega$.

Dla skończonych obliczeń $u\in L(S)$ wszystkie trzy pojęcia definiujemy jednakowo:
- u jest sprawiedliwe $\Leftrightarrow$ u jest uczciwe $\Leftrightarrow$ u jest superuczciwe $\Leftrightarrow$ u jest nierozszerzalne w $L(S)$.

Wprost z definicji wynika, że każde obliczenie superuczciwe jest uczciwe, a każde uczciwe jest sprawiedliwe. Poniższy przykład pokazuje, że oba zawierania są ostre:

Przykład 2.35. *Sieć elementarna*



Rys. 15. Obliczenie $(abc)^{\omega}$ jest uczciwe, ale nie superuczciwe, obliczenie $ab(acb)^{\omega}$ jest sprawiedliwe, ale nie uczciwe

W sieci z Przykładu 2.35 akcja d (nie występująca w obliczeniu $(abc)^{\omega}$) nie jest w nim w ogóle umożliwiona – więc obliczenie jest uczciwe, ale nie jest superuczciwe, bo akcja

d jest w nim żywa: umożliwiona w każdym stanie $M_0((abc)^n ab)(a)$. W obliczeniu $ab(acb)^\omega$ akcja d jest nieskończenie często umożliwiona – w każdym stanie $M_0(ab(acb)^n a)$, a nie wykonana, więc obliczenie nie jest uczciwe, ale jest sprawiedliwe, bo d nie jest stale umożliwione w $ab(acb)^\omega$.

Fakt 2.36. Niech S będzie dowolnym systemem tranzycyjnym. Każde skończone obliczenie $w \in L(S)$ można przedłużyć do obliczenia uczciwego, tzn. istnieje takie $v \in A^\infty$, że $wv \in L^\infty(S)$ i wv jest uczciwe.

Dowód. Uporządkujmy litery alfabetu: $a_1 < a_2 < \ldots < a_n$. Niech X_1 będzie zbiorem liter umożliwionych po w. Jeśli X_1 jest pusty, to w jest nierozszerzalne, czyli jest uczciwe. Jeśli nie, to rozszerzamy w o najmniejszą literę ze zbioru X_1 otrzymując nowe obliczenie wx_1, a następnie zmieniamy uporządkowanie liter, przesuwając x_1 na koniec (x_1 jest teraz większa od każdej z pozostałych liter). Postępujemy tak samo dla obliczenia wx_1 – ze zbioru X_2 liter umożliwionych po wx_1 wybieramy najmniejszą (w sensie aktualnego porządku), przedłużamy aktualne obliczenie otrzymując wx_1x_2 i zmieniamy porządek, przesuwając x_2 na koniec. Postępując w ten sposób uzyskamy (skończone nierozszerzalne lub nieskończone) uczciwe obliczenie $wx_1 \ldots x_i \ldots$ ¤

Etykę procesów współbieżnych (zdefiniowanych jako ślady) systemów tranzycyjnych omówimy w rozdziale 5.

3 Unikanie konfliktów w sieciach bezpiecznych

Sieci bezpieczne (zdefiniowane w rozdziale 2.2.2) są interesującym rozszerzeniem sieci elementarnych. Tutaj zajmiemy się zagadnieniem unikania konfliktów w tej klasie sieci.

3.1 Odwracanie sieci bezpiecznych

Zanim przejdziemy do analizy bezkonfliktowych obliczeń, pokażemy pewną interesującą cechę sieci bezpiecznych, ułatwiającą badanie ich zachowania. Proces „odwracania" sieci polega na zamianie łuków typu „reset" na „set", „set" na „reset", „inhib" na „read", „read" na „inhib", „in" na „out", „out" na „in", i zamianie stanu początkowego na jego dopełnienie. Okazuje się, że dowolna sieć i jej odpowiednik odwrócony są zachowaniowo równoważne.

Definicja 3.1. Dla danej sieci bezpiecznej N = (A, P, F, M_0), *odwróceniem* N jest sieć $\underline{\mathrm{N}}$ = (A, P, $\underline{F}$, $\underline{M_0}$), gdzie

$$\underline{F}(a,p) = \begin{cases} \text{nop} & \text{jeśli } F(a,p) = \text{nop} \\ \text{swop} & \text{jeśli } F(a,p) = \text{swop} \\ \text{out} & \text{jeśli } F(a,p) = \text{in} \\ \text{in} & \text{jeśli } F(a,p) = \text{out} \\ \text{set} & \text{jeśli } F(a,p) = \text{reset} \\ \text{reset} & \text{jeśli } F(a,p) = \text{set} \\ \text{inhib} & \text{jeśli } F(a,p) = \text{read} \\ \text{read} & \text{jeśli } F(a,p) = \text{inhib} \end{cases}$$

i $\underline{M_0} = P - M_0$ jest dopełnieniem M_0 do P.

W tym podrozdziale, dla dowolnego stanu $M \subseteq P$, przez $\underline{M} = P - M$ będziemy oznaczać dopełnienie M do P.

Uwaga 3.2. Operacja odwracania jest inwolucją, tzn. $\underline{\underline{\mathrm{N}}}$ = N.

Stwierdzenie 3.3. *Sieć odwrócona jest zachowaniowo równoważna sieci wyjściowej*

Słowo $w \in A^*$ jest obliczeniem w N wtedy i tylko wtedy, gdy jest obliczeniem w $\underline{\mathrm{N}}$.

Dowód. Najpierw pokażemy, że dla dowolnych stanów $X,Y \subseteq P$ i dowolnej akcji $a \in A$, jeśli XaY zachodzi w sieci N, to $\underline{X}a\underline{Y}$ zachodzi w sieci $\underline{\mathrm{N}}$.

Niech Test1(a,N) = in(a,N) $\cup$ read(a,N) = Test0(a,$\underline{\mathrm{N}}$)

i Test0(a,N) = out(a,N) $\cup$ inhib(a,N) = Test1(a,$\underline{\mathrm{N}}$).

Ponieważ a jest umożliwione w X, to Test1(a,N) $\subseteq X$ i Test0(a,N) $\cap X = \emptyset$. Stąd Test1(a,N) $\cap \underline{X} = \emptyset$ i Test0(a,N) $\subseteq \underline{X}$, z definicji $\underline{X}$. Stąd i z poprzednich równości otrzymujemy Test0(a,$\underline{\mathrm{N}}$) $\cap \underline{X} = \emptyset$ i Test1(a,$\underline{\mathrm{N}}$) $\subseteq \underline{X}$, a zatem a jest umożliwione w $\underline{X}$.

Niech teraz Set1(a,N) = out(a,N) $\cup$ set(a,N) = Set0(a,$\underline{\mathrm{N}}$)

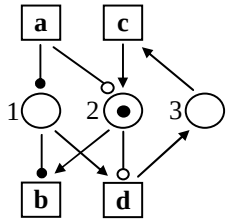
i Set0(a,N) = in(a,N) $\cup$ reset(a,N) = Set1(a,$\underline{\mathrm{N}}$).

Mamy zatem $Y = X \cup$ Set1(a,N) – Set0(a,N).

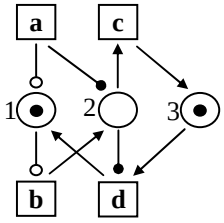
Stąd $\underline{Y} = P - Y = P - (X \cup \text{Set1}(a,\text{N}) - \text{Set0}(a,\text{N})) = P - X - \text{Set1}(a,\text{N}) \cup \text{Set0}(a,\text{N}) = \underline{X} - \text{Set0}(a,\underline{\mathrm{N}}) \cup \text{Set1}(a,\underline{\mathrm{N}})$. Stan $\underline{Y}$ jest więc osiągalny z $\underline{X}$ po wykonaniu a. Stąd

indukcyjnie dla dowolnych stanów $X,Y \subseteq P$ i skończonego obliczenia $w \in A^*$, jeśli XwY w sieci N, to $\underline{X}w\underline{Y}$ w sieci $\underline{N}$.
Dowód w drugą stronę wynika bezpośrednio z Uwagi 3.2. ¤

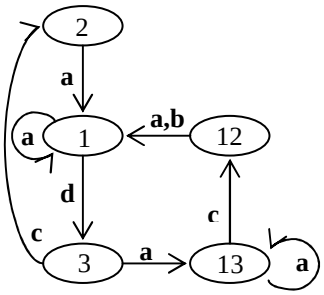
Przykład 3.4. *Sieć bezpieczna i jej odwrócenie*



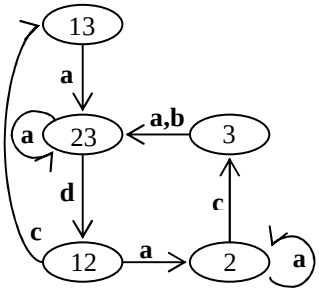
Rys. 16. Sieć bezpieczna N



Rys. 17. Sieć bezpieczna $\underline{N}$ – odwrócenie sieci N



Rys. 18. Graf osiągalności sieci N



Rys. 19. Graf osiągalności sieci $\underline{N}$

Wniosek 3.5. *Odwrócenie konfliktów*
Jeśli $M[a\#b]$ jest konfliktem w sieci bezpiecznej N, to $\underline{M}\ [a\#b]$ jest konfliktem w sieci odwróconej $\underline{N}$.
Dowód. Bezpośrednio ze Stwierdzenia 3.3. ¤

3.2 Obliczenia bezkonfliktowe w sieciach bezpiecznych

Niech $N=(A,P,F,M_0)$ będzie siecia bezpieczną. Mówimy, że warunek $p \in P$ jest *źródłem* konfliktu $M[a\#b]$, jeśli $M_p[a\#b]$, gdzie $M_p=M \cap \{p\}$, w sieci N' otrzymanej z N przez usunięcie wszystkich pozostałych warunków (i łuków z nimi powiązanych) z sieci oryginalnej.

Lemat 3.6. Niech N będzie siecia bezpieczną. Jeśli Ma i Mb, to
$M[a\#b] \Leftrightarrow$ istnieje źródło konfliktu między a i b.

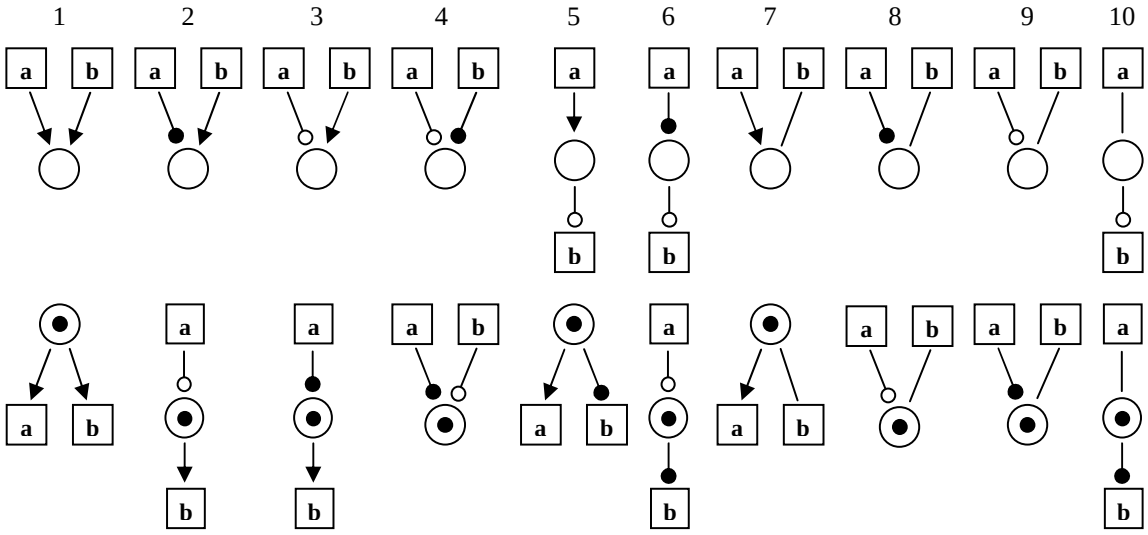
Dowód. ($\Rightarrow$) Przypuśćmy, że $M[a\#b]$ w sieci N, i nie istnieje źródło tego konfliktu. Wtedy dla każdego warunku p, podsieć N_p sieci N, zawierająca tylko p, a, b i łuki między nimi, nie ma konfliktu w stanie M_p. Ponieważ obie akcje a i b są umożliwione w M, są też umożliwione w M_p. Ponieważ ponadto nie ma konfliktu w sieci N_p w stanie M_p, zachodzi równość $M_pab=M_pba$. Niech $S_p=M_pab=M_pba$. Zauważmy, że $S_p=\{p\}$ lub $S_p=\varnothing$. Ale wtedy $Mab=Mba=\bigcup\{S_p \mid p \in P\}$, więc $\neg M[a\#b]$ w sieci N. Sprzeczność.

(⇐) Przypuśćmy, że istnieje źródło konfliktu, czyli warunek p taki, że $M_p[a\#b]$. Wtedy po dodaniu innych warunków do sieci, konflikt pozostaje, o ile tylko akcje a i b są nadal umożliwione. ¤

Lemat 3.7. Niech N będzie siecia bezpieczną. Jeśli $M[a\#b]$ i McM', to McM' jest krokiem konfliktowym albo $M'[a\#b]$.

Dowód. Przypuśćmy, że McM' nie jest krokiem konfliktowym. Pokażemy najpierw, że akcja c nie może być powiązana z warunkiem będącym źródłem konfliktu $M[a\#b]$, tzn. $F(c, p)$=nop dla każdego takiego warunku p.

Można sprawdzić, że istnieje 10 par możliwych rodzajów źródeł konfliktu między a i b. Są one pokazane na Rys. 22:



Rys. 20. Wszystkie przypadki źródeł konfliktu w sieciach bezpiecznych

Zauważmy, że konflikty z dolnego wiersza są odwróceniem konfliktów z wiersza górnego. Ze Stwierdzenia 3.3 i Wniosku 3.5 wynika, że wystarczy więc sprawdzić tylko jeden wiersz, weźmy wiersz górny. Przypuśćmy, że akcja c jest powiązana z warunkiem p będącym źródłem konfliktu między a i b. Wszystkie możliwości są pokazane w poniższej tabeli. Wynika z niej, że w każdym przypadku albo akcja c nie jest umożliwiona, albo jest w konflikcie z a lub b.

Tabela 1. Co się stanie gdy $F(c,p)\neq$nop (przypadki z wiersza górnego Rys. 22)

$F(c,p)$	1	2	3	4	5	6	7	8	9	10
in	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc
out	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b
set	c#a, c#b	c#b	c#a, c#b	c#a	c#a, c#b	c#b	c#a, c#b	c#b	c#a, c#b	c#a, c#b
reset	c#a, c#b	c#a, c#b	c#b	c#b	c#a	c#a	c#a, c#b	c#a, c#b	c#b	c#a
read	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc	¬Mc
inhib	c#a, c#b	c#a, c#b	c#b	c#b	c#a	c#a	c#a, c#b	c#a, c#b	c#b	c#a
swop	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a, c#b	c#a	c#a	c#a	c#b

Zatem akcja *c* nie może być powiązana z warunkiem będącym źródłem konfliktu między *a* i *b*. Jeśli wcale nie ma wspólnych warunków z *a* i *b*, wtedy akcje *a* i *b* są nadal umożliwione w stanie M' i są nadal w konflikcie.

Przypuśćmy teraz, że *c* ma wspólny warunek z *a*, który nie jest źródłem konfliktu. Nazwijmy go *warunkiem neutralnym*. Ponieważ *a* i *c* nie są w konflikcie, akcja *a* jest nadal umożliwiona w stanie M', i konflikt pozostaje w stanie M'. Analogicznie, jeśli *c* ma wspólny warunek neutralny z *b* albo zarówno z *a*, jak i *b*. ¤

Stwierdzenie 3.8. W sieciach bezpiecznych każde obliczenie sprawiedliwe, rozpoczynające się w stanie konfliktowym, zawiera krok konfliktowy.

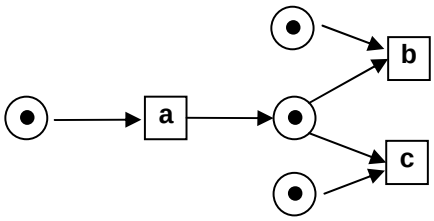
Dowód. Niech M będzie stanem konfliktowym, czyli $M[a\#b]$ dla pewnych $a,b\in A$. Niech $u=Ma_1M_1a_2M_2\ldots$ będzie obliczeniem sprawiedliwym. Przypuśćmy, że jest ono bezkonfliktowe. Wtedy z Lematu 3.7, mamy $M_i[a\#b]$ dla wszystkich $i\geq 1$. Ponieważ *u* jest bezkonfliktowe, zatem $a_i\neq a$ i $a_i\neq b$ dla wszystkich $i\geq 1$. Stąd *u* nie jest sprawiedliwe. Sprzeczność. ¤

Wniosek 3.9. W sieciach bezpiecznych obliczenie sprawiedliwe jest bezkonfliktowe wtedy i tylko wtedy, gdy wszystkie jego stany są bezkonfliktowe.

Dowód. Bezpośrednio ze Stwierdzenia 3.8. ¤

Warto zauważyć, że Stwierdzenie 3.8 i Wniosek 3.9 nie są prawdziwe dla sieci markowanych, nawet skończenie stanowych (ograniczonych).

Przykład 3.10. *Stan konfliktowy w obliczeniu bezkonfliktowym*



Stan początkowy powyższej sieci markowanej jest konfliktowy, ale obliczenie sprawiedliwe *abc* jest bezkonfliktowe.

3.3 Wyszukiwanie obliczeń bezkonfliktowych

Następujący algorytm tworzy dla danej sieci bezpiecznej N automat generujący wszystkie skończone obliczenia bezkonfliktowe, które mają bezkonfliktowe przedłużenia sprawiedliwe w N, i tylko takie. Algorytm działa w oparciu o graf osiągalności traktowany jako automat – redukuje go poprzez usunięcie wszystkich konfliktowych stanów, i tych, które nieuchronnie prowadzą do stanu konfliktowego. Poprawność algorytmu wynika z Wniosku 3.9.

Algorytm 3.11. *Selekcja bezkonfliktowych obliczeń sprawiedliwych*

Wejście: Sieć bezpieczna N=(A,P,F,M_0)

Krok wstępny:

Zbuduj graf osiągalności dla N:

$G_N = (V,E,M_0)$, gdzie V=RS(N) – zbiór stanów osiągalnych
E={MaM' | $M,M' \in$ RS(N) i MaM' jest krokiem w N};

Ustaw

(1) G := G_N

Krok początkowy:

Usuń wszystkie stany konfliktowe z V:

(2) V := {$M \in V$ | M jest bezkonfliktowy};

Zbuduj pomocniczy zbiór SE półkrawędzi:

(3) SE := {Ma | ($\exists M'$) $MaM' \in E$ i M' jest stanem konfliktowym};

Zredukuj zbiór krawędzi:

(4) E := {$MaM' \in E$ | M i M' są bezkonfliktowe};

Sprawdzenie:

if $M_0 \notin V$ **go to Wyjście 1**;

if $SE=\emptyset$ **go to Wyjście 2**;

Weź półkrawędź $Ma \in SE$, ustaw

(5) $SE := SE-\{Ma\}$,

Sprawdź, czy istnieje ścieżka od M (w aktualnym grafie G) przechodząca przez a,
tzn. ścieżka $Ma_1M_1a_2M_2\ldots a_kM_k$ taka, że dla pewnego i mamy $a_i=a$;

Jeśli taka ścieżka istnieje, **go to Sprawdzenie**;

Jeśli nie:

Usuń M z V:

(6) $V := V-\{M\}$;

Dodaj nowe krawędzie do SE:

(7) $SE := SE \cup \{M'a \mid M'aM \in E\}$;

Zredukuj zbiór krawędzi:

(8) $E := E-\{M'aM'' \mid M''=M\}$;

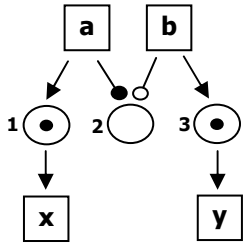
go to Sprawdzenie;

Wyjście 1: Nie istnieje w N sprawiedliwe obliczenie bezkonfliktowe; **STOP**;

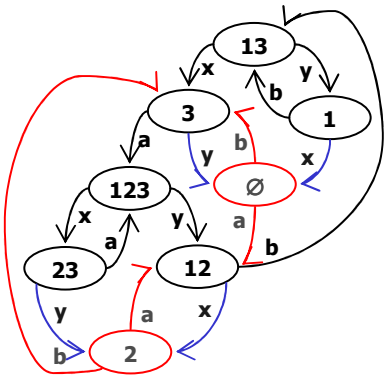
Wyjście 2: Wynikowym zredukowanym grafem G_{cf} jest aktualny graf G; **STOP**.

Koniec algorytmu 3.11 ¤

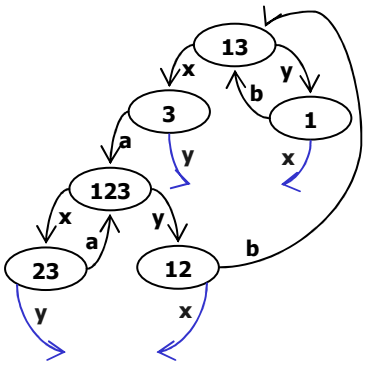
Przykład 3.12. *Działanie algorytmu*



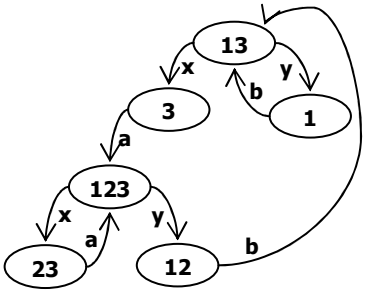
Rys. 21. Przykładowa sieć N



Rys. 22. Graf osiągalności sieci N



Rys. 23. Po usunięciu stanów konfliktowych



Rys. 24. Wynikowy graf osiągalności

Stwierdzenie 3.13. *Poprawność algorytmu 3.11*
Wynikowy graf G_{cf} (traktowany jako automat) generuje dokładnie wszystkie skończone obliczenia bezkonfliktowe sieci N, które mają bezkonfliktowe i sprawiedliwe rozszerzenia.

Dowód. Oczywiście, oryginalny graf osiągalności G_N generuje dokładnie wszystkie skończone obliczenia sieci N. Zatem G_N przechowuje wszystkie (skończone i nieskończone) obliczenia sieci N. Z Wniosku 3.9, wszystkie bezkonfliktowe obliczenia sprawiedliwe są nadal możliwe w grafie G, zredukowanym w kroku początkowym. Podczas pętli sprawdzającej usuwane są tylko stany, które nieuchronnie prowadzą do konfliktu. Stąd znowu z Wniosku 3.9, wszystkie bezkonfliktowe obliczenia sprawiedliwe sieci N są nadal zachowane w grafie G. Ponieważ każde skończone obliczenie posiada rozszerzenie sprawiedliwe (Fakt 2.36 + fakt, że każde obliczenie uczciwe jest sprawiedliwe), zbiór wszystkich bezkonfliktowych obliczeń sprawiedliwych sieci N jest niepusty, o ile stan początkowy M_0 należy do G. ¤

Graf G_{cf} wyprodukowany przez Algorytm 3.11 może pełnić funkcję sterującą bezkonfliktowym działaniem sieci N. Warunek, że sieć może wykonywać jedynie ruchy dozwolone przez graf G_{cf} zapewnia bezkonfliktowość obliczeń sieci, na mocy Stwierdzenia 3.13.

4 Wyznaczanie relacji niezależności dla sieci Petriego

W podrozdziale 2.4 zdefiniowaliśmy zachowaniową wersję relacji niezależności akcji (Definicje 2.25 i 2.26). Pojawia się pytanie o to, czy tak zdefiniowana relacja jest obliczalna, tzn. czy można rozstrzygnąć dla każdej pary akcji (a,b), czy $a\mathrm{I}_S b$ ($a\mathrm{I}_L b$). Zauważmy, że jeśli system tranzycyjny S jest skończenie stanowy, to niezależności I_S i I_L są obliczalne. Stąd niezależność jest obliczalna dla sieci bezpiecznych (więc także elementarnych) oraz dla ograniczonych sieci markowanych, ponieważ ich grafy osiągalności są skończone. W tym rozdziale przyjrzymy się problemowi obliczalności niezależności dla dowolnych sieci markowanych i niektórych ich rozszerzeń.

Najpierw przypomnijmy pewne znane problemy związane z sieciami Petriego. Ich rozstrzygalność lub nierozstrzygalność będzie pomocna w dowodzeniu rozstrzygalności lub nierozstrzygalności problemu wyznaczenia relacji niezależności.

Problem osiągalności: Dla danej sieci (N,M_0) i stanu M, rozstrzygnąć, czy M jest osiągalny w (N,M_0);

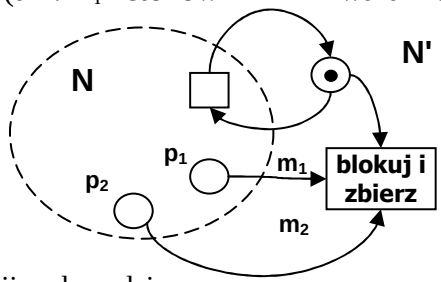
Problem pokrywalności: Dla danej sieci (N,M_0) i stanu M, rozstrzygnąć, czy istnieje stan osiągalny M', taki, że $M'\geq M$;

Problem pustości warunku: Dla danej sieci (N,M_0) i warunku p, rozstrzygnąć, czy istnieje stan osiągalny M, taki, że $M(p)=0$.

Wiadomo, że wszystkie te problemy są rozstrzygalne w klasie sieci markowanych. Najtrudniejszy z nich to problem osiągalności, którego rozstrzygalności dowiedli Mayr [31] i Kosaraju [26]. Jest on równoważny problemowi osiągalności stanu pustego, tj. takiego stanu M, że $M(p)=0$ dla każdego warunku p.

Stwierdzenie 4.1. (Hack [21]) Problem osiągalności jest rozstrzygalny wtedy i tylko wtedy, gdy problem osiągalności stanu pustego jest rozstrzygalny.

Dowód. ($\Rightarrow$) Oczywiste. ($\Leftarrow$) Załóżmy, że problem osiągalności stanu pustego jest rozstrzygalny. Niech $\mathrm{N}=(A,P,F,M_0)$ będzie siecią, w której $P=\{p_1, p_2, \ldots, p_k\}$, a $M=[m_1,\ldots,m_k]$ niech będzie stanem, którego osiągalność chcemy zbadać, $m_i\in\mathbb{N}$. Konstruujemy nową sieć N' dodając do N nowy warunek będący jednocześnie wejściem i wyjściem każdej akcji sieci N oraz nową akcję *blokuj i zbierz*, która blokuje działanie sieci N (zdejmuje żeton z dodanego warunku) i zdejmuje odpowiednią ilość żetonów z każdego warunku sieci N (tzn. m_i żetonów z warunku p_i).



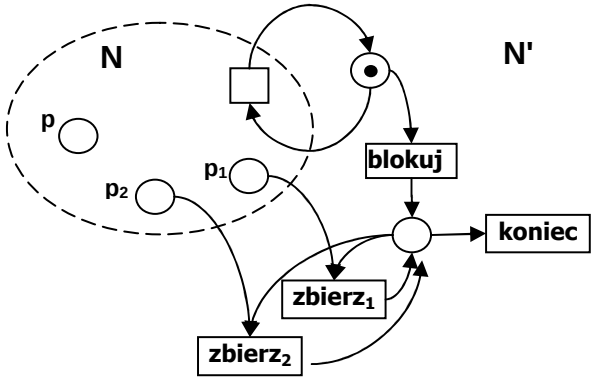
Rys. 25. Schemat konstrukcji w dowodzie Stwierdzenia 4.1

Zauważmy teraz, że stan M jest osiągalny w N wtedy i tylko wtedy, gdy stan pusty jest osiągalny w sieci N', co kończy dowód. ¤

Powyższe twierdzenie jest prawdziwe w dowolnej klasie sieci rozszerzającej klasę sieci markowanych. Kolejna konstrukcja dowodząca równoważności problemów osiągalności i pustości warunku została pokazana przez Hacka [21]:

Stwierdzenie 4.2. W klasie sieci markowanych problem osiągalności stanu pustego jest rozstrzygalny wtedy i tylko wtedy, gdy problem pustości warunku jest rozstrzygalny.

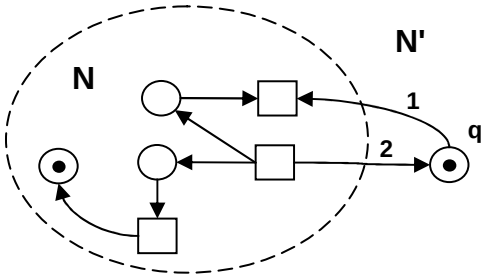
Dowód. (⇒) Niech N=(A,P,F,M_0) będzie siecią i $p \in P$ warunkiem, którego pustość chcemy zbadać. Nowa sieć N' ma mechanizm blokujący podobny jak w poprzedniej konstrukcji i dodatkowo akcje $zbierz_i$ dla każdego warunku $p_i \neq p$ – każda z nich zbiera po jednym żetonie ze „swojego" warunku p_i.



Rys. 26. Schemat konstrukcji w dowodzie Stwierdzenia 4.2 (⇒)

Stan z pustym warunkiem p jest osiągalny w N wtedy i tylko wtedy, gdy stan pusty jest osiagalny w N'.

(⇐) Teraz załóżmy, że umiemy rozstrzygać o osiągalności stanu z pustym warunkiem. Niech N=(A,P,F,M_0) będzie siecią, dla której chcemy zbadać osiągalność stanu pustego. Nowa sieć N' ma jeden dodatkowy warunek q, w którym w stanie początkowym znajduje się tyle żetonów, ile w całej sieci N w stanie początkowym: $M'_0(q)=M_0(p_1)+\ldots+M_0(p_k)$. Warunek q łączymy z każdą akcją a łukiem $a \xrightarrow{m} q$ z wagą m, jeśli wykonanie akcji a zwiększa ilość żetonów w sieci N o m, a łukiem $q \xrightarrow{m} a$, jeśli zmniejsza o m. Jeśli akcja nie zmienia liczby żetonów, to nie łączymy jej z warunkiem q. W ten sposób warunek q stale przechowuje aktualną ilość żetonów w całej sieci N.



Rys. 27. Schemat konstrukcji w dowodzie Stwierdzenia 4.2 (⇐)

Stan pusty jest osiągalny w sieci N wtedy i tylko wtedy, gdy stan z pustym warunkiem q jest osiągalny w sieci N'. ¤

Konstrukcje ze Stwierdzenia 4.2, zastosowane do sieci inhibitorowych lub przerzucających, pokazują, że analogiczne twierdzenie zachodzi dla tychże sieci:

Stwierdzenie 4.3. W sieciach inhibitorowych i przerzucających problem osiągalności stanu pustego jest rozstrzygalny wtw, gdy problem pustości warunku jest rozstrzygalny.

Zauważmy, że konstrukcje te nie są już odpowiednie dla sieci, w których z góry nie wiadomo, ile żetonów będzie dołożonych lub usuniętych z warunku po wykonaniu jakiejś akcji, czyli w szczególności dla sieci czyszczących.

4.1 Niezależność w sieciach markowanych

W tym podrozdziale będziemy pomijać oznaczenie, o którą niezależność chodzi (językową czy systemową), gdyż są one równe w sieciach markowanych: $I=I_L=I_S$. Pokażemy, że niezależność jest obliczalna w dowolnych sieciach markowanych.

Najpierw rozważymy pewne pomocnicze problemy, których rozstrzygalność będzie wykorzystana w dowodzie rozstrzygalności zależności.

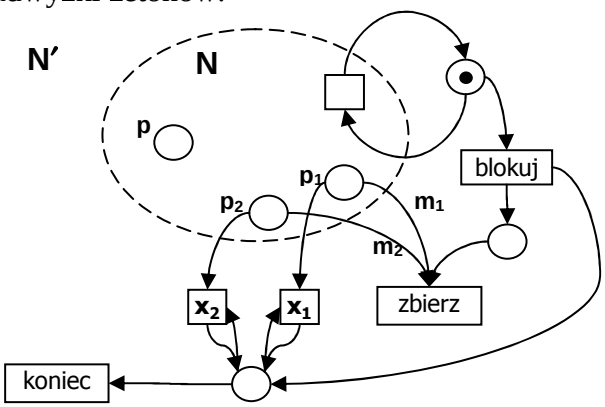
Lemat 4.4. Problem

Dane: Sieć N, stan M, warunek p.

Pytanie: Czy istnieje osiągalny stan M' taki, że $M'(p)=0$ i $(\forall q\neq p)M'(q)\geq M(q)$?

jest rozstrzygalny w klasie sieci markowanych.

Dowód. Niech k będzie ilością warunków sieci N i niech $M = [m_1,\ldots, m_k]$. Konstruujemy nową sieć N' (z wagami) dodając mechanizm blokujący działanie sieci N jak w poprzednich konstrukcjach. Akcja *blokuj* uniemożliwia wykonywanie akcji sieci N oraz inicjuje zliczanie ilości żetonów w warunkach sieci N. Zliczanie odbywa się w dwóch etapach: najpierw zdejmujemy wymaganą (przez stan M) ilość żetonów, potem (pojedynczo) nadwyżki. Zatem z każdego warunku p_i różnego od p dodajemy łuki z wagą m_i do akcji *zbierz*. Oprócz tego dla każdego warunku p_i różnego od p dodajemy akcję x_i zbierającą nadwyżki żetonów.



Rys. 28. Schemat konstrukcji w dowodzie Lematu 4.4

Teraz stan M' taki, że $M'(p)=0$ i $M'(p_i)\geq M(p_i)$ dla $p_i\neq p$ jest osiągalny w sieci N wtedy i tylko wtedy, gdy stan pusty jest osiągalny w sieci N'. ¤

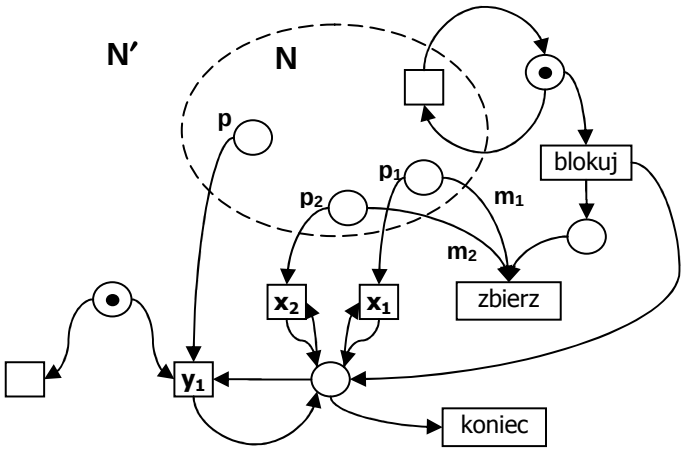
Lemat 4.5. Problem

Dane: Sieć N, stan M, warunek p, $k \in \{1, 2, \ldots\}$.

Pytanie: Czy istnieje osiągalny stan M' taki, że $M'(p) \le k$ i $(\forall q \ne p) M'(q) \ge M(q)$?

jest rozstrzygalny w klasie sieci markowanych.

Dowód. Do poprzedniej konstrukcji trzeba dołączyć k akcji $y_1, y_2, \ldots, y_k$, każda z nich zdejmuje (co najwyżej) jeden żeton z warunku p.



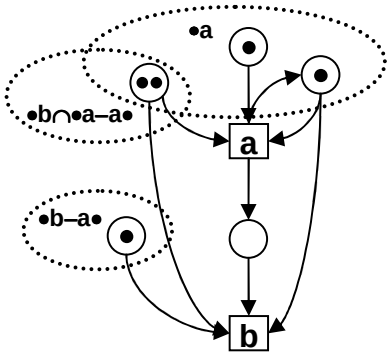
Rys. 28. Schemat konstrukcji w dowodzie Lematu 4.5

Teraz stan M' taki, że $M'(p) < k$ i $M'(p_i) \ge M(p_i)$ dla $p_i \ne p$ jest osiągalny w sieci N wtedy i tylko wtedy, gdy stan pusty jest osiągalny w sieci N'. ¤

Możemy teraz pokazać, że relacja zależności (a więc i niezależności) akcji jest obliczalna w sieciach markowanych.

Twierdzenie 4.6. Problem zależności „Czy dane dwie akcje a i b sieci N są zależne?" jest rozstrzygalny w klasie sieci markowanych.

Dowód. Najpierw rozważmy, jaki musi być minimalny stan M, aby zachodziło Mab. Warunki wejściowe dla akcji a i b możemy podzielić na trzy grupy: wejścia do a (muszą mieć co najmniej 1 żeton); wejścia do b, które nie są wyjściami a (muszą mieć co najmniej 1 żeton); wspólne wejścia do a i b, które nie są wyjściami dla a (muszą mieć co najmniej 2 żetony).



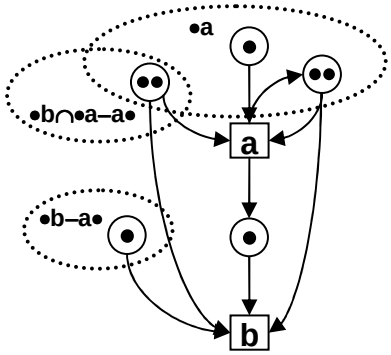
Zatem Mab zachodzi wtedy i tylko wtedy, gdy

(1) $(\forall p \in {}^\bullet a)\ M(p) \ge 1$

i (2) $(\forall p \in {}^\bullet b - a^\bullet)\ M(p) \ge 1$

i (3) $(\forall p \in {}^\bullet b \cap ({}^\bullet a - a^\bullet))\ M(p) \ge 2$.

Aby dodatkowo zachodziło *Mba*, warunek, który jest wyjściem *a* i wejściem *b*, musi mieć co najmniej 1 żeton, oraz warunek, który jest wejściem dla *a* i *b* oraz wyjściem dla *a*, ale nie dla *b*, musi mieć co najmniej 2 żetony.



Pokażemy, że $Mab \wedge (*) \Rightarrow Mba$, gdzie

(*) $(\forall p \in a^\bullet \cap {}^\bullet b)\ M(p) \geq 1 \wedge$
$(\forall p \in {}^\bullet a \cap a^\bullet \cap ({}^\bullet b - b^\bullet))\ M(p) \geq 2.$

Mba wtedy i tylko wtedy, gdy

(4) $(\forall p \in {}^\bullet b)\ M(p) \geq 1$ z (2) i (*)

i (5) $(\forall p \in {}^\bullet a - b^\bullet)\ M(p) \geq 1$ z (1)

i (6) $(\forall p \in {}^\bullet a \cap ({}^\bullet b - b^\bullet))\ M(p) \geq 2$ z (3) i (*).

Z definicji $a\mathrm{D}b \Leftrightarrow (\exists M)\ (Mab \wedge \neg Mba) \vee (Mba \wedge \neg Mab)$. Z powyższych rozważań $Mab \wedge \neg Mba \Leftrightarrow Mab \wedge \neg(*)$. Aby rozstrzygnąć, czy $a\mathrm{D}b$, wystarczy więc sprawdzić, czy istnieje stan *M* taki, że $Mab \wedge \neg(*)$, tzn. należy sprawdzić, czy osiągalny jest stan *M* spełniający warunki (1), (2), (3) oraz taki, że w pewnym warunku $p \in a^\bullet \cap {}^\bullet b$ zachodzi $M(p)=0$ (sprawdzalne z Lematu 4.4) lub w pewnym $p \in {}^\bullet a \cap a^\bullet \cap ({}^\bullet b - b^\bullet)$ zachodzi $M(p)<2$ (sprawdzalne z Lematu 4.5). ¤

Aby wyznaczyć relację niezależności należy sprawdzić, czy $a\mathrm{D}b$ dla każdej pary akcji *a,b*. Stąd otrzymujemy wniosek:

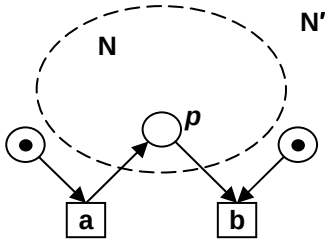
Wniosek 4.7. Niezależność jest obliczalna w klasie sieci markowanych.

4.2 Niezależność w niektórych rozszerzeniach sieci markowanych

Poniższe stwierdzenie zachodzi w dowolnej klasie sieci rozszerzającej klasę sieci markowanych i jest prawdziwe dla obu typów niezależności.

Stwierdzenie 4.8. Jeśli problem pustości warunku jest nierozstrzygalny w pewnej klasie sieci Petriego, to problem zależności „Czy dane dwie akcje *a* i *b* sieci N są zależne?" jest nierozstrzygalny dla tej klasy sieci.

Dowód. Przypuśćmy, że problem zależności jest rozstrzygalny. Pokażemy, że wtedy problem pustości warunku jest rozstrzygalny. Dla danej sieci $N=(A,P,F,M_0)$ i warunku $p \in P$ chcemy rozstrzygnąć, czy istnieje osiągalny stan *M* taki, że $M(p)=0$. Konstruujemy sieć N′ w następujący sposób: dodajemy dwie nowe akcje *a,b* i dwa warunki powiązane z nimi jak na Rys. 29.



Rys. 29. Schemat konstrukcji w dowodzie Stwierdzenia 4.8

Stan M taki, że $M(p)=0$, jest osiągalny w sieci (N,M_0) wtedy i tylko wtedy, gdy akcje a i b są zależne w sieci N'. ¤

W powyższym dowodzie możemy pod pojęciem „zależność akcji" rozumieć zarówno zależność językową, jak i systemową – nie ma to wpływu na konstrukcję, dlatego że akcje a i b z konstrukcji używają tylko zwykłych łuków, są więc zależne językowo dokładnie wtedy, gdy są zależne systemowo.

Stwierdzenie 4.8 pozwala na orzekanie o nierozstrzygalności niezależności w klasach sieci Petriego z nierozstrzygalnym problemem pustości warunku. Pokażemy teraz, że problem pustości warunku jest nierozstrzygalny w rozważanych wcześniej rozszerzeniach sieci markowanych: w sieciach inhibitorowych, czyszczących i przerzucających.

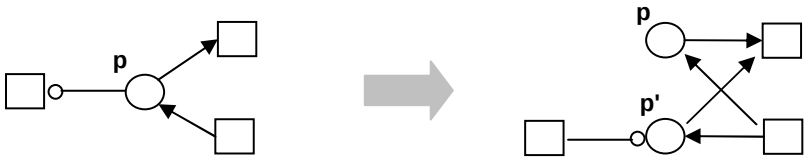
Przytoczymy najpierw twierdzenia o nierozstrzygalności problemu osiągalności dla sieci inhibitorowych (Minsky [36]) i czyszczących (Araki/Kasami [1]).

Twierdzenie 4.9. Problem osiągalności jest nierozstrzygalny w sieciach inhibitorowych.
Dowód. Chrząstowski-Wachtel [7] udowodnił nierozstrzygalność problemu osiągalności w sieciach inhibitorowych, konstruując sieć inhibitorową, realizującą mnożenie i korzystając z nierozstrzygalności dziesiątego problemu Hilberta (Matiyasevich [30]). ¤

Wniosek 4.10. Problem pustości warunku jest nierozstrzygalny w sieciach inhibitorowych.
Dowód. Nierozstrzygalność problemu pustości warunku dla sieci inhibitorowych wynika wprost z nierozstrzygalności osiągalności w tych sieciach (Twierdzenie 4.9) i równoważności problemu osiągalności i pustości warunku (Stwierdzenie 4.3). ¤

Jak zauważyliśmy wcześniej, konstrukcji z dowodów Stwierdzeń 4.2 i 4.3 nie można zastosować do sieci czyszczących. Nierozstrzygalność problemu pustości warunku w tej klasie sieci pokażemy wprost, korzystając z pomysłu Dufourd/Finkel/Schoebelen [15] i rozszerzając konstrukcję ze Stwierdzenia 4.2. Najpierw przypomnimy ideę z pracy [15] pokazującą, w jaki sposób z nierozstrzygalności osiągalności dla sieci inhibitorowych wynika nierozstrzygalność osiągalności dla sieci czyszczących:

Twierdzenie 4.11. Problem osiągalności jest nierozstrzygalny w sieciach czyszczących.
Dowód. (Dufourd/Finkel/Schoebelen [15]) Pokażemy, że gdyby problem osiągalności dla sieci czyszczących był rozstrzygalny, to także problem osiągalności dla sieci inhibitorowych byłby rozstrzygalny: Niech N będzie siecią inhibitorową. Konstruujemy z niej sieć czyszczącą N' w taki sposób, że dla każdego warunku inhibitorowego p (tzn. takiego, że istnieje akcja połączona z nim łukiem inhibitorowym) tworzony jest warunek bliźniaczy p', z łukami (zwykłymi) takimi jak warunek p. Każdy łuk inhibitorowy wychodzący z p jest zamieniany na łuk czyszczący wchodzący do p'.



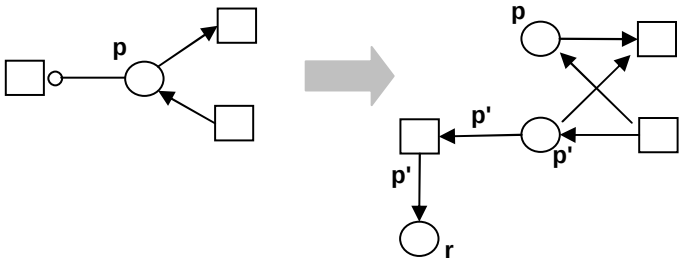
Rys. 30. Schemat konstrukcji w dowodzie Twierdzenia 4.11

Zauważmy, że w każdym osiągalnym stanie M sieci N' mamy $M(p) \geq M(p')$. Równość $M(p)=M(p')$ zachodzi wtedy, gdy obliczenie prowadzące do M jest jednocześnie obliczeniem sieci N (sieć N' działa zgodnie z siecią N), tzn. akcja czyszcząca jest wykonywana tylko wtedy, gdy warunek czyszczony p' jest pusty. Jeśli natomiast akcja czyszcząca wykona się, gdy p' nie jest pusty, to $M(p)>M(p')$ i ponadto ta ostra nierówność zostanie zachowana dla wszystkich stanów osiągalnych z M, tzn. jeśli $M(p)>M(p')$ i MaM' dla pewnej akcji a, to $M'(p)>M'(p')$. Zauważmy teraz, że stan M jest osiągalny w N wtedy i tylko wtedy, gdy w N' jest osiągalny stan M' taki, że $M'(p)=M(p)$ dla wszystkich warunków p sieci N oraz $M'(p')=M(p)$ dla wszystkich (nowo utworzonych) warunków p' bliźniaczych dla warunków inhibitorowych p. ¤

Analogiczną konstrukcję można zastosować do udowodnienia nierozstrzygalności problemu osiągalności dla sieci przerzucających.

Twierdzenie 4.12. Problem osiągalności jest nierozstrzygalny w sieciach przerzucających.

Dowód. Konstrukcję z Twierdzenia 4.11 wystarczy uzupełnić o dodatkowy warunek r, do którego będą „przerzucane” żetony z warunków p'.



Rys. 31. Schemat konstrukcji w Twierdzeniu 4.12

Stan M jest osiągalny w N wtedy i tylko wtedy, gdy w N' jest osiągalny stan M' taki, że $M'(p)=M(p)$ dla warunków p sieci N, $M'(p')=M(p)$ dla warunków p' bliźniaczych dla warunków inhibitorowych p oraz $M'(r)=0$. ¤

Stwierdzenie 4.13. Problem pustości warunku jest nierozstrzygalny w sieciach przerzucających.

Dowód. Nierozstrzygalność problemu pustości warunku dla sieci przerzucających wynika wprost z nierozstrzygalności osiągalności w tych sieciach (Twierdzenie 4.12) i równoważności problemu osiągalności i pustości warunku (Stwierdzenie 4.3). ¤

Niestety powyższej konstrukcji nie da się bezpośrednio zastosować do udowodnienia nierozstrzygalności problemu pustości warunku w sieciach czyszczących, gdyż nie jest prawdziwa równoważność: istnieje osiągalny stan M sieci N taki, że $M(p)=0 \Leftrightarrow$ gdy istnieje osiągalny stan M sieci N' taki, że $M(p)=0$. Nie jest ona prawdziwa zarówno dla zwykłych warunków (Rys. 32), jak i dla warunków inhibitorowych (Rys. 33). Dzieje się tak dlatego, że pustość warunku w sieci N' mogła być osiągnięta poprzez nie-inhibitorowe działanie sieci N', czego nie można sprawdzić badając lokalnie zawartość tylko jednego warunku, nie odwołując się do całego stanu sieci.

Rys. 32. W sieci N' pustość warunku *q* jest osiągalna, w sieci N – nie

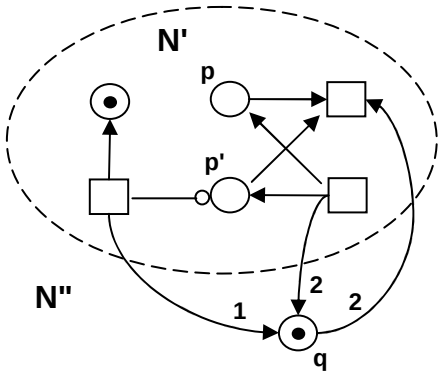
Rys. 33. W sieci N' pustość (inhibitorowego) warunku *p* jest osiągalna, w sieci N – nie

Można jednak udowodnić nierozstrzygalność problemu pustości warunku w sieciach czyszczących, łącząc powyższą konstrukcję z konstrukcją ze Stwierdzenia 4.2 i sprowadzając problem pustości warunku dla sieci czyszczących do problemu osiągalności stanu pustego w sieciach inhibitorowych. Ten ostatni problem jest nierozstrzygalny, ponieważ problem osiągalności i osiągalności stanu pustego są równoważne w sieciach będących rozszerzeniem sieci markowanych (Stwierdzenie 4.1), więc także w sieciach inhibitorowych, a problem osiągalności jest nierozstrzygalny w sieciach inhibitorowych (Twierdzenie 4.9).

Stwierdzenie 4.14. Problem pustości warunku jest nierozstrzygalny w sieciach czyszczących.

Dowód. Podobnie jak w dowodzie Twierdzenia 4.11 konstruujemy sieć czyszczącą z sieci inhibitorowej, dodając warunki bliźniacze i zamieniając łuki inhibitorowe na czyszczące, a następnie dodajemy (jak w Stwierdzeniu 4.2) zbiorczy warunek q, który początkowo ma tyle żetonów, ile jest w całej sieci N' w stanie początkowym. Bezpośrednia konstrukcja Hacka nie była możliwa dla sieci czyszczących, ponieważ nie można z góry określić, ile żetonów znika z sieci po użyciu akcji czyszczącej. Teraz łączymy q z każdą akcją a łukiem ważonym $q \xrightarrow{i} a$, gdzie i jest liczbą warunków

wchodzących do a, i łukiem ważonym $a \xrightarrow{j} q$, gdzie j jest liczbą warunków wychodzących z a, bez uwzględniania warunków czyszczonych. Teraz warunek q nie rejestruje zmian wynikających z użycia akcji czyszczących. Jeśli akcja w sieci N' czyści warunek niepusty p', to liczba żetonów w q staje się większa niż liczba żetonów w sieci N'. Ponieważ odtąd zawsze $M(p)>M(p')$, więc stan pusty jest teraz nieosiągalny w N'', zatem q też będzie już zawsze niepusty.



Rys. 34. Schemat konstrukcji w dowodzie Stwierdzenia 4.14

Stąd stan pusty jest osiągalny w N ⇔ stan pusty jest osiągalny w N' ⇔ stan z pustym warunkiem q jest osiągalny w N''. ¤

Następujące twierdzenie prosto wynika z wyżej uzyskanych wyników:

Twierdzenie 4.15. Problem zależności „Czy dane akcje a i b są zależne?" jest nierozstrzygalny w sieciach inhibitorowych, czyszczących i przerzucających.
Dowód. Bezpośrednio z Wniosku 4.10 i Stwierdzeń 4.8, 4.13, 4.14. ¤

4.3 Problem śladowości zachowania

W pewnych systemach współbieżnych (np. w sieciach elementarnych), lokalna współbieżność pociąga za sobą globalną niezależność. Współbieżne zachowanie takich systemów jest w pełni opisywalne za pomocą śladów. Istnieją jednakże systemy (na przykład sieci markowane), które dopuszczają istnienie par akcji zależnych (tzn. aD_qb w pewnym stanie q), a jednocześnie lokalnie współbieżnych (tzn. $a||_{q'}b$ w innym stanie q'). Rozważmy następujący przykład:

Przykład 4.16. *Sieć markowana z nieśladowym zachowaniem*

Rys. 35. Akcje są zależne, ale lokalnie współbieżne

Akcje *a, b* są zależne, ponieważ w stanie $M=\varnothing$ zachodzi *Mab*, ale nie zachodzi *Mba*. Graf śladu [*aaab*] wygląda zatem następująco: $a \to a \to a \to b$, podczas gdy po wykonaniu pierwszego *a* akcje *a, b* są współbieżne: $a \to a \to a$ z $a \searrow b$

To jest powód, dla którego proponujemy pojęcie systemów z zachowaniem śladowym.

Definicja 4.17. *Zachowanie śladowe*
Zachowanie systemu tranzycyjnego $S=(A,Q,q_0)$ jest *śladowe* wtedy i tylko wtedy, gdy

$$(\forall a,b\in A)\ ((\exists q\in Q)\ a||_q b\ ) \Rightarrow aI_S b.$$

Innymi słowy, zachowania systemu tranzycyjnego nie jest śladowe, jeśli istnieje para akcji *a,b* i stany q,q', takie, że *a,b* są lokalnie współbieżne w *q*, i lokalnie zależne w q':

$$(\exists a,b\in A)\ ((\exists q\in Q)\ a||_q b) \wedge ((\exists q'\in Q)\ aD_{q'}b).$$

Problem decyzyjny „Czy zachowanie danego systemu (danej sieci) jest śladowe?" nazwiemy *problemem śladowości (zachowania).*

W systemach o zachowaniu śladowym ślady nie gubią żadnej informacji o możliwości współbieżnego wykonania akcji. Współbieżne zachowanie takiego systemu jest więc w pełni wyrażalne śladami. Sieci elementarne są takimi systemami (Mazurkiewicz [32]), a sieci markowane na ogół nie (Przykład 4.16).

Pokażemy, że problem problemem śladowości jest rozstrzygalny w klasie sieci markowanych. Najpierw zauważmy, że problem lokalnej współbieżności jest rozstrzygalny:

Lemat 4.18. Problem współbieżności
Dane: Sieć N, akcje *a* i *b*.
Pytanie: czy istnieje stan *M* taki, że $a||_M b$?
jest rozstrzygalny w sieciach markowanych.

Dowód. Wynika to z rozstrzygalności problemu pokrywalności. Rozstrzygamy mianowicie, czy istnieje stan osiągalny *M* pokrywający stan *M'* taki, że

$M'(p)=2$ dla wspólnych wejść, czyli dla $p\in {}^\bullet a\cap {}^\bullet b$;
$M'(p)=1$ dla pozostałych wejść, czyli dla $p\in {}^\bullet a - {}^\bullet b \cup {}^\bullet b - {}^\bullet a$;
$M'(p)=0$ dla pozostałych warunków *p*. ¤

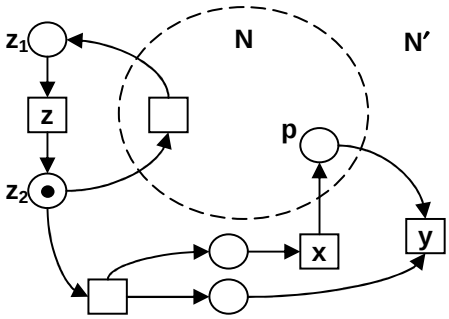
Twierdzenie 4.19. Problem śladowości jest rozstrzygalny w sieciach markowanych.
Dowód. Dla każdej pary akcji sprawdzamy, czy są one zależne (Twierdzenie 4.6) i czy są lokalnie współbieżne w jakimś stanie (Lemat 4.18). Jeśli odpowiedź jest twierdząca dla obu pytań, to zachowanie sieci nie jest śladowe, w przeciwnym przypadku – jest. ¤

Przykładem sieci markowanej o zachowaniu śladowym jest sieć z Przykładu 2.24.

Teraz pokażemy, że problem śladowości jest nierozstrzygalny w tych klasie sieci, w których problem pustości warunku jest nierozstrzygalny. Tu również konstrukcja ma

tę własność, że jest odpowiednia zarówno dla śladów wygenerowanych przez niezależność językową, jak i systemową; i również zachodzi w dowolnej klasie sieci będącej rozszerzeniem klasy sieci markowanych.

Lemat 4.20. Jeśli problem śladowości zachowania jest rozstrzygalny w pewnej klasie sieci (rozszerzającej klasę sieci markowanych), to problem pustości warunku jest też rozstrzygalny w tej klasie sieci.
Dowód. Aby rozstrzygnąć, czy dla danej sieci N = (A, P, F, M_0) i placu $p \in P$ istnieje osiągalny stan M taki, że $M(p)=0$, konstruujemy sieć N′ w następujący sposób: dodajemy nowe akcje *x,y* powiązane jak na rysunku, nową akcję *z* połączoną z dwoma warunkami z_1 i z_2, i dla każdej akcji z sieci N dodajemy nowe łuki prowadzące z warunku z_2 i do warunku z_1.



Rys. 36. Schemat konstrukcji w dowodzie Lematu 4.20

Zauważmy, że każde obliczenie *abc* sieci N ma postać *azbzcz* w sieci N′ – powoduje to, że żadna para oprócz (*x,y*) nie może być lokalnie współbieżna w sieci N′. Akcje *x,y* są ponadto lokalnie współbieżne dokładnie wtedy, gdy istnieje osiągalny stan M taki, że $M(p)\neq 0$ i są lokalnie zależne dokładnie wtedy, gdy istnieje osiągalny stan M taki, że $M(p)=0$.
Zatem jeśli $M_0(p)=0$, to stan M taki, że $M(p)=0$ jest oczywiście osiągalny w N. Jeśli $M_0(p)>0$, to $x \|_{M_0} y$ (są lokalnie współbieżne w M_0). Zatem wtedy stan M taki, że $M(p)=0$, jest osiągalny w N wtedy i tylko wtedy, gdy akcje *x, y* są lokalnie współbieżne (w M_0) i lokalnie zależne (w M), czyli dokładnie wtedy, gdy zachowanie sieci N′ nie jest śladowe. Zatem rozstrzygalność problemu śladowości zachowania implikuje rozstrzygalność problemu pustości warunku. ¤

Twierdzenie 4.21. Problem śladowości zachowania jest nierozstrzygalny dla sieci inhibitorowych, czyszczących i przerzucających.
Dowód. Bezpośrednio z Wniosku 4.10, Stwierdzeń 4.13, 4.14 i Lematu 4.20. ¤

5 Etyka procesów współbieżnych

Zagadnienie uogólnienia sekwencyjnych pojęć uczciwości na obliczenia współbieżne nie jest nowe. Potraktowanie śladu jako zbioru jego linearyzacji (podejście przeplotowe) prowadzi do podzielenia pewnych własności (np. uczciwości) na uniwersalne (wszystkie linearyzacje mają tę własność) i egzystencjalne (jakaś linearyzacja ma tę własność). Podobne typy uczciwości były definiowane i badane w kilku pracach, ale jedynie dla sieci elementarnych (Peled/Pnueli [39], Völzer [46]), bez wykorzystania teorii śladów. Tutaj zbadamy dowolne sieci markowane i przedstawimy dokładną hierarchię uczciwości procesów dla tych sieci.

Można też przenieść pojęcia uczciwości na ślady bez operacji linearyzacji. W tym podejściu (nieprzeplotowym) nie dzielimy śladu na jego linearyzacje (podział poziomy), ale na prefiks i ogon (podział pionowy). To podejście omówione będzie w podrozdziale 5.3.

5.1 Procesy systemów tranzycyjnych – klasyfikacja etyczna

Następujące definicje prezentują podejście przeplotowe do śladów – traktują ślad jako zbiór jego linearyzacji.

Definicja 5.1. *Procesy sprawiedliwe, uczciwe i superuczciwe*

Niech $S = (A, Q, q_0)$ będzie systemem tranzycyjnym, I – niezależnością indukowaną przez S, i niech w będzie obliczeniem w $L^{\infty}(S)$. Wtedy $\sigma=[w]$ jest śladem w $[L^{\infty}(S)]$, całkowicie zawartym (jako zbiór) w $L^{\infty}(S)$.

Proces σ jest:

- *uniwersalnie superuczciwy* $\Leftrightarrow$ $(\forall u \in \sigma)$ u jest obliczeniem superuczciwym;
- *egzystencjalnie superuczciwy* $\Leftrightarrow$ $(\exists u \in \sigma)$ u jest obliczeniem superuczciwym;
- *uniwersalnie uczciwy* $\Leftrightarrow$ $(\forall u \in \sigma)$ u jest obliczeniem uczciwym;
- *egzystencjalnie uczciwy* $\Leftrightarrow$ $(\exists u \in \sigma)$ u jest obliczeniem uczciwym;
- *uniwersalnie sprawiedliwy* $\Leftrightarrow$ $(\forall u \in \sigma)$ u jest obliczeniem sprawiedliwym;
- *egzystencjalnie sprawiedliwy* $\Leftrightarrow$ $(\exists u \in \sigma)$ u jest obliczeniem sprawiedliwym.

Klasy procesów danego systemu S zdefiniowane wyżej będą oznaczane odpowiednio przez uSFAIR_S, eSFAIR_S, uFAIR_S, eFAIR_S, uJUST_S i eJUST_S. Gdy nie prowadzi to do niejednoznaczności, indeks S będzie pomijany.

Dla dowolnego systemu tranzycyjnego S prawdziwe są następujące inkluzje:

$$\begin{array}{ccccc} \text{eSFAIR} & \subseteq & \text{eFAIR} & \subseteq & \text{eJUST} \\ \cup\!| & & \cup\!| & & \cup\!| \\ \text{uSFAIR} & \subseteq & \text{uFAIR} & \subseteq & \text{uJUST} \end{array}$$

5.2 Hierarchia uczciwości procesów sieci Petriego

Teraz pokażemy, jak wyglądają zależności pomiędzy tak zdefiniowanymi klasami uczciwości dla procesów generowanych przez elementarne i markowane sieci Petriego.

5.2.1 Procesy superuczciwe

Udowodnimy, że dwie klasy obliczeń superuczciwych są równe w dowolnym systemie tranzycyjnym, więc w szczególności także w dowolnej sieci Petriego.

Stwierdzenie 5.2. W dowolnym systemie tranzycyjnym uSFAIR = eSFAIR.

Dowód. Z Definicji 5.1, uSFAIR $\subseteq$ eSFAIR. Pokażemy inkluzję odwrotną. Niech x będzie nieskończonym superuczciwym obliczeniem systemu S. Przypuśćmy, że jego równoważnik $y \approx x$ nie jest superuczciwy. Wtedy istnieje akcja a, żywa w y, która pojawia się w y tylko n razy. Zatem a pojawia się w x też tylko n razy. Z definicji 2.22, dla każdego skończonego prefiksu u obliczenia x ($u \leq^{fin} x$) istnieje skończony prefiks v obliczenia y ($v \leq^{fin} y$) taki, że $v \approx uw$ dla pewnego w. Ponadto akcja a jest żywa w $q_0(v)=q_0(uw)$, więc a jest żywa $q_0(u)$. Stąd a jest żywa w x i wystepuje w x tylko n razy, zatem x nie jest superuczciwe. Sprzeczność. ¤

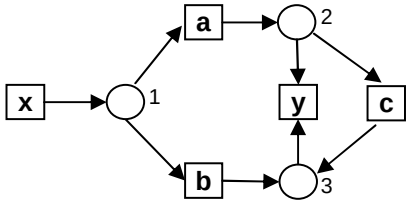
Wobec Stwierdzenia 5.2, klasa uSFAIR = eSFAIR będzie zapisywana jako SFAIR.

5.2.2 Procesy egzystencjalnie i uniwersalnie uczciwe

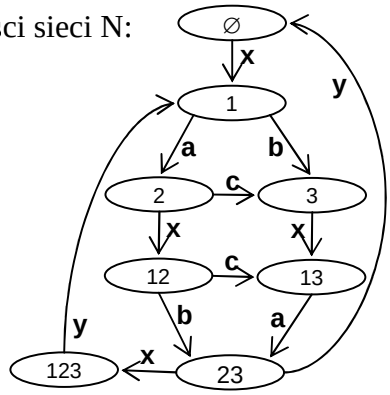
Z Definicji 5.1 i Stwierdzenia 5.2 mamy SFAIR $\subseteq$ uFAIR $\subseteq$ eFAIR. Następujące przykłady pokazują, że obie inkluzje są ostre, nawet w klasie sieci elementarnych:

Przykład 5.3. *Sieć elementarna N, w której* $SFAIR_N \subsetneq uFAIR_N$

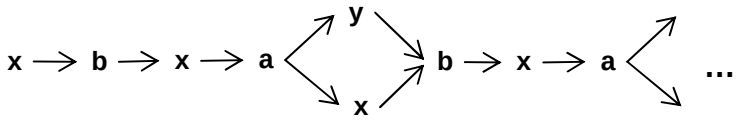
Sieć N:



Graf osiągalności sieci N:



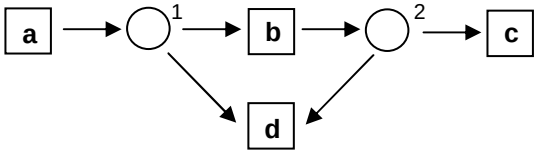
Graf śladu $[(xbxay)^{\omega}]$:



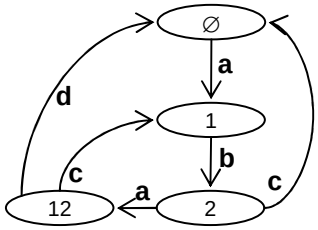
Rys. 37. Proces $[(xbxay)^{\omega}]$ jest uniwersalnie uczciwy, ale nie superuczciwy

Przykład 5.4. *Sieć elementarna N, w której* $uFAIR_N \subsetneq eFAIR_N$

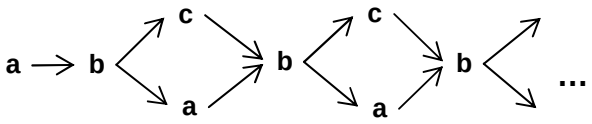
Sieć N:



Graf osiągalności sieci N:



Graf śladu $[(abc)^{\omega}]$:



Rys. 38. Proces $[(abc)^{\omega}]=[ab(acb)^{\omega}]$ jest egzystencjalnie uczciwy, ale nie uniwersalnie uczciwy

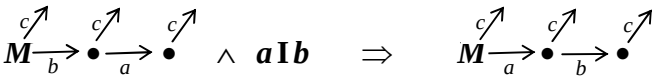
5.2.3 Procesy egzystencjalnie i uniwersalnie sprawiedliwe

Poniższy lemat sformułowany jest w języku sieci Petriego, ale zachodzi dla dowolnych systemów tranzycyjnych.

Lemat 5.5. Jeśli dla dowolnego stanu osiągalnego M i dowolnych akcji a, b, c zachodzi implikacja $(Mc \wedge Mbc \wedge Mbac \wedge aIb) \Rightarrow Mac$, to każdy egzystencjalnie sprawiedliwy proces jest uniwersalnie sprawiedliwy.

Graficzne sformułowanie:

Jeśli

$(\forall M)\,(\forall a,b,c)$  to

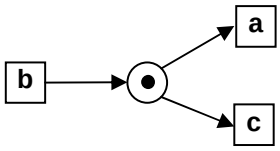
$\text{eJUST} \subseteq \text{uJUST}$.

Dowód. Pokażemy, że wszystkie równoważniki obliczenia sprawiedliwego są sprawiedliwe. Dla obliczeń skończonych jest to oczywiste. Niech x będzie nieskończonym obliczeniem sprawiedliwym i niech $y \approx x$. Niech akcja $c \in A$ będzie stale umożliwiona w y (jeśli taka akcja nie istnieje, to oczywiście y jest sprawiedliwe). Niech teraz y' będzie takim ogonem y ($y=vy'$), że c jest umożliwione w każdym stanie y'. Ponieważ $y \approx x$, to z Definicji 2.22 istnieje skończony prefiks u obliczenia x (tzn. $x=ux'$) taki, że $u \approx vw$ dla pewnego w. Stąd $x \approx vwx'$, zatem $y' \approx wx'$ i oczywiście wx' jest sprawiedliwe. Teraz z założenia lematu stwierdzamy, że c jest umożliwione w każdym stanie obliczenia wx'. Ponieważ wx' jest sprawiedliwe, c pojawia się nieskończenie często w wx', więc także w $x=ux'$, więc także w y, bo $y \approx x$. ¤

Stwierdzenie 5.6. W elementarnych sieciach Petriego każdy proces egzystencjalnie sprawiedliwy jest uniwersalnie sprawiedliwy: uJUST = eJUST.

Dowód. Najpierw zauważmy, że w sieciach elementarnych, jeśli $(\exists M)\ Ma \wedge Mc \wedge \neg Mac$, to $(\forall M)\ Ma \wedge Mc \Rightarrow \neg Mac$. Zatem jeśli $\neg Mac$, to także $\neg Mbac$. Stąd z Lematu 5.5, otrzymujemy uJUST = eJUST. ¤

Przykład 5.7. *Dowód Stwierdzenia 5.6 nie działa dla sieci markowanych*



Rys. 39. Sieć markowana. Zachodzi $M_0a \wedge M_0c \wedge \neg M_0ac$,
ale dla $M=M_0b$ mamy $Ma \wedge Mc \wedge Mac$

Niemniej jednak, równość uJUST = eJUST zachodzi dla bezpętelkowych sieci markowanych, ale dowód tego faktu jest nieco bardziej złożony.

Stwierdzenie 5.8. W bezpętelkowych sieciach markowanych uJUST = eJUST.

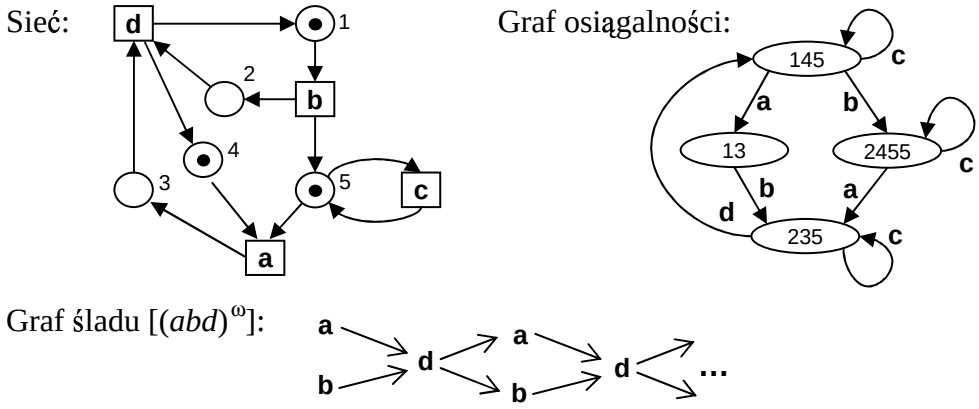
Dowód. Udowodnimy, że założenie Lematu 5.5 zachodzi dla bezpętelkowych sieci markowanych. Przytoczmy to założenie w wygodniejszej postaci:

$$(\forall a,b\in A)\ ((\exists M)\ (\exists c)\ Mc \wedge Mbc \wedge Mbac \wedge Ma \wedge \neg Mac) \Rightarrow a\mathrm{D}b$$

Pokażemy, że jeśli $Mc \wedge Mbc \wedge Mbac \wedge Ma \wedge \neg Mac$, to $\neg Mcab \wedge Mcba$.
Nazwijmy kilka potrzebnych stanów: McM_1, MbM_2cM_3, MbM_2ac.
Mcba: Mamy M_2c i M_2ac, więc (z Faktu 2.4) MbM_2cM_3a; podobnie ponieważ MbM_2cM_3 i McM_1, to (z Faktu 2.4) McM_1bM_3. Zatem McM_1bM_3a.
¬Mcab: Mamy *Ma*, *Mc* i *¬Mac*, zatem (z Faktu 2.4), *¬Mca*, więc także *¬Mcab*.
Pokazaliśmy, że $Mcba \wedge \neg Mcab$, zatem $M_1ba \wedge \neg M_1ab$, czyli $\neg a\mathrm{I}b$. ¤

W dowodzie Stwierdzenia 5.8 używaliśmy założenia, że sieć jest bezpętelkowa. Następujący przykład pokazuje, że było to konieczne:

Przykład 5.9. *Stwierdzenie 5.8 nie zachodzi dla sieci markowanych z pętelkami*



Rys. 40. Proces $[(abd)^{\omega}]=[(bad)^{\omega}]$ jest egzystencjalnie sprawiedliwy, ale nie uniwersalnie

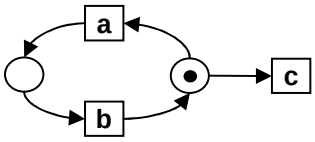
5.2.4 Podsumowanie

Podsumujmy wyniki i przykłady tego podrozdziału. Hierarchiczna struktura procesów z punktu widzenia ich etycznych właściwości wygląda następująco:

$$\begin{array}{ccccc} & & \text{eFAIR} & \overset{(4)}{\subseteq} & \text{eJUST} \\ & \overset{(2)}{\cup} & & & \cup \overset{(5)}{} \\ \text{SFAIR} & \underset{(1)}{\subseteq} & \text{uFAIR} & \underset{(3)}{\subseteq} & \text{uJUST} \end{array}$$

Dla każdej inkluzji powyższego diagramu istnieje przykład sieci markowanej, dla której jest ona ostra: dla inkluzji (1) – Przykład 5.3, dla (2) – Przykład 5.4, dla (3) i (4) – Przykład 5.10 poniżej, dla (5) – Przykład 5.11.

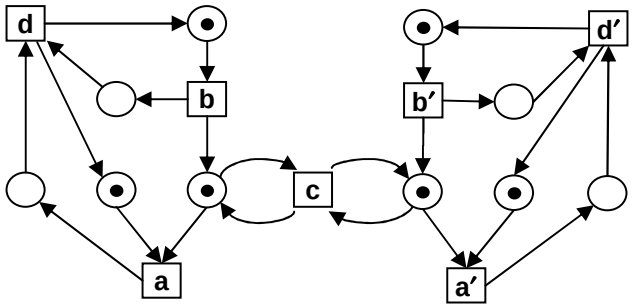
Przykład 5.10. *Sieć elementarna, w której uFAIR $\subsetneq$ uJUST i eFAIR $\subsetneq$ eJUST*



Rys. 41. Proces $[(ab)^{\omega}]$ jest uJUST, więc eJUST, ale nie eFAIR, więc też nie uFAIR

Klasy eFAIR i uJUST są nieporównywalne. Przykład 5.10 pokazuje przypadek procesu uJUST, ale nie eFAIR, przypadek procesu eFAIR ale nie uJUST jest podany poniżej:

Przykład 5.11. *Sieć markowana z procesem eFAIR, który nie jest uJUST*



Rys. 42. Obliczenie $aa'(bdab'd'a')^{\omega}$ jest uczciwe,
ale jego równoważnik $(bb'aa'dd')^{\omega}$ nie jest sprawiedliwy

Dla sieci elementarnych i markowanych bezpętelkowych hierarchia staje się liniowa:

$$\text{SFAIR} \subseteq \text{uFAIR} \subseteq \text{eFAIR} \subseteq \text{eJUST} = \text{uJUST}$$

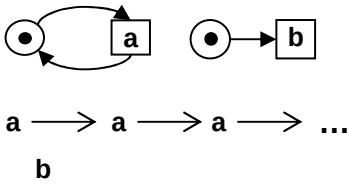
Dla każdej inkluzji istnieje sieć, nawet elementarna, dla której inkluzja jest ostra. Są to odpowiednio Przykłady 5.3, 5.4, i 5.10. Trzecia inkluzja wynika z faktu, że każde uczciwe obliczenie jest sprawiedliwe, a równość – ze Stwierdzeń 5.6 i 5.8.

5.3 Etyka w podejściu nieprzeplotowym

Potrzebę nieprzeplotowego podejścia do definiowania procesów uczciwych zasygnalizowała Kwiatkowska [27]. Zauważmy, że próba bezpośredniego przeniesienia sekwencyjnej definicji uczciwości (jeśli *a* umożliwione nieskończenie często, to wykonane nieskończenie często) nie jest wystarczająca dla śladów, co pokazuje poniższy przykład.

Przykład 5.12.

Rozważmy następującą sieć składającą się z dwóch niezależnych akcji:



Proces $[ba^{\omega}]$ tej sieci jest w naturalny sposób uczciwy, a nawet superuczciwy – wszystko, co jest umożliwione, jest wykonane.

Przy bezpośrednim przeniesieniu definicji uczciwości proces ten byłby nieuczciwy, gdyż *b* jest umożliwione po każdym prefiksie a^n tego procesu, a wykonane tylko raz.

Podamy teraz nieprzeplotowe definicje pojęć etycznych (istotnie różne od [27]) i pokażemy, że klasy uczciwości zdefiniowane w ten sposób pokrywają się z pewnymi klasami uczciwości zdefiniowanymi przeplotowo w podrozdziale 5.1. Nieprzeplotowe definicje nie wnoszą więc nic nowego, jednak okażą się przydatne jako narzędzie dowodowe, np. w dowodzie twierdzenia 5.22.

Definicja 5.13. *Procesy sprawiedliwe, uczciwe i superuczciwe (wersja nieprzeplotowa)*

Niech $S = (A, Q, q_0)$ będzie systemem tranzycyjnym, I – niezależnością indukowaną przez *S*, i niech *w* będzie obliczeniem w $L^{\infty}(S)$. Wtedy $\alpha=[w]$ jest śladem w $T=[L^{\infty}(S)]$.

Proces $\alpha \in T$ jest:

- *η-superuczciwy* $\Leftrightarrow$ $(\forall a \in A)\ |\alpha|_a<\infty \Rightarrow (\exists \beta \leq^{fin} \alpha)\ (\forall \gamma)\ \beta\gamma[a] \notin T$
  tzn. *a* nie jest żywe po β;
- *η-uczciwy* $\Leftrightarrow$ $(\forall a \in A)\ |\alpha|_a<\infty \Rightarrow (\exists \beta \leq^{fin} \alpha)\ (\forall \gamma)\ \beta\gamma \leq^{fin} \alpha \Rightarrow \beta\gamma[a] \notin T$
  tzn. *a* nie jest nigdzie umożliwione w α po β;
- *η-sprawiedliwy* $\Leftrightarrow$ $(\forall a \in A)\ |\alpha|_a<\infty \Rightarrow (\forall \beta \leq^{fin} \alpha)\ (\exists \gamma)\ \beta\gamma \leq^{fin} \alpha \wedge \beta\gamma[a] \notin T$
  tzn. *a* nie jest stale umożliwione w żadnym ogonie śladu α.

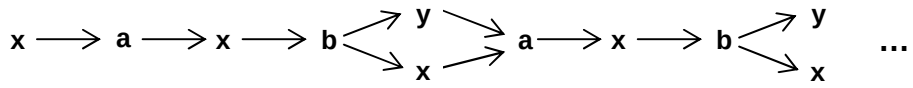
Klasy procesów systemu *S* zdefiniowane powyżej, będą oznaczane przez ηSFAIR_S, ηFAIR_S i ηJUST_S, odpowiednio. Jeśli nie prowadzi to do niejednoznaczności, indeksy *S* będą pomijane. Następujące zawierania wynikają wprost z definicji:

$$\eta\text{SFAIR} \subseteq \eta\text{FAIR} \subseteq \eta\text{JUST}.$$

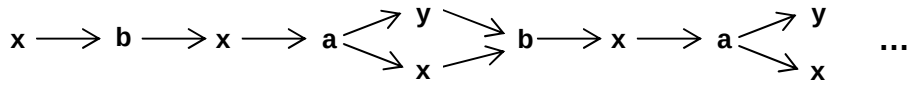
Przykład 5.14.

Rozważmy sieć z Przykładu 5.3.

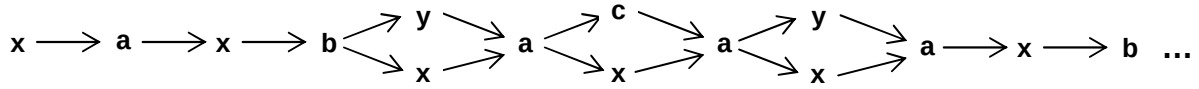
$\alpha=[xa(xbxya)^{\omega}]\in \eta$JUST, ale $\alpha\notin \eta$FAIR



$\beta=[xb(xaxyb)^{\omega}]\in \eta$FAIR, ale $\beta\notin \eta$SFAIR



$\gamma=[x(axbxyaxcaxy)^{\omega}] \in \eta$SFAIR



Stwierdzenie 5.15. W dowolnym systemie tranzycyjnym ηSFAIR = SFAIR.

Dowód. ($\subseteq$) Niech $\alpha\in \eta$SFAIR i *a* niech będzie akcją skończenie wiele razy występującą w α. Istnieje skończony ślad β taki, że $\alpha=\beta\gamma$ i w γ nie ma stanu umożliwiającego akcję *a*. Niech *u* będzie dowolną linearyzacją β, a *w* – dowolną linearyzacją γ: $[u]=\beta$, $[w]=\gamma$. Wtedy *uw* jest superuczciwą linearyzacją α, co wobec Stwierdzenia 5.2 wystarcza do pokazania superuczciwości α.

($\supseteq$) Niech $\alpha\in$ SFAIR i niech *w* będzie dowolną jego (superuczciwą) linearyzacją: $[w]=\alpha$. Istnieją takie słowa *u*, *v*, że *w*=*uv*, *u* jest skończone i akcja *a* występująca skończoną ilość razy w α nie jest żywa w stanie $q_0(u)$, a zatem także w żadnym stanie obliczenia *v*, więc również w żadnym stanie dowolnego równoważnika obliczenia *v*. Szukanym prefiksem jest zatem $\beta=[u]$. ¤

Stwierdzenie 5.16. W dowolnym systemie tranzycyjnym ηJUST = eJUST.

Dowód. ($\subseteq$) Niech $\alpha\in \eta$JUST i niech $\alpha=\beta\gamma$, gdzie β jest skończonym prefiksem α zawierającym wszystkie akcje $a_1, a_2, ..., a_n$ pojawiające się skończoną ilość razy w α, a ogon γ zawiera tylko takie akcje $b_1, b_2, ..., b_m$, które pojawiają się nieskończenie często w α. Niech *u* będzie dowolną linearyzacją β, czyli $\beta=[u]$. Konstruujemy sprawiedliwą linearyzację α w następujący sposób: $w_0=u$, $w_i=w_{i-1}v_i$, dla i=1,...*n*, gdzie wszystkie b_j (j=1,…,m) występują w v_i i stan po v_i nie umożliwia a_i, a następnie ponownie od a_1 do a_n i tak dalej. Taka nieskończona procedura jest możliwa, ponieważ każdy ogon zawiera stan nie umożliwiający a_i. Nieskończone obliczenie $w = uv_1\ldots v_n v'_1\ldots v'_n\ldots$ jest sprawiedliwą linearyzacją α.

($\supseteq$) Niech $\alpha\in$ eJUST i niech *w* będzie sprawiedliwą linearyzacją α. Przypuśćmy, że α nie jest ηJUST, zatem istnieje akcja *a*, skończony prefiks β i ogon γ takie, że $|\alpha|_a<\infty$, $\alpha=\beta\gamma$, i *a* jest stale umożliwione w γ. Niech $\beta=[u]$, $\gamma=[v]$, zatem *a* jest stale umożliwione w *v* i $|v|_a<\infty$. Stąd $[w]=[uv]$ i *w* jest sprawiedliwe, ale *uv* nie jest. Z Definicji 2.22,

istnieje rozkład $w=zy$ taki, że $[z]=[ux]$ dla pewnego x. Ponieważ $[w]=[zy]=[uxy]=[uv]$, to $[xy]=[v]=\gamma$. Ale a jest stale umożliwione w $\gamma=[v]=[xy]$, zatem także w y. Ale $w=zy$ i $|w|_a=|\alpha|_a<\infty$, stąd w nie jest sprawiedliwe. Sprzeczność. ¤

Stwierdzenie 5.17. W dowolnym systemie tranzycyjnym ηFAIR = uFAIR.
Dowód. ($\subseteq$) Niech $\alpha\in\eta$FAIR i niech $w\in\alpha$ będzie dowolną linearyzacją α. Niech a będzie akcją nieskończenie często umożliwioną w obliczeniu w. Ponieważ stany w są jednocześnie stanami α, proces α ma nieskończenie wiele stanów umożliwiających a. Zatem każdy ogon α ma nieskończenie wiele takich stanów. Ponieważ α jest ηFAIR, to akcja a występuje nieskończenie często w α, więc także w w, zatem w jest uczciwe. Ponieważ w było dowolną linearyzacją α, proces α jest uFAIR.
($\supseteq$) Niech $\alpha\in$ uFAIR i przypuśćmy, że α nie jest ηFAIR, zatem istnieje taka akcja a, że $|\alpha|_a<\infty$ i każdy ogon α zawiera stan umożliwiający a. Konstruujemy linearyzację jak w dowodzie Stwierdzenia 5.16($\subseteq$), kończąc każde v_i stanem umożliwiającym a. Jest to możliwe, ponieważ każdy ogon zawiera taki stan. Taka linearyzacja nie jest uczciwa, zatem α nie jest uFAIR. Sprzeczność. ¤

Możemy więc uzupełnić diagram hierarchii etycznej procesów następująco:

$$\begin{array}{ccccc} & & \text{eFAIR} & \subseteq & \text{eJUST} = \eta\text{JUST} \\ & & \cup\!| & & \cup\!| \\ \text{SFAIR} & \subseteq & \text{uFAIR} & \subseteq & \text{uJUST} \\ \|\! & & \|\! & & \\ \eta\text{SFAIR} & & \eta\text{FAIR} & & \end{array}$$

5.4 Kryteria równości eFAIR=uFAIR i uFAIR=SFAIR

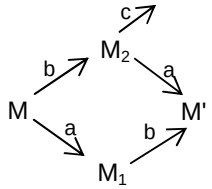
Sytuacja, w której proces egzystencjalnie uczciwy nie jest uniwersalnie uczciwy, wprowadza pewną trudność w badaniu i projektowaniu systemów współbieżnych, ponieważ sprawdzenie uczciwości pojedynczego obliczenia nie zapewnia uczciwości całego procesu. Pojawia się pytanie o to, w jaki sposób zapewnić równoważnościową domkniętość procesów ze względu na uczciwość, czyli jak projektować systemy, w których każdy proces egzystencjalnie uczciwy jest też uniwersalnie uczciwy. Nazwijmy takie procesy stabilnie uczciwymi. W tym podrozdziale podamy kryteria rozstrzygające, czy w danym systemie skończenie stanowym zachodzi eFAIR=uFAIR oraz uFAIR=SFAIR.

Przypuszczenie, że występowanie niestabilnie uczciwych procesów ma jakiś związek z konfuzjami, pojawiło się u Kwiatkowskiej [27]. Przytoczmy najpierw definicję konfuzji.

Definicja 5.18. *Konfuzja*
Niech M będzie stanem pewnego systemu tranzycyjnego, $a,b\in A$ – jego akcjami. Trójka (M,a,b) jest *konfuzją*, jeśli a, b są niezależne, umożliwione w M oraz zbiór akcji skonfliktowanych z akcją a w stanie M jest inny niż w stanie Mb:

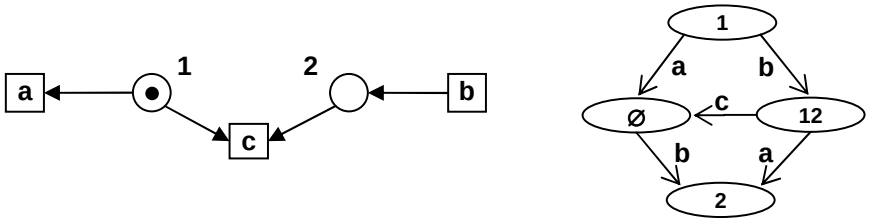
$$a\mathrm{I}b \wedge MaM_1 \wedge MbM_2 \wedge \{c\in A;\, M[a\#c]\}\neq\{c\in A;\, M_2[a\#c]\}.$$

Na pewno istnienie procesu niestabilnego wymaga istnienia konfuzji. Będzie ona miała pewną szczególną postać: w stanie *M* i na ścieżce *ab* akcja *c* nie jest umożliwiona, natomiast na ścieżce *ba* pojawia się stan M_2, w którym akcja *c* jest umożliwiona – będzie to ta akcja, która nie jest wykonana w obliczeniu i która sprawia, że równoważnik zawierający ścieżkę *ba* jest nieuczciwy.



Jednak samo istnienie takiej konfuzji nie wystarcza, aby istniał proces niestabilny:

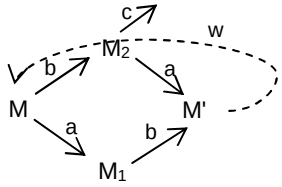
Przykład 5.19. *Sieć z konfuzją, ale eFAIR=uFAIR*



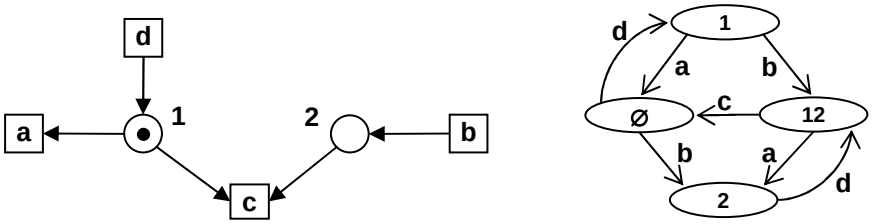
Rys. 43. Sieć elementarna z konfuzją w stanie {1} i wszystkimi procesami superuczciwymi

W Przykładzie 5.19 wszystkie obliczenia są skończone, więc wszystkie obliczenia nieprzedłużalne są uczciwe. Potrzebne jest dodatkowe założenie „ożywiające" konfuzję – musi istnieć ścieżka ze stanu *M*' do *M*.

Jednak okazuje się, że to założenie również nie jest wystarczające:



Przykład 5.20. *Sieć z żywą konfuzją, ale eFAIR=uFAIR*



Rys. 44. Sieć elementarna z żywą konfuzją w stanie {1} i z eFAIR=uFAIR

W Przykładzie 5.20 nieskończone obliczenie zawierające podsłowo *ba* musi zawierać wszystkie litery, a więc nie da się skonstruować nieuczciwego obliczenia zawierającego *ba*. Musimy narzucić dodatkowe założenie na ścieżkę *w* „ożywiającą" konfuzję – nie może ona zawierać litery *c* (aby obliczenie $(baw)^{\omega}$ było nieuczciwe). Może się także zdarzyć, że na ścieżce *w* umożliwiona jest jakaś akcja nie wykonana (*c* lub inna) – co powoduje, że obliczenie $(abw)^{\omega}$ jest także nieuczciwe. Będzie to ostatni warunek, który trzeba nałożyć na ścieżkę *w*.

Niech M będzie stanem pewnego systemu, a $w \in A^*$ skończonym obliczeniem tego systemu. Jeśli MwM, to taką pętlę nazywamy egzystencjalnie (uniwersalnie) uczciwą wtedy i tylko wtedy, gdy proces $[w^\omega]$ jest egzystencjalnie (uniwersalnie) uczciwy.

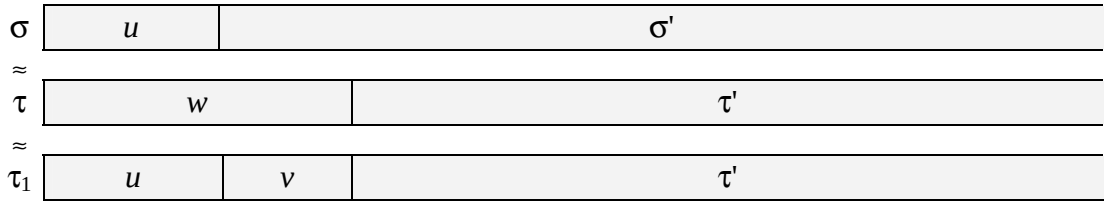
Twierdzenie 5.21. W systemach skończenie stanowych

eFAIR=uFAIR $\Leftrightarrow$ każda pętla egzystencjalnie uczciwa jest uniwersalnie uczciwa

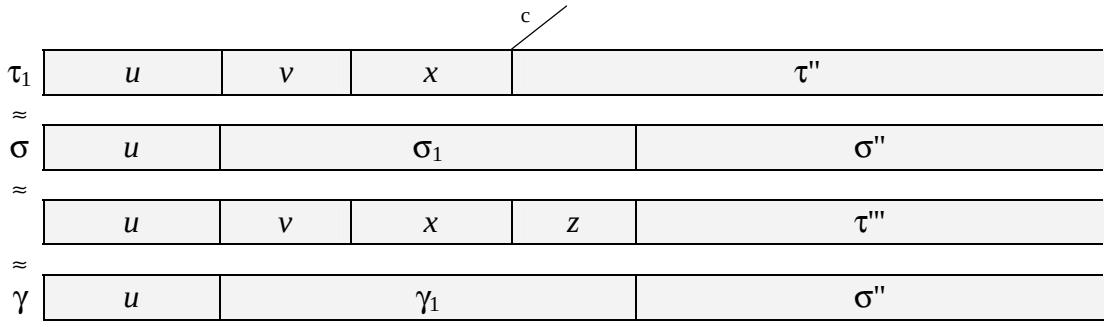
Dowód. ($\Rightarrow$) Przypuśćmy, że istnieje pętla MwM egzystencjalnie uczciwa, która nie jest uniwersalnie uczciwa. Wtedy proces $[w]^\omega$ jest egzystencjalnie uczciwy, ale nie uniwersalnie. Zatem eFAIR≠uFAIR.

($\Leftarrow$) Przypuśćmy, że eFAIR≠uFAIR i niech σ będzie obliczeniem uczciwym, a τ jego równoważnikiem nieuczciwym: $\tau \approx \sigma$. Na podstawie obliczenia σ skonstruujemy takie obliczenie γ, w którym znajdziemy ścieżkę abw taką, że $aIb \wedge MaM_1bM' \wedge MbM_2aM' \wedge M_2c \wedge M'wM \wedge |w|_c=0 \wedge$ obliczenie $(abw)^\omega$ jest uczciwe.

Niech σ' będziem takim ogonem σ, w którym wszystkie litery występują nieskończenie często $\sigma=u\sigma'$ dla pewnego skończonego u i niech Z będzie zbiorem tych liter, które występują w σ'. Z definicji 2.22 (równoważności obliczeń nieskończonych) istnieje skończony prefiks w obliczenia τ ($\tau=w\tau'$) taki, że $w \approx uv$ dla pewnego v. Oczywiście skoro $\tau=w\tau'$ to $\tau \approx uv\tau' \approx u\sigma'$, stąd $v\tau' \approx \sigma'$.



W τ' istnieje nieskończenie wiele stanów, które umożliwiają jakąś literę spoza zbioru Z. Niech x będzie najkrótszym prefiksem τ', takim, że jego stan końcowy umożliwia literę spoza Z – nazwijmy ją c, czyli $\tau'=x\tau''$ i c jest umożliwione po x.



Z definicji 2.22 istnieje taki prefiks σ_1 obliczenia σ' ($\sigma'=\sigma_1\sigma''$), że $\sigma_1 \approx vxz$ dla pewnego z, czyli vxz jest takim równoważnikiem słowa σ_1, którego stan wewnętrzny umożliwia c. Są to słowa skończone, zatem można otrzymać jedno (vxz) z drugiego (σ_1) poprzez skończoną ilość pojedynczej zamiany miejscami pary liter niezależnych. Rozważmy

taki proces – w pewnym momencie jakaś zamiana powoduje, że w nowym obliczeniu pojawia się stan umożliwiający literę spoza Z (w „najgorszym" przypadku będzie to ostatnia zamiana, a umożliwioną literą będzie c, ale może wystąpić wcześniej taka sytuacja i może to być inna litera). Zatrzymujemy się w momencie, gdy zajdzie taka sytuacja i niech γ_1 będzie otrzymanym słowem. Słowo γ_1 zawiera podsłowo a_1b_1, takie, że $a_1 I b_1$ i po a_1 jest umożliwiona litera spoza Z, a wszystkie inne stany pośrednie nie umożliwiają litery spoza Z.

W τ''' nadal istnieje stan umożliwiający literę spoza Z, zatem w taki sam sposób konstruujemy odpowiedni równoważnik γ_2 prefiksu słowa σ'', i tak dalej.

Utworzyliśmy nieskończone obliczenie $\gamma = u\gamma_1\gamma_2\ldots$ Każde γ_i zawiera podsłowo a_ib_i takie, że $a_i I b_i$ i stan po a_i jest (jedynym w γ_i) stanem umożliwiającym literę c_i spoza Z. Stan przed a_i nazwijmy M_i. Ponieważ stanów i liter jest skończenie wiele, istnieje taka czwórka (M_i, a_i, b_i, c_i), która się powtarza nieskończenie często. Możemy więc wybrać γ_k i γ_n takie, że $M_k=M_n=M$, $a_k=a_n$, $b_k=b_n$, $c_k=c_n$ oraz w słowie pomiędzy M_k a M_n (nazwijmy je w) pojawiają się wszystkie litery ze zbioru Z. W słowie w zamieniamy wszystkie podsłowa a_ib_i na b_ia_i otrzymując słowo w'. Pętla $Mw'M$ jest uczciwa, ponieważ stany po a_i były jedynymi stanami umożliwiającymi literę spoza Z. ¤

Okazuje się, że analogiczne kryterium można przyjąć do rozstrzygania, czy w danym skończenie stanowym systemie zachodzi równość uFAIR=SFAIR.

Twierdzenie 5.22. W systemach skończenie stanowych

uFAIR=SFAIR $\Leftrightarrow$ każda pętla uniwersalnie uczciwa jest superuczciwa

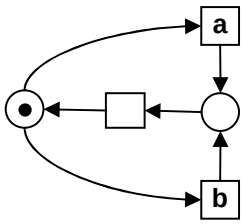
Dowód. ($\Rightarrow$) Przypuśćmy, że istnieje pętla MwM, która jest uniwersalnie uczciwa, ale nie superuczciwa. Stąd proces $[w^\omega]$ jest uniwersalnie uczciwy, ale nie superuczciwy. Zatem uFAIR$\neq$SFAIR.

($\Leftarrow$) Przypuśćmy, że uFAIR$\neq$SFAIR i niech α będzie procesem uniwersalnie uczciwym, ale nie superuczciwym. Z twierdzenia 5.17 α jest procesem η-uczciwym, a to oznacza (z definicji), że dla każdej litery $a\in A$ występującej skończoną ilość razy w α istnieje skończony prefiks β procesu α (tzn. $\alpha=\beta\gamma$ dla pewnego nieskończonego śladu γ), taki, że żaden skończony prefiks procesu α dłuższy od β nie umożliwia litery a (tzn. jeśli $\beta\delta\leq^{fin}\alpha$ to $\beta\delta[a]$ nie jest procesem badanego systemu). Teraz wybierzmy najdłuższy spośród wszytkich takich prefiksów (istnieje, ponieważ ilość liter jest skończona), tzn. $\alpha=\beta\gamma$ i żaden prefiks dłuższy od β nie umożliwia żadnej spośród liter występujących skończoną ilość razy w α. Niech $w\in\gamma$ będzie dowolną linearyzacją γ. Ponieważ system ma skończoną ilość stanów, to istnieje w obliczeniu w taki stan M, który pojawia się w nim nieskończenie często. Zatem można podzielić w na takie słowa w_1, w_2, w_3, że $w=w_1Mw_2Mw_3$ (w_1 prowadzi do stanu M i w_1w_2 prowadzi do M) oraz w słowie w_2 występują wszystkie litery obliczenia w. Wówczas Mw_2M jest szukaną pętlą: proces $[w_2^\omega]$ jest uniwersalnie uczciwy, ponieważ żaden prefiks γ (a więc także w_2) nie umożliwia litery nie wykonanej, a nie jest superuczciwy, bo w γ jest żywa jakaś akcja nie wykonana. ¤

6 Podsumowanie

Myślą przewodnią rozprawy jest zastosowanie teorii śladów do opisu i badania zachowań sieci Petriego. Wiemy, że ślady idealnie opisują zachowanie sieci elementarnych. Zachowania sieci markowanych (i ich rozszerzeń) były dotychczas modelowane innymi obiektami (np. półśladami, słowami częściowymi, sieciami wystąpień). Istnieją jednak sieci markowane, których zachowanie jest w pełni opisywalne za pomocą śladów – nazwaliśmy je sieciami o zachowaniu śladowym. Ślady (tak skończone, jak i nieskończone) są klasami równoważności, mają więc rozłączne zbiory linearyzacji. Badanie własności procesów reprezentowanych przez ślady możemy więc sprowadzić do badania ich pojedynczych przebiegów sekwencyjnych (linearyzacji).

Wykorzystanie teorii śladów do badania zachowania sieci Petriego, nie tylko elementarnych, ale także markowanych, a nawet ich rozszerzeń, wymaga odpowiedniego zdefiniowania alfabetu współbieżnego dla sieci, tak, aby ślady w pełni odzwierciedlały naturę procesów sieci. Zaproponowana tutaj ogólna definicja niezależności akcji powoduje, że indukowane przez nią ślady nie zawsze przechowują pełną informację o zachowaniach współbieżnych. Jednak narzucając zakaz współbieżnego wykonywania akcji zależnych, co da się technicznie zrealizować prostą konstrukcją uzupełniającą (Rys. 48 poniżej), otrzymujemy sieć o zachowaniu śladowym symulującą działanie sieci pierwotnej.



Rys. 45. Konstrukcja uzupełniająca każdą parę *a,b* akcji zależnych (i współbieżnych)

Obliczalność relacji zależności w sieciach markowanych (Twierdzenie 4.6) pozwala na dokładne (tzn. tam, gdzie trzeba) wykonanie powyższej konstrukcji. Można powiedzieć, że tak zbudowana sieć (jej zachowanie jest śladowe) symuluje działanie sieci pierwotnej, tracąc możliwość współbieżnego wykonania pewnych (ale tylko zależnych) akcji.

Sieci markowane możemy więc traktować – podobnie jak sieci elementarne – jako generatory języków śladów. Oczywiście klasa języków śladów przez nie generowanych jest istotnie większa od klasy języków generowanych przez sieci elementarne.

Relacja niezależności akcji okazała się nieobliczalna w pewnych rozszerzeniach sieci markowanych. Dowody oparte są na nierozstrzygalności problemu pustości warunku w tych klasach sieci, co również pokazano w rozprawie.

Analiza rozstrzygalności problemów zależności akcji i zachowania śladowego w sieciach rozszerzonych doprowadziła przy okazji do bardzo ciekawego nowego problemu, z początku sprawiającego wrażenie oczywistości. We wszystkich znanych klasach sieci problemy osiągalności i osiągalności stanu z pustym warunkiem są albo rozstrzygalne, albo nierozstrzygalne. Nie istnieje jednak ogólny dowód równoważności tych problemów w dowolnej klasie sieci Petriego.

Problem: Czy istnieje taka klasa sieci, w której problem osiągalności jest nierozstrzygalny, a problem pustości warunku – rozstrzygalny?

Śladowy model zachowania sieci umożliwił precyzyjne uogólnienie klasycznych pojęć sekwencyjnej uczciwości, odpowiadające zjawiskom zachodzącym w systemach współbieżnych. Podaliśmy ogólną klasyfikację i hierarchię tych pojęć (dla dowolnych systemów tranzycyjnych), a następnie szczegółowo zbadaliśmy je w podstawowych klasach sieci Petriego – sieciach elementarnych i markowanych. Sformułowaliśmy również nieprzeplotowe definicje najważniejszych pojęć etyki obliczeń i pokazaliśmy, że nie wnoszą one nic nowego w porównaniu z definicjami przeplotowymi.

Interesującym problemem wydaje się znalezienie ogólnych kryteriów, warunkujących równość pewnych klas uczciwości. W rozprawie takie kryteria znaleziono jedynie dla systemów tranzycyjnych skończenie stanowych.

W pracy przedstawiono również pewne wyniki związane z zagadnieniem unikania konfliktów. Opracowano algorytm produkujący graf sterujący bezkonfliktowym działaniem sieci. Algorytm ten działa dla tzw. sieci bezpiecznych – rozszerzenia sieci elementarnych o inne typy łuków. Nasuwa się możliwość dwóch nurtów dalszych badań związanych z konfliktami: unikanie konfliktów w sieciach markowanych (i ich rozszerzeniach) oraz współbieżne podejście do konfliktów (procesy z bezkonfliktową linearyzacją).

Bibliografia

1. T. Araki, T. Kasami: Some Decision Problems Related to the Reachability Problem for Petri Nets. Theoretical Computer Science, 3(1):85-104, 1977
2. E. Badouel, Ph. Darondeau: Trace Nets and Process Automata. Acta Informatica 32, pp. 647-679, 1995.
3. E. Best: Fairness and Conspiracies. Information Processing Letters 18, pp. 215-220, 1984. Erratum: IPL 19, p.162, 1984.
4. E. Best, R. Devillers: Sequential and Concurrent Behaviour in Petri Net Theory. Theoretical Computer Science 55, pp. 87-136, 1987.
5. N. Busi, R. Gorrieri: Positive Non-interference in Elementary and Trace Nets. LNCS 3099, pp. 1-16, 2004.
6. P. Cartier, D. Foata: Problèmes Combinatoires de Communication et Réarrangements. Lecture Notes in Mathematics 85, Springer, Berlin-Heidelberg-New York 1969.
7. P. Chrząstowski-Wachtel: Testing Undecidability of the Reachability in Petri Nets with the Help of 10th Hilbert Problem. LNCS 1639, pp.268-281. Springer, 1999.
8. R. Cori, D. Perrin: Automates et Commutations Partielles. Informatique Theoretique et Applications 19, pp. 21-32, 1985.
9. J. Desel, W. Reisig: Place/Transition Petri Nets. In [41], LNCS 1491, pp. 122-173. Springer, 1998.
10. V. Diekert: Combinatorics on Traces. LNCS 454. Springer, 1990.
11. V. Diekert: A Partial Trace Semantics for Petri Nets. Theoretical Computer Science 134(1), pp. 87-105, 1994.
12. V. Diekert: On the concatenation of infinite traces. Tech. Rep. TUM, Technische Universität München, 1990.
13. V. Diekert, G. Rozenberg: Book of Traces. World Scientific, 1995.
14. E.W. Dijkstra: Hierarchical Ordering of Sequential Processes. Acta Informatica 1, pp. 115-138. 1971.
15. C. Dufourd, A. Finkel, Ph. Schnoebelen: Reset Nets Between Decidability and Undecidability. LNCS 1443, pp.103-115. Springer, 1998.
16. M.P. Flé, G. Roucairol: Maximal Serializability of Iterated Transactions. Theoretical Computer Science 38, pp. 1-16, 1985.
17. N. Francez: Fairness. Springer, 1986.
18. N. Francez, R.-J. Back, R. Kurki-Suonio: On Equivalence-Completions of Fairness Assumptions. Formal Aspects of Computing 4, pp. 582-591, 1992.
19. P. Gastin: Infinite Traces. Semantics of Systems of Concurrent Processes, pp. 277-308, 1990.
20. P. Gastin, A. Petit: Infinite Traces. In [13], 1995.
21. M.H.T. Hack: Decidability Questions for Petri Nets. Ph. D. Thesis, M.I.T. 1976.
22. H. J. Hoogeboom, G. Rozenberg: Dependence Graphs. In [13], 1995.
23. P.W. Hoogers, H.C.M. Kleijn, P.S. Thiagarajan: A Trace Semantics for Petri Nets. Information and Computation 117(1), pp. 98-114, 1995.
24. J. Jółkowska, E. Ochmański: On Trace-Expressible Behaviour of Petri Nets. Fundamenta Informaticae 85(1-4), pp. 281-295, 2008.

25. R. M. Karp, R. E. Miller: Parallel program schemata. J. Comp. Sys. Sc. 3, pp. 147-195, 1969.
26. S. R. Kosaraju: Decidability of Reachability in Vector Addition Systems. Proceedings of the 14th Annual ACM Symposium on Theory of Computing, 267-281, 1982.
27. M. Kwiatkowska: Defining Process Fairness for Non-Interleaving Concurrency. LNCS 472, pp. 286-300. Springer, 1990.
28. L. Lamport: Fairness and Hyperfairness. Distributed Computing 13(4), pp. 239-245, 2000.
29. D. Lehman, A. Pnueli, J. Stavi: Impartiality, Justice and Fairness: the Ethics of Concurrent Termination. LNCS 115, pp. 264-277. Springer, 1981.
30. Y. Matiyasevich: Enumerable Sets Are Diophantine. Dokl. Akad. Nauk SSSR, 191, pp. 279-282, 1970.
31. E.W. Mayr: An Algorithm for the General Petri Net Reachability Problem. Proceedings of the 13th Annual ACM Symposium on Theory of Computing, 238-246, 1981.
32. A. Mazurkiewicz: Concurrent Program Schemes and their Interpretations. Report DAIMI-PB-78, Aarhus University, 1977.
33. A. Mazurkiewicz: Trace Theory. LNCS 255, pp. 279-324, Springer, 1987.
34. A. Merceron: Fair Processes. In Advances in Petri Nets, LNCS 266, pp. 181-195. Springer, 1987.
35. Ł. Mikulski: Projection Representation of Mazurkiewicz Traces. Fundamenta Informaticae 85(1-4), pp. 399-408, 2008.
36. M. Minsky: Computation: Finite and Infinite Machines. Prentice-Hall, 1967.
37. E. Ochmański, J. Pieckowska: Trace Nets and Conflict-free Computations. Fundamenta Informaticae 72(1-3), pp. 311-321, 2006.
38. E. Ochmański, J. Pieckowska: On Ethics of Mazurkiewicz Traces. Fundamenta Informaticae 80(1-3), pp. 259-272, 2007.
39. D. Peled, A. Pnueli: Proving Partial Order Properties. Theoretical Computer Science 126, pp. 143-182, 1994.
40. C.A. Petri: Kommunikation mit Automaten. Schriften des Institutes für Instrumentelle Mathematik. Bonn, 1962.
41. W. Reisig, G. Rozenberg (eds.): Lectures on Petri Nets. LNCS 1491. Springer, 1998.
42. G. Rozenberg, J. Engelfriet: Elementary Net Systems. In [41], LNCS 1491, pp. 12-121. Springer, 1998.
43. E.W. Stark: Concurrent Transition Systems. TCS 64, pp. 221-269, 1989
44. P. H. Starke: Sieci Petri. PWN, Warszawa 1987.
45. R. Valk: Self-modifying Nets, a Natural Extension of Petri Nets. LNSC 62, pp. 464-476. Springer-Verlag, 1978.
46. H. Völzer: On Conspiracies and Hyperfairness in Distributed Computing. DISC 2005, LNCS 3724, pp. 33-47. Springer, 2005.