Prace Naukowe Instytutu Cybernetyki Technicznej
Politechniki Wrocławskiej
Seria: Monografie

80
15

Jan Magott

Sieci Petriego w ocenie wydajności systemów komputerowych

Wrocław 1989

PRACE NAUKOWE POLITECHNIKI WROCŁAWSKIEJ

Scientific Papers of the Institute of Technical Cybernetics
No. 80 of the Technical University of Wrocław No. 80

Monographs No. 15 1989

Jan MAGOTT

Petri nets in computer systems performance evaluation

Prace Naukowe Instytutu Cybernetyki Technicznej
Politechniki Wrocławskiej 80

Seria :
Monografie 15

Jan Magott

Sieci Petriego w ocenie wydajności systemów komputerowych

Wydawnictwo Politechniki Wrocławskiej · Wrocław 1989

Redaktor naukowy
Jacek JARNICKI

Recenzenci
Jacek BŁAŻEWICZ
Tadeusz SAWIK

Opracowanie redakcyjne
Alina KACZAK

Korekta
Barbara WACHOWSKA

WYDAWNICTWO POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

ISSN 0324-9786

Nakład 200 + 60 egz. Ark. wyd. 9,75. Ark. druk. 8 1/8. Papier offset. kl. III, 70 g, B1.
Oddano do druku w styczniu 1989 r. Druk ukończono w styczniu 1989 r.
Zakład Graficzny Politechniki Wrocławskiej. Zam. 3/89. Cena zł 250,–

Prace Naukowe Instytutu Cybernetyki Technicznej
Nr 80 Politechniki Wrocławskiej Nr 80
Monografie Nr 15 1989

Czasowa sieć Petriego,
złożoność obliczeniowa,
proces sekwencyjny,
wzajemne wykluczanie,
komunikujące się procesy sekwencyjne,
komunikacja poprzez bufory

Jan MAGOTT*

SIECI PETRIEGO W OCENIE WYDAJNOŚCI SYSTEMÓW KOMPUTEROWYCH

Praca poświęcona jest ocenie wydajności systemów komputerowych z zastosowaniem czasowych sieci Petriego. W pracy rozwinięto podstawy deterministycznych sieci Petriego (czas palenia przejścia zadany jest liczbą rzeczywistą, natomiast wybór przejść do palenia musi spełniać ograniczenia zadane wektorem wzajemnych częstości palenia przejść) do oceny wydajności systemów komputerowych. Ponadto przeanalizowano wydajność takich zasadniczych mechanizmów współpracy współbieżnie przebiegających procesów sekwencyjnych, jak wzajemne wykluczanie, komunikacja poprzez synchronizację nadawcy i odbiorcy, komunikacja przez bufory. Szacowania wydajności dokonano na podstawie sieci Petriego (ze szczególnym uwzględnieniem sieci deterministycznych). Ze względu na strukturalną i funkcjonalną złożoność systemów komputerowych, podstawowym zagadnieniem w dziedzinie problemów oceny wydajności jest złożoność obliczeniowa tych problemów. Dla systemów procesów sekwencyjnych z wymienionymi mechanizmami współpracy określono własności, które są krytyczne dla złożoności obliczeniowej.

SPIS CZĘŚCIEJ UŻYWANYCH OZNACZEŃ
(LIST OF SYMBOLS)

X^{**} - liczba mnoga znaczenia przypisanego skrótowi X (plural for meaning of symbol X)

$|Y|$ - liczba elementów skończonego zbioru Y (number of elements of finite set Y)

SP - sieć Petriego (Petri net)

ZSP - znakowana sieć Petriego (marked Petri net)

---

* Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

DSP - deterministyczna sieć Petriego (deterministic Petri net)
DPCW - decyzyjny problem czasu wykonania (recognition execution time problem)
OPCW - optymalizacyjny problem czasu wykonania (optimization execution time problem)
$DPCC_1$($OPCC_1$) - decyzyjny (optymalizacyjny) problem czasu cyklu pierwszego rodzaju (first recognition (optimization) cycle time problem)
$DPCC_2$($OPCC_2$) - decyzyjny (optymalizacyjny) problem czasu cyklu drugiego rodzaju (second recognition (optimization) cycle time problem)
DPCC(OPCC) - decyzyjny (optymalizacyjny) problem czasu cyklu drugiego rodzaju (second recognition (optimization) cycle time problem)
PCC - problem czasu cyklu drugiego rodzaju (second cycle time problem)
USSP - uogólniona stochastyczna sieć Petriego (generalized stochastic Petri net)
WŁM - włożony łańcuch Markowa (embedded Markov chain)
ZWŁM - zredukowany włożony łańcuch Markowa (reduced embedded Markov chain)
CSP - komunikujące się procesy sekwencyjne (communicating sequential processes)
CCS - rachunek systemów komunikujących się (calculus of communicating systems)
SPKB - system procesów sekwencyjnych komunikujących się przez bufory (system of sequential processes communicating by buffers)

WSTĘP

0.1. Współbieżność w systemach komputerowych

W systemach komputerowych występuje współbieżnie wiele zjawisk. Współbieżnie działają elementy logiczne w danym urządzeniu systemu. Już nawet w pierwszych systemach komputerowych wprowadzanie danych dokonywane było poprzez współbieżną pracę czytnika, procesora (przez którego akumulator przesyłano dane) oraz pamięci operacyjnej.

Wprowadzenie współbieżności procesów przebiegających w systemach komputerowych jest metodą zwiększenia wydajności tych systemów. W miarę rozwoju systemów komputerowych rośnie stopień współbieżności procesów. Na początkowym etapie rozwoju wzrost wydajności osiągnięto przez współbieżność procesu obliczeniowego i procesów wejścia/wyjścia. Dalszą fazą rozwoju było wprowadzenie wieloprogramowania, a w końcu wieloprzetwarzania. W zakresie realizacji idei wieloprzetwarzania wprowadzono komputery równoległe o wielu procesorach oraz sieci komputerowe.

Przebiegające współbieżnie procesy współzawodniczą w dostępie do wspólnych zasobów oraz komunikują się między sobą.

Wiele zasobów, o które współzawodniczą procesy, w danej chwili może być użytkowanych przez co najwyżej jeden proces. Dlatego często dostęp do dzielonych zasobów jest realizowany w trybie wzajemnego wykluczania.

Komunikacja między procesami w środowisku zwartym (wspólna pamięć operacyjna) jest dokonywana za pomocą zmiennych dzielonych, natomiast w środowisku rozproszonym (procesory posiadają pamięci lokalne, np. sieci komputerowe, sieci mikroprocesorowe) - poprzez wymianę komunikatów.

W komunikacji za pomocą zmiennych dzielonych, często stosuje się wzajemne wykluczanie w dostępie do tych zmiennych.

Wymiana komunikatów jest realizowana przez wysyłanie i przyjmowanie komunikatów, stanowiących ciąg wartości wyrażeń, a nie przez zapis i odczyt zmiennych globalnych we wspólnej pamięci.

Między nadawcą i odbiorcą może istnieć lub nie istnieć mechanizm buforowania.

W razie braku mechanizmu buforowania wymiana komunikatów wymaga synchronizacji nadawcy i odbiorcy. Jest to najczęściej stosowana metoda - zastosowana m.in. w paradygmatycznych językach programowania współbieżnego: CSP (komunikujące się procesy sekwencyjne [49], CCS (rachunek systemów komunikujących się) [87]. Znaczenie tych paradygmatycznych języków wynika z następujących faktów. Język CSP zastosowano w projektowaniu języka Ada, który w programowaniu współbieżnym może odegrać taką rolę, jak język Pascal w programowaniu sekwencyjnym. Ponadto na języku CSP oparty jest język Occam dla Transputerów (systemów mikroprocesorowych, z których można budować komputery wieloprocesorowe, np. systoliczne). Na podstawie języka CCS sformułowano LOTOS - język formalnych specyfikacji usług i protokołów w sieciach komputerowych.

Asynchroniczna wymiana komunikatów wymaga nieograniczonego bufora, ponieważ nadawca może dowolnie wyprzedzić odbiorcę.

Między tymi dwiema skrajnościami jest buforowana wymiana komunikatów - z zastosowaniem bufora o ograniczonej pojemności, co jest stosowane np. w języku programowania współbieżnego CHILL oraz w większości protokołów sieci komputerowych.

Z danych rozważań wynika, że zasadniczymi mechanizmami stosowanymi w zagwarantowaniu współpracy procesów są:

- wzajemne wykluczanie,
- wymiana komunikatów poprzez synchronizację nadawcy i odbiorcy,
- wymiana komunikatów przez bufory.

Ocena wydajności systemów komputerowych o współbieżnie przebiegających procesach sekwencyjnych wymaga szacowania nie tylko charakterystyk czasowych poszczególnych procesów, ale również wpływu mechanizmów współpracy procesów. Zagadnienie szacowania charakterystyk czasowych procesów sekwencyjnych [69], [118] jest znacznie lepiej zbadane i dużo prostsze niż analogiczne zagadnienie dla systemów współbieżnie przebiegających procesów sekwencyjnych. W drugim przypadku trzeba uwzględnić wpływ mechanizmów współpracy procesów na wydajność systemu komputerowego.

Pojęcie procesu obejmuje nie tylko ciąg akcji wynikających z wykonania procedury, ale również ciąg akcji w sprzęcie systemu komputerowego. Podejmowane są próby korzystania z dorobku teorii programowania w syntezie układów bardzo dużej skali integracji [42], [86]. Prawdopodobnie z czasem coraz więcej algorytmów będzie odwzorowywanych w tego rodzaju układach. Kompilatory krzemowe są istotnym elementem tego kierunku rozwoju.

0.2. Znaczenie sieci Petriego w ocenie wydajności systemów komputerowych

Sieci Petriego umożliwiają reprezentację współbieżnych zjawisk występujących w systemach komputerowych w sposób najbardziej naturalny spośród istniejących formalizmów opisu systemów. Sieci te są stosowane w specyfikacji [83], [124], w projektowaniu [7], [22], [64], [97], [121], weryfikacji [11], [15], [59], [61], [91], w ocenie wydajności i niezawodności [5], [125] systemów komputerowych.

W celu badania wydajności systemów komputerowych, sieci Petriego są wzbogacane o czynnik czasu. Czasowe sieci Petriego są stosowane do oceny wydajności: architektury komputerów jednoprocesorowych [127], magistralowej architektury wieloprocesorowej [3], [4], architektury komputerów potokowych (pipeline) [45], [46], [50], [102], komputerów sterowanych przepływem danych [58], [114], komputerów równoległych opartych na Transputerach [2], programów sekwencyjnych [69], współbieżnych systemów czasu rzeczywistego [117], oprogramowania z sekcjami krytycznymi [12], [74], oprogramowania z komunikacją między procesami [2], [77], protokołów komunikacyjnych w sieciach komputerowych [89], [103], rozproszonych baz danych [96], [108].

Czasowe sieci Petriego są stosowane również w niezawodności systemów komputerowych [5], [125].

Sieci Petriego z czynnikiem czasu są stosowane w analitycznych i symulacyjnych metodach oceny wydajności systemów komputerowych. W pracy badamy tylko metody analityczne.

Obecnie przedstawimy miejsce modeli opartych na sieciach Petriego w klasie modeli analitycznych stosowanych do oceny wydajności systemów komputerowych.

Modele analityczne na potrzeby oceny wydajności systemów komputerowych można podzielić na:

- deterministyczne,
- stochastyczne,
- mieszane.

Wśród modeli deterministycznych można wymienić: modele szeregowania deterministycznego [17], [18], [26], [27], [32], [126], deterministyczne sieci Petriego [73], [102], [113].

W deterministycznych sieciach Petriego czas palenia przejścia jest zadany liczbą rzeczywistą, natomiast wybór przejść do palenia musi spełniać ograniczenia zadane wektorem wzajemnych częstości palenia przejść.

W grupie modeli stochastycznych wyróżnia się modele oparte na łańcuchach i procesach Markowa [118], modele kolejkowe [6], [13], [27], [47], [105], modele szeregowania stochastycznego [32], stochastyczne sieci Petriego [4], [90].

Stochastyczne sieci Petriego charakteryzują się czasem palenia przejść zadanym zmienną losową oraz losowym wyborem przejść do palenia.

Do klasy modeli mieszanych należą mieszane sieci Petriego.

W mieszanych sieciach Petriego czas palenia przejścia jest zadany liczbą rzeczywistą, natomiast wybór przejść do palenia jest opisany rozkładem prawdopodobieństwa.

Stosowalność poszczególnych modeli jest wyznaczona ich możliwościami opisowymi (mocą ekspresji) oraz złożonością obliczeniową. Model powinien umożliwiać wyrażanie możliwie wielu cech systemu komputerowego. Z drugiej strony, ze względu na złożoność systemów komputerowych model powinien być dekomponowalny, w celu zmniejszenia złożoności obliczeniowej jego rozwiązania.

Sieci Petriego, dzięki ich mocy ekspresji, sięgającej mocy ekspresji maszyn Turinga, dopuszczają wyrażanie praktycznie wszystkich cech systemów komputerowych.

Wśród modeli deterministycznych teoria szeregowania deterministycznego jest znacznie bardziej rozwinięta niż tematyka deterministycznych sieci Petriego. Jednakże problemy szeregowania deterministycznego mogą być wyrażane za pomocą sieci Petriego, o czym świadczą wyniki prac [23], [76]. Ponadto problematyka sieci Petriego sugeruje stawianie nowych problemów szeregowania deterministycznego. Przykładem może być problem minimalnego czasu cyklu dla sieci Petriego [73], [101], [102], którego odpowiednik nie jest zawarty w tematyce cykliczności teorii

szeregowania. Z drugiej strony, tam gdzie jest to możliwe, należy korzystać z wyników teorii szeregowania w zakresie sieci Petriego.

Dokonajmy dalszego porównania modeli szeregowania deterministycznego z deterministycznymi sieciami Petriego.

W deterministycznej sieci Petriego przejście może być w danej chwili palone wielokrotnie, tzn. przejście może być zapalone po raz kolejny nawet, jeśli poprzedni proces jego palenia nie został zakończony. Możliwość taka jest uzasadniona własnościami komputerów. Na przykład w razie komputerów potokowych, jednostka funkcjonalna, wykonująca operacje dodawania dwu liczb zmiennoprzecinkowych, może rozpocząć dodawanie kolejnych dwu liczb nawet, jeśli poprzednie dodawanie nie zostało ukończone. Istnieją procedury, tzw. procedury czyste, których inicjowanie po raz kolejny może wystąpić przed zakończeniem ich poprzedniego wykonania. W języku teorii szeregowania powyższe przypadki odpowiadają jednoczesnemu użytkowaniu zasobu przez wiele zadań. Klasyczne modele teorii szeregowania takiej możliwości nie zakładają.

Teoria szeregowania dla kryteriów takich, jak: czas wykonania zbioru zadań, średni czas przepływu, maksymalne opóźnienie, zakłada skończony zbiór zadań, których kolejność wykonywania jest zadana relacją częściowego porządku. W konstrukcji rozwiązania problemu szeregowania dla wymienionych kryteriów, za podstawę przyjmuje się jednokrotne wystąpienie zbioru zadań. Podejście oparte na sieciach Petriego dopuszcza cykliczność procesu palenia przejść. Podejście to, w odniesieniu do zbioru zadań, pozwala przed ukończeniem danego zadania w i-tej realizacji zbioru zadań - zainicjować jego wykonanie w (i+1)-tej i kolejnych realizacjach tego zbioru. Uwzględnianie tej ewentualności można uzasadnić poprzez odwołanie się do przykładów, takich jak zadanie wykonywane przez jednostkę funkcjonalną komputera potokowego czy zadanie realizacji procedury czystej.

Poprzez rozwiązanie problemu minimalnego czasu cyklu dla sieci Petriego możemy wyznaczyć przepustowość systemu komputerowego. Klasyczne modele teorii szeregowania nie są zorientowane na badanie przepustowości systemu. Ponadto rozwiązanie problemu minimalnego czasu cyklu może służyć do określenia maksymalnych wartości współczynników wykorzystania elementów systemu (w warunkach obciążenia systemu nie mniejszego niż jego przepustowość).

W klasie modeli stochastycznych dominującymi są sieci kolejkowe. Ranga modeli kolejkowych wynika ze stosunkowo niedużej złożoności obliczeniowej metod korzystających z tych modeli.

Modele kolejkowe, mimo wielu rozszerzeń wprowadzanych do nich, nie umożliwiają wyrażania pewnych elementów współbieżności systemów

komputerowych [120]. Rozszerzenia powodują często zwiększenie złożoności obliczeniowej metod rozwiązywania sieci kolejkowych.

Podejmowane są próby łączenia modeli kolejkowych z sieciami Petriego [12],[120], w celu wykorzystania mniejszej złożoności obliczeniowej modeli kolejkowych i większych możliwości opisowych sieci Petriego.

Ze względu na zaawansowanie teorii łańcuchów Markowa [118], teorii kolejek [47], teorii szeregowania stochastycznego [32] jest uzasadnione stosowanie wyników tych teorii do stochastycznych sieci Petriego.

Zaletą sieci Petriego, w stosunku do innych wymienionych modeli do oceny wydajności, jest możliwość stosowania sieci w specyfikacji, weryfikacji i projektowaniu systemów komputerowych w warstwie syntaktycznej i semantycznej [59],[61],[64],[82],[94],[95],[121],[122]. Względem pozostałych modeli, z wyłączeniem łańcuchów i procesów Markowa, sieci Petriego mają przewagę wynikającą z możliwości ich zastosowania w ocenie niezawodności.

0.3. Cel pracy

Celem pracy jest:

1. Rozwój podstaw deterministycznych sieci Petriego do oceny wydajności systemów komputerowych,

2. Rozwój opartych na sieciach Petriego metod oceny wydajności zasadniczych mechanizmów współpracy współbieżnie przebiegających procesów sekwencyjnych (ze szczególnym uwzględnieniem sieci deterministycznych).

Obecnie przedstawimy uzasadnienie adekwatności deterministycznych sieci Petriego jako modeli systemów komputerowych do oceny ich wydajności.

Architektura wielu komputerów równoległych, np. komputerów potokowych składa się ze specjalizowanych jednostek funkcjonalnych. Instrukcje języka wewnętrznego są wykonywane poprzez realizację mikroinstrukcji w tych jednostkach. Czasy wykonywania mikroinstrukcji są zwykle znane. Dla danego typu instrukcji języka wewnętrznego są znane zwykle liczby wykonań poszczególnych mikroinstrukcji.

Systemy komputerowe sterujące systemami produkcyjnymi nadzorują operacje obróbki detali, operacje montażu itd. Czasy wykonywania tych operacji są w wielu przypadkach znane jako czasy cyklicznie przebiegających procesów technologicznych. Zwykle znane są również liczby wykonań poszczególnych operacji w cyklu technologicznym.

Dla wielu rozwiązań architektury komputerów i systemów komputerowych sterujących produkcją, deterministyczne sieci Petriego są adekwatnym modelem do oceny wydajności. Czasy wykonywania mikroinstrukcji czy operacji technologicznych mogą być odwzorowywane za pomocą czasów palenia przejść. Z kolei liczby wykonań mikroinstrukcji w procesie realizacji instrukcji języka wewnętrznego, liczby wykonań operacji w cyklu technologicznym mogą być przetransformowane w wektor wzajemnych częstości palenia przejść.

Nawet jeśli czasy wykonywania mikroinstrukcji, operacji czy liczby ich wykonań nie są jednoznacznie określone, to można zastosować model deterministyczny do szacowania charakterystyk wydajnościowych.

Jeśli dla każdego z czasów wykonywania jest znane jego górne oszacowanie oraz są znane górne oszacowania krotności wykonań mikroinstrukcji, operacji, to korzystając z deterministycznej sieci Petriego, można otrzymać pesymistyczne oszacowanie wydajności.

Z drugiej strony można otrzymać optymistyczne oszacowanie wydajności, jeśli jako czasy palenia przejść deterministycznej sieci Petriego przyjmie się wartości średnie zmiennych losowych czasów wykonywania oraz wzajemne częstości palenia przejść wyznaczy się na podstawie wartości średnich liczb wykonań.

Rozważmy system współbieżnie przebiegających procesów sekwencyjnych. Niech czas wykonania każdej instrukcji będzie znany. Dla instrukcji decyzyjnych niech będą znane prawdopodobieństwa wyboru poszczególnych kierunków przepływu sterowania. Przetransformujmy sieć działań systemu procesów w deterministyczną sieć Petriego: niech czasy palenia przejść będą równe czasom wykonywania instrukcji, natomiast prawdopodobieństwa wyboru poszczególnych kierunków przepływu sterowania odwzorujmy w wektor wzajemnych częstości palenia przejść. Na podstawie uzyskanej sieci można otrzymać optymistyczne oszacowanie charakterystyk wydajnościowych wyjściowego systemu procesów.

Jeśli dla systemu współbieżnych procesów instrukcje są instrukcjami złożonymi lub modułami, to czasy ich wykonywania można opisać zmiennymi losowymi. W celu odwzorowania sieci działań w deterministyczną sieć Petriego, przyjmiemy czasy palenia przejść jako równe wartościom średnim zmiennych losowych czasów wykonywania instrukcji (modułów). W tym przypadku również można otrzymać optymistyczne oszacowanie wydajności systemu procesów.

W zakresie oceny wydajności mechanizmów współpracy współbieżnie przebiegających procesów sekwencyjnych badamy nie tylko systemy procesów opisywalne deterministycznymi sieciami Petriego, ale również systemy opisane sieciami stochastycznymi. Szacowanie wydajności systemów procesów o stochastycznym charakterze za pomocą deterministycznych sie-

ci Petriego może być przyczyną znacznych błędów. Dlatego do oceny wydajności tego rodzaju systemów często niezbędne jest skorzystanie ze stochastycznych sieci Petriego.

Zakres stosowalności modeli deterministycznych i stochastycznych zależy nie tylko od stopnia adekwatności modelu do opisywanego systemu rzeczywistego. Ważna jest również możliwość znalezienia rozwiązania zagadnienia oceny wydajności systemu w zależności od wybranego modelu. Jeśli nie potrafimy uzyskać rozwiązania dokładnego o satysfakcjonującej złożoności obliczeniowej, to próbujemy konstruować metody przybliżone. Nierzadko w celu znalezienia rozwiązań przybliżonych zagadnień oceny wydajności systemów o stochastycznej naturze, korzystamy z modeli deterministycznych.

Czynnikiem wpływającym na wybór modelu jest również zasób informacji o zjawiskach losowych systemu rzeczywistego. Znaczenie tego czynnika zilustrujemy przykładem.

W systemach o współbieżnie przebiegających procesach, często jest wymagana synchronizacja procesów. Niech w celu osiągnięcia punktu synchronizacji, każdy z synchronizowanych procesów musi osiągnąć wymagany stan. Dlatego realizacja zmiennej losowej czasu zsynchronizowania procesów równa jest maksymalnej spośród realizacji zmiennych losowych czasów osiągania punktów synchronizacji przez procesy. Załóżmy, że dysponujemy tylko wartościami średnimi zmiennych losowych czasów osiągania punktu synchronizacji przez procesy. Wtedy najlepszym dolnym oszacowaniem wartości średniej zmiennej losowej czasu zsynchronizowania procesów jest maksymalna spośród wartości średnich zmiennych losowych czasów osiągania punktu synchronizacji. Zatem w tym przypadku, w celu uzyskania najlepszego dolnego oszacowania możemy posłużyć się modelem deterministycznym, przyjmując zamiast zmiennych losowych ich wartości średnie.

0.4. Zakres pracy

Systemy komputerowe charakteryzują się ogromną złożonością strukturalną i funkcjonalną. Stąd wyznaczanie ich charakterystyk wydajnościowych jest złożone obliczeniowo. Zatem dany problem oceny wydajności należy przebadać pod kątem złożoności obliczeniowej. Zależy nam na znalezieniu jak najefektywniejszego obliczeniowo algorytmu, aby rozwiązać problem oceny wydajności nawet dla znacznych jego rozmiarów. Przykładem rozmiaru problemu oceny wydajności może być liczba instrukcji, liczba procesów, liczba procesorów itd.

W pierwszej kolejności omówimy zagadnienie złożoności obliczeniowej problemów oceny wydajności rozwiązywanych na podstawie deterministycznych sieci Petriego. Ponieważ problemy te są problemami kombinatorycznymi, a więc analizując je korzystamy z teorii złożoności obliczeniowej problemów kombinatorycznych. Zgodnie z tą teorią; jeśli dla danego problemu nie potrafimy znaleźć algorytmu efektywnego (o wielomianowej złożoności czasowej), to pragniemy dowieść, że dla tego problemu algorytm taki najprawdopodobniej nie istnieje (wykazując, że dany problem jest NP-trudny).

Zwykle dla danej klasy systemów, np. systemów procesów z określonym mechanizmem współpracy i zadanej funkcji kryterialnej, można wyodrębnić wiele problemów oceny wydajności. Różne problemy wyznacza się poprzez specyficzne dla nich cechy. Na przykład dla systemu procesów z mechanizmem wzajemnego wykluczania, różne problemy otrzymujemy dla różnych liczb procesów, rozmaitych postaci procesów, zróżnicowanych liczb dostępów do zasobów itd. W przypadku komunikacji poprzez bufory, rozmaite postaci komunikacji determinują różne problemy oceny wydajności.

Rozwiązanie zagadnienia złożoności obliczeniowej problemów oceny wydajności dla danej klasy systemów i zadanej funkcji kryterialnej, przez przebadanie każdego z problemów z osobna, jest zbyt skomplikowane. Inną drogą jest szukanie granicy między problemami "łatwymi" (z algorytmem o wielomianowej złożoności czasowej) a problemami "trudnymi" (o których udowodniono, że są NP-trudne). Jeśli granica ta jest dobrze zbadana, to oszacowania złożoności obliczeniowej problemu oceny wydajności można dokonać poprzez wyznaczenie położenia tego problemu względem granicy.

Zaklasyfikowanie problemu oceny wydajności do jednej z dwu klas problemów (problemów "łatwych" lub "trudnych") ukierunkowuje dalsze jego badanie. Udowodnienie "łatwości" problemu jest dokonywane poprzez znalezienie efektywnego algorytmu. Dalsze badania mogą być próbami znalezienia algorytmu jeszcze efektywniejszego (o mniejszej czasowej złożoności obliczeniowej). Wykazanie "trudności" problemu orientuje dalszą jego analizę na badanie efektywności metod nieefektywnych lub przybliżonych.

W przypadku zagadnienia złożoności obliczeniowej problemów oceny wydajności z zastosowaniem stochastycznych sieci Petriego, nie można skorzystać z odpowiednika teorii złożoności obliczeniowej problemów kombinatorycznych. Jednak można wyznaczać rząd złożoności obliczeniowej algorytmu lub jego części.

W zakresie realizacji punktu 1 celu pracy badamy deterministyczne sieci Petriego spełniające ograniczenia charakterystyczne dla rzeczywistych systemów komputerowych. Ze względów praktycznych szczególnie

ważne są systemy komputerowe, w których nie istnieje możliwość wystąpienia blokady. Rzeczywiste systemy komputerowe charakteryzuje skończona moc zbioru zasobów, zbioru procesów. Systemy komputerowe o powyższych własnościach można opisać za pomocą żywych i ograniczonych sieci Petriego. Sieci takie charakteryzują się cyklicznością działania.

W przypadku systemów komputerowych wyrażalnych żywymi i ograniczonymi deterministycznymi sieciami Petriego, maksymalną wydajność (rozumianą jako największa szybkość przebiegu systemu instrukcji, operacji, procesów) uzyskuje się dla minimalnego czasu cyklu.

W rozdziale drugim, poświęconym problemowi czasu cyklu dla wybranych klas deterministycznych sieci Petriego, udowodniliśmy twierdzenia dotyczące relacji między klasami sieci i twierdzenie o rozstrzygalności problemu czasu cyklu. Ponadto uzyskaliśmy wyniki z zakresu złożoności obliczeniowej problemu czasu cyklu dla wybranych klas sieci, m.in. rezultaty dotyczące granicy między klasami sieci, dla których problem czasu cyklu jest "łatwy" a klasami, dla których problem ten jest "trudny". Przedstawiliśmy wielomianowe oszacowania czasu cyklu.

Rozdział drugi jest syntezą prac autora [73], [75], [78], [79] na temat problemu czasu cyklu dla deterministycznych sieci Petriego, a zawarte w nim rezultaty stanowią znaczne rozszerzenie prac innych autorów [24], [25], [101], [102], [111].

W ramach realizacji punktu 2 celu pracy badamy wydajność wszystkich zasadniczych mechanizmów współpracy procesów sekwencyjnych, a mianowicie:

- wzajemnego wykluczania,
- komunikacji poprzez synchronizację nadawcy i odbiorcy (na przykładzie komunikujących się procesów sekwencyjnych - CSP),
- komunikacji poprzez bufory.

W systemach komputerowych (np. w ich systemach operacyjnych) szczególnie duże znaczenie mają cykliczne procesy sekwencyjne. Dlatego zasadnicza część uzyskanych wyników dotyczy systemów cyklicznych procesów sekwencyjnych. Badamy również systemy procesów niecyklicznych.

Miarą wydajności systemów procesów sekwencyjnych jest szybkość przebiegu systemu procesów.

Systemy procesów opisywanych deterministycznymi sieciami Petriego. Największą szybkość przebiegu systemu procesów uzyskujemy przy minimalnym czasie cyklu systemu - dla procesów cyklicznych oraz przy minimalnym czasie wykonania - dla procesów niecyklicznych.

Dla systemów procesów wyrażanych deterministycznymi sieciami Petriego, szukamy takich własności tych systemów, które mają decydujący wpływ na złożoność obliczeniową problemów oceny wydajności.

W obszarze złożoności obliczeniowej problemów oceny wydajności systemów procesów sekwencyjnych z wzajemnym wykluczaniem - w przypadku procesów bez cykli można skorzystać z wyników teorii szeregowania deterministycznego [17], [26]. Natomiast dla procesów cyklicznych - rezultaty uzyskane przez autora w pracach [74], [77] są pierwszymi w literaturze. W rozdziale trzecim pracy zawarte są rezultaty mocniejsze niż wyniki dwu powyższych prac autora. W rozdziale tym są badane procesy sekwencyjne o dwu rodzajach struktur: procesy opisane obwodami oraz automatami. Procesy wyrażone obwodami stanowią podklasę procesów definiowanych automatami. Udowodnione w pracy twierdzenia wskazują, że własnością krytyczną dla zdolności obliczeniowej problemu czasu cyklu dla systemów procesów sekwencyjnych z wzajemnym wykluczaniem jest liczba dostępów do zasobów w czasie cyklu systemu.

W zakresie złożoności obliczeniowej problemów oceny wydajności komunikujących się procesów sekwencyjnych (CSP) wyniki rozdziału czwartego są pierwszymi w literaturze. W rozdziale tym, oprócz procesów cyklicznych, badamy również procesy niecykliczne.

Dla systemów procesów niecyklicznych (z instrukcjami stopu) rozważamy procesy reprezentowane drogami prostymi oraz automatami. W przypadku procesów cyklicznych analizujemy procesy **ilustrowane** obwodami oraz procesy wyrażone automatami.

Na podstawie sformułowanych twierdzeń wnioskujemy, że zarówno w przypadku procesów niecyklicznych, jak również cyklicznych decydujący wpływ na złożoność obliczeniową problemów oceny wydajności ma struktura procesu.

W pracy [104] analizowana jest złożoność obliczeniowa problemów oceny wydajności systemów procesów komunikujących się poprzez bufory dla szczególnego przypadku procesów, a mianowicie procesów opisanych obwodami. Dla procesów wyrażonych automatami, pierwsze wyniki zawiera praca autora [77]. Rezultaty mocniejsze od przedstawionych w tej pracy uzyskano w rozdziale piątym. W rozdziale tym badana jest złożoność obliczeniowa problemu czasu cyklu dla komunikacji między procesami opisanymi automatami, poprzez bufory bez ograniczeń na ich pojemność i komunikacji poprzez bufory o ograniczonej pojemności. W przypadku buforów o nieograniczonej pojemności, własnością decydującą dla złożoności obliczeniowej jest istnienie obwodów w tzw. strukturze zgrubnej. Struktura ta służy ilustracji kierunków komunikacji między procesami. Problem czasu cyklu dla komunikacji poprzez bufory o ograniczonej pojemności jest nie mniej złożony niż w przypadku komunikacji poprzez bufory bez ograniczeń na ich pojemność.

Systemy procesów opisywanych stochastycznymi sieciami Petriego. Miarą wydajności systemu jest średni czas cyklu systemu - dla procesów

cyklicznych oraz średni czas wykonania systemu procesów - dla procesów niecyklicznych.

W celu uzyskania rozwiązań analitycznych problemów oceny wydajności systemu procesów wyrażonego stochastyczną siecią Petriego, przyjmujemy takie rozkłady czasów palenia przejść (modelujących czasy wykonywania instrukcji, operacji), które umożliwiają zastosowanie wyników teorii łańcuchów Markowa [118].

Dla systemów procesów sekwencyjnych z wzajemnym wykluczaniem, komunikujących się procesów sekwencyjnych (CSP) opisywalnych stochastycznymi sieciami Petriego, sformułowaliśmy w pracy specyficzne dla tych systemów algorytmy do oceny wydajności. Algorytmy te są efektywniejsze obliczeniowo niż ogólny algorytm macierzowy dla sieci stochastycznych zawarty w pracy [4]. Porównanie sformułowanych w pracy algorytmów z macierzowym obrazuje przyczyny, z których powodu algorytmy te są efektywniejsze od metody macierzowej. Systemy procesów z komunikacją poprzez bufory wyrażone sieciami stochastycznymi nie są w pracy badane ze względu na podobieństwo do CSP.

W przypadku mieszanych sieci Petriego również można skorzystać z teorii łańcuchów Markowa. Przypadek ten ze względu na podobieństwo metod jego analizy do metod badania sieci stochastycznych nie jest w pracy rozważany.

Omówienie rezultatów poszczególnych rozdziałów zawarte jest w podsumowaniach rozdziałów.

Kierunki dalszych badań przedstawione są w zakończeniu pracy.

1. DEFINICJE

1.1. Definicje podstawowe

$N = \{0,1,\ldots\}$ jest zbiorem liczb naturalnych. Sieć Petriego (SP) jest czwórką uporządkowaną $\mathcal{N} = <P,T,F,W>$, gdzie $P = \{p_1,\ldots,p_m\}$ - zbiór miejsc, $T = \{t_1,\ldots,t_n\}$ - zbiór przejść, $F \subset (P \times T) \cup (T \times P)$ - zbiór łuków, $W : F \longrightarrow N \setminus \{0\}$ - funkcja krotności łuku.

W reprezentacji graficznej, miejsca są przedstawiane w postaci kółek, natomiast przejścia - w postaci kresek. Jeśli krotność łuku $f \in F$ spełnia zależność $W(f) > 1$, to łuk obciążony jest liczbą $W(f)$. Jeśli $W(f) = 1$, to symbol 1 jest pomijany. Skróty opatrzone symbolem $*$ u góry wyrażać będą liczbę mnogą (np. symbol SP^* oznacza sieci Petriego). Znakowanie SP jest funkcją $M : P \longrightarrow N$. Znakowanie alternatywnie jest przedstawiane jako wektor kolumnowy M o liczbie składowych równej liczbie miejsc SP. Wartość $M(p_i)$ wyraża liczbę znaczników w miejscu p_i. Liczba

znaczników jest oznaczana na rysunkach przez liczbę kropek. Znakowana sieć Petriego (ZSP) jest parą $\mathcal{M} = < \mathcal{N}, M_0 >$, gdzie M_0 jest znakowaniem SP $\mathcal{N}$ zwanym znakowaniem początkowym. Zbiór ${}^\bullet t = \{ p \in P : < p,t > \in F \}$ ($t^\bullet = \{ p \in P : < t,p > \in F \}$) jest zbiorem miejsc wejściowych (wyjściowych) przejścia t. W podobny sposób są zdefiniowane zbiory ${}^\bullet p$ ($p^\bullet$) przejść wejściowych (wyjściowych) miejsca p. Przejście t_j jest przygotowane do palenia, jeśli w każdym miejscu $p_i \in {}^\bullet t_j$ znajduje się co najmniej $W(< p_i, t_j >)$ znaczników. Jeśli przejście jest przygotowane do palenia, to może nastąpić jego palenie. Gdy przejście t_j pali się, wówczas $W(< p_i, t_j >)$ znaczników jest usuwanych z każdego miejsca $p_i \in {}^\bullet t_j$ oraz $W(< t_j, p_k >)$ znaczników jest dodawanych do każdego miejsca $p_k \in t_j^\bullet$.

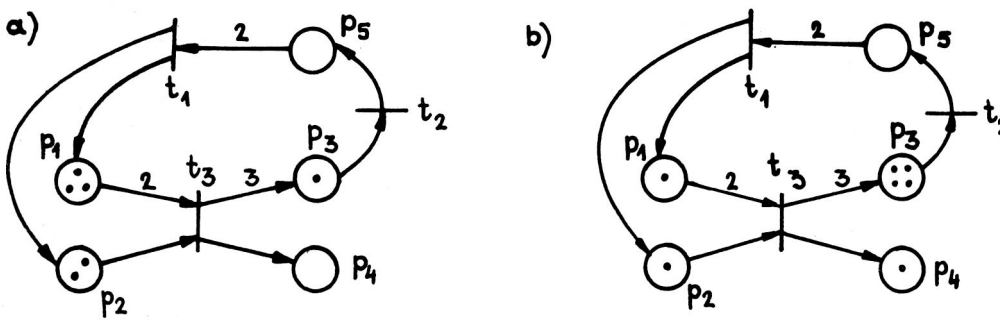
Rozważmy ZSP z rys. 1.1a.

Przejście t_3 jest przygotowane do palenia. Po paleniu tego przejścia otrzymujemy ZSP z nowym znakowaniem ilustrowaną rys. 1b.

SP $\mathcal{N} = <P,T,F,W>$ jest czystą SP, jeśli

${}^\bullet t \cap t^\bullet = \emptyset$ dla każdego $t \in T$.

Dla czystej SP z $|P| = m$, $|T| = n$ definiujemy następujące macierze:



Rys. 1.1

$$C = [c_{ij}]_{mxn},$$

$$c_{ij} = \begin{cases} -W(< p_i, t_j >) & \text{jeśli } < p_i, t_j > \in F, \\ W(< t_j, p_i >) & \text{jeśli } < t_j, p_i > \in F, \\ 0 & \text{w przeciwnym przypadku,} \end{cases}$$

$$C^- = [c_{ij}^-]_{mxn},$$

$$c_{ij}^- = \begin{cases} W(< p_i, t_j >) & \text{jeśli } < p_i, t_j > \in F, \\ 0 & \text{w przeciwnym przypadku,} \end{cases}$$

$$C^+ = [c^+_{ij}]_{mxn},$$

$$c^+_{ij} = \begin{cases} W(<t_j,p_i>) & \text{jeśli } <t_j,p_i> \in F, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Macierz C nazywamy macierzą incydencji SP. Powyższe macierze spełniają zależność

$$C = C^+ - C^-.$$

Jeśli w wyniku palenia przejścia t_i następuje zmiana znakowania z M_i na M_{i+1}, to stosujemy oznaczenie $M_i \xrightarrow{t_i} M_{i+1}$. Sekwencją palenia nazywamy taką sekwencję przejść $\sigma = t_{i_1},\ldots,t_{i_s}$, dla której istnieją znakowania $M_p, M_{p+1},\ldots,M_{p+s}$ spełniające warunek

$$M_p \xrightarrow{t_{i_1}} M_{p+1},\ldots,M_{p+s-1} \xrightarrow{t_{i_s}} M_{p+s}.$$

W danym przypadku stosujemy oznaczenie $M_p \xrightarrow{\sigma} M_{p+s}$. Znakowanie M_{p+s} nazywamy znakowaniem osiągalnym ze znakowania M_p.

ZSP jest żywą, jeśli dla każdego znakowania M osiągalnego ze znakowania początkowego M_0 i dla każdego przejścia t istnieje sekwencja palenia umożliwiająca palenie przejścia t. ZSP jest ograniczona, jeśli istnieje taka naturalna liczba k, że dla każdego miejsca p i dla każdego znakowania M osiągalnego z M_0 prawdziwa jest zależność $M(p) \leqslant k$. Jeśli k = 1, to ZSP jest nazywana bezpieczną. Bezpieczne znakowane sieci Petriego są zawarte w klasie sieci ograniczonych.

Problem osiągalności dla danej klasy znakowanych sieci Petriego polega na rozstrzyganiu dla dowolnej ZSP z tej klasy i dowolnego znakowania, czy znakowanie to jest osiągalne ze znakowania początkowego.

Problem żywotności dla danej klasy znakowanych sieci Petriego polega na rozstrzyganiu, czy dowolna ZSP z tej klasy jest żywa.

Problemy ograniczoności i bezpieczeństwa dla danej klasy ZSP^* są definiowane podobnie jak problem żywotności.

Dla ZSP^*, problem żywotności równoważny jest problemowi osiągalności [52]. Ponieważ problem osiągalności dla ZSP^* jest rozstrzygalny [63], a więc problem żywotności jest również rozstrzygalny.

Problem ograniczoności jest rozstrzygalny za pomocą algorytmu opartego na pojęciu drzewa pokrywalności [98].

Dla sieci ograniczonych, problemy osiągalności, żywotności i bezpieczeństwa są rozstrzygalne za pomocą grafu znakowań osiągalnych [52] ze znakowania początkowego.

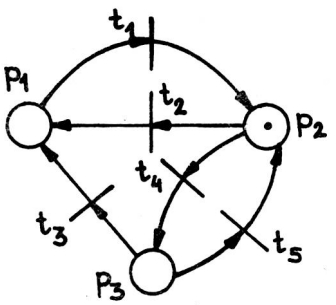
SP $\mathcal{N}$ jest strukturalnie ograniczona, jeśli ZSP^* $\mathcal{M} = <\mathcal{N}, M_0>$ są żywe dla każdego znakowania początkowego M_0. SP jest strukturalnie żywa, jeśli istnieje takie znakowanie M_0, że ZSP $\mathcal{M} = <\mathcal{N}, M_0>$ jest żywa.

Drogą skierowaną w SP nazywamy ciąg wierzchołków $v_0, v_1, \ldots, v_n$ takich, że $v_i \in P \cup T$ oraz $\langle v_{i-1}, v_i \rangle \in F$ dla $i = 1, \ldots, n$. Drogą nieskierowaną w SP nazywamy ciąg wierzchołków $v_0, v_1, \ldots, v_n$ takich, że $v_i \in P \cup T$ oraz $\langle v_{i-1}, v_i \rangle \in F$ lub $\langle v_i, v_{i-1} \rangle \in F$ dla $i = 1, \ldots, n$. SP nazywamy silnie spójną (spójną), jeśli dla dowolnej pary wierzchołków $v_i, v_j \in P \cup T$ istnieje droga skierowana z v_i do v_j (droga nieskierowana między v_i a v_j).

Drogę skierowaną w SP nazywamy drogą prostą w SP (krótko: drogą prostą), jeśli wszystkie wierzchołki v_i są różne. Obwodem w SP (krótko: obwodem) nazywamy taką drogę skierowaną w SP $v_0, v_1, \ldots, v_n$, że wierzchołki v_i są różne z wyjątkiem wierzchołków v_0, v_n ($v_0 = v_n$).

SP jest automatem [44], jeśli do każdego przejścia jest skierowany dokładnie jeden łuk oraz z każdego przejścia jest skierowany dokładnie jeden łuk, tzn. $|{}^\bullet t| = |t^\bullet| = 1$ (symbol $|X|$ oznacza liczbę elementów zbioru X).

Proces sekwencyjny jest ZSP $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$ taką, że SP jest automatem oraz $\sum_{p_i \in P} M_0(p_i) = 1$. Przykład procesu sekwencyjnego przedstawiono na rys. 1.2.



Rys. 1.2

Dla procesu sekwencyjnego sumaryczna liczba kropek w miejscach sieci jest równa jedności dla każdego znakowania osiągalnego z M_0, w danym bowiem momencie może być wykonywana co najwyżej jedna operacja.

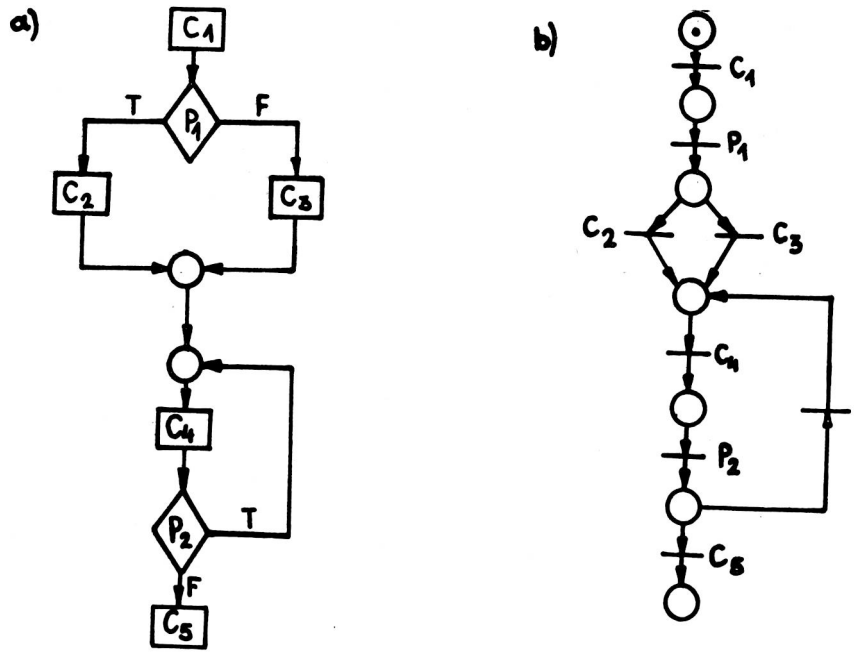
P r z y k ł a d 1.1

Sieć działań programu sekwencyjnego z rys. 1.3a można zinterpretować jako SP $\mathcal{N}$ procesu sekwencyjnego $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$ z rys. 1.3b.

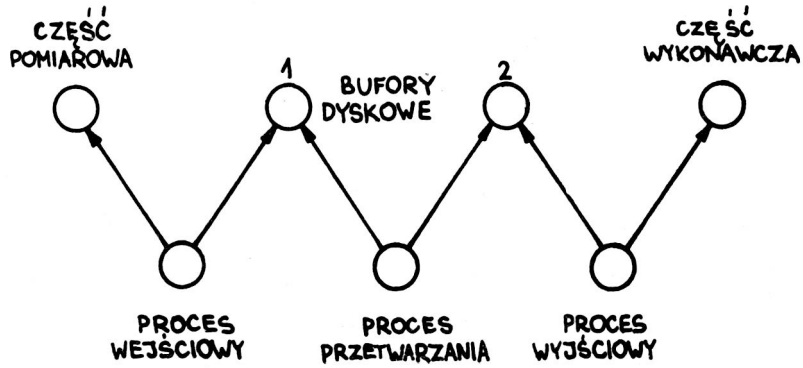
P r z y k ł a d 1.2

Rozważymy przykład sieci Petriego znacznie bardziej złożony niż poprzednie sieci. Ponieważ duża część pracy dotyczy zastosowań sieci Petriego do oceny wydajności systemów współbieżnych procesów sekwencyjnych - charakterystycznych dla systemów operacyjnych, a więc za przykład posłuży system operacyjny dla systemu komputerowego czasu rzeczywistego (rys. 1.4).

Proces wejściowy po wczytaniu danych pomiarowych tworzy z nich blok wielkości pewnej jednostki pamięci dyskowej, a następnie przesyła blok do pierwszego bufora dyskowego. Proces przetwarzania pobiera bloki z pierwszego bufora, przetwarza je, generuje dane dla części wykonaw-



Rys. 1.3

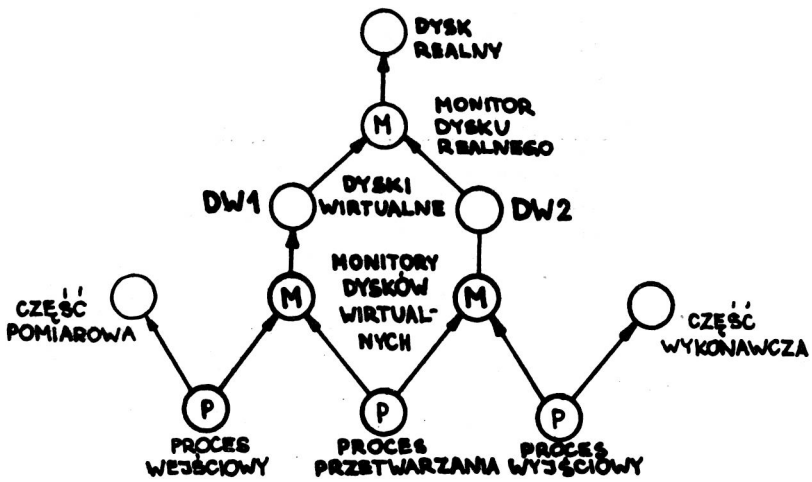


Rys. 1.4

czej, grupuje te dane w bloki i przesyła bloki do drugiego bufora dyskowego. Proces wyjściowy pobiera bloki z drugiego bufora dyskowego i wysyła dane do części wykonawczej. W danej chwili mogą być wykonywane nawet trzy procesy jednocześnie.

System operacyjny dla omawianego systemu czasu rzeczywistego może być wyrażony w takich językach programowania współbieżnego, jak Concurrent Pascal [19], [20] lub Modula-2. W celu pominięcia wielu szczegółów - mało istotnych dla modelowania systemu operacyjnego za pomocą sieci Petriego - system ten wyrazimy na stosownym poziomie abstrakcji. W celu zwiększenia czytelności prezentacji, będziemy operować konstrukcjami zaczerpniętymi z języka polskiego.

Bardziej szczegółową ilustracją analizowanego systemu czasu rzeczywistego jest rys. 1.5.



Rys. 1.5

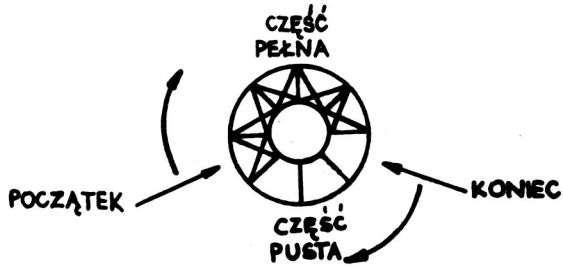
Proces wejściowy tworzy bloki z danych pomiarowych. Tworzenie jednego bloku oznaczymy napisem TWORZENIE BLOKU. Uzyskany blok jest przesyłany do bufora dyskowego (PRZESYŁANIE DO BUFORA) będącego pierwszym dyskiem wirtualnym (DW1). Proces wejściowy jest procesem cyklicznym, którego cykl składa się z operacji TWORZENIE BLOKU i PRZESYŁANIE DO BUFORA DW1.

Z bufora DW1 blok jest pobierany przez proces przetwarzania (POBIERANIE Z BUFORA). Proces ten przetwarza bloki kolejno pobierane z bufora DW1 (PRZETWARZANIE BLOKU), a następnie przesyła do drugiego dysku wirtualnego (DW2). Cykl procesu przetwarzania jest tworzony z operacji POBIERANIE Z BUFORA DW1, PRZETWARZANIE BLOKU, PRZESYŁANIE DO BUFORA DW2. Proces wyjściowy pobiera bloki z bufora DW2 i wysyła dane zawarte w bloku do części wykonawczej (WYPROWADZANIE BLOKU). Proces ten jest również procesem cyklicznym.

Wymienione procesy są zdefiniowane następująco:

```
PROCES WEJŚCIOWY
     CYKL
       TWORZENIE BLOKU
       PRZESYŁANIE DO BUFORA DW1
PROCES PRZETWARZANIA
     CYKL
       POBIERANIE Z BUFORA DW1
       PRZETWARZANIE BLOKU
       PRZESYŁANIE DO BUFORA DW2
PROCES WYJŚCIOWY
     CYKL
       POBIERANIE Z BUFORA DW2
       WYPROWADZANIE BLOKU.
```

Dyski wirtualne są buforami cyklicznymi o postaci przedstawionej na rys. 1.6.



Rys. 1.6

Wartość zmiennej POCZĄTEK wskazuje tę komórkę bufora wirtualnego, z której w danej chwili może być pobrany blok. Wartość zmiennej KONIEC wskazuje tę komórkę, do której może być przesłany blok. Zmienna DŁUGOŚĆ określa liczbę pełnych komórek bufora w danej chwili, natomiast stała POJEMNOŚĆ definiuje liczbę komórek bufora. Wartości początkowe zmiennych POCZĄTEK, KONIEC, DŁUGOŚĆ są równe zeru. Poza tym dysk wirtualny jest charakteryzowany stałą BAZA określającą początkową komórkę bufora.

Z każdego z dysków wirtualnych DW1, DW2 korzystają po dwa procesy. Dwa procesy użytkowujące jeden bufor nie mogą z niego korzystać jednocześnie. Monitor dysku wirtualnego gwarantuje, że w danej chwili na dysku tym może być wykonywana najwyżej jedna z dwu operacji PRZESYŁANIE DO BUFORA lub POBIERANIE Z BUFORA. Monitor ten zawiera dwie procedury o nazwach identycznych z nazwami wymienionych operacji - wzajemnie wykluczające się w zakresie dostępu do dysku wirtualnego.

Procedury te mają następującą postać:

```
PROCEDURA PRZESYŁANIE DO BUFORA
     JEŚLI DŁUGOŚĆ = POJEMNOŚĆ TO WSTRZYMAJ PROCES PRZESYŁAJĄCY
     PISZ(BLOK, BAZA + KONIEC)
     KONIEC := (KONIEC+1) MOD POJEMNOŚĆ
     DŁUGOŚĆ := DŁUGOŚĆ+1
PROCEDURA POBIERANIE Z BUFORA
     JEŚLI DŁUGOŚĆ = 0 TO WSTRZYMAJ PROCES POBIERAJĄCY
     CZYTAJ(BLOK, BAZA + POCZĄTEK)
     POCZĄTEK := (POCZĄTEK+1) MOD POJEMNOŚĆ
     DŁUGOŚĆ := DŁUGOŚĆ-1
```

Symbolem MOD oznaczona jest operacja modulo.

Instrukcje PISZ(...), CZYTAJ(...) zawarte w procedurach dysku wirtualnego powodują wywołanie procedur o następującej postaci:

```
PROCEDURA PISZ(BLOK, NR BLOKU)
     ŻĄDANIE DYSKU REALNEGO
     ŻAPIS(BLOK, NR BLOKU)
     ZWOLNIENIE DYSKU REALNEGO
PROCEDURA CZYTAJ(BLOK, NR BLOKU)
     ŻĄDANIE DYSKU REALNEGO
     ODCZYT(BLOK, NR BLOKU)
     ZWOLNIENIE DYSKU REALNEGO
```

Procedury PISZ(...), CZYTAJ(...) wymagają dostępu do dysku realnego. Monitor dysku wirtualnego gwarantuje wzajemne wykluczanie procesów w zakresie dostępu do dysku wirtualnego. Ponieważ w systemie istnieją dwa dyski wirtualne, a więc należy wykluczyć jednoczesny dostęp dwu dysków wirtualnych do dysku realnego. Wykluczanie to jest realizowane za pomocą monitora dysku realnego. Monitor ten zawiera dwie procedury ŻĄDANIE DYSKU REALNEGO, ZWOLNIENIE DYSKU REALNEGO (wywoływane w procedurach PISZ(...), CZYTAJ(...)). Pierwsza z tych procedur umożliwia uzyskanie przez dysk wirtualny dostępu do dysku realnego, jeżeli drugi dysk wirtualny nie korzysta z dysku realnego. W przeciwnym razie proces żądający dostępu zostaje wstrzymany. Druga procedura zwalnia dysk realny oraz umożliwia uzyskanie dostępu do dysku realnego przez wstrzymany proces lub przez proces, który jako kolejny będzie żądał dostępu.

Rozpatrywany system jest systemem hierarchicznym, w którym można wyróżnić trzy poziomy:

1) poziom procesów,
2) poziom dysków wirtualnych,
3) poziom dysku realnego.

Na potrzeby interpretacji sieci Petriego, wyrażającej system operacyjny, wprowadzimy następujące skróty:

POBIERANIE - PROCEDURA POBIERANIE Z BUFORA
PRZESYŁANIE - PROCEDURA PRZESYŁANIE DO BUFORA
PISZ BLOK - PROCEDURA PISZ(BLOK, NR BLOKU)
NAST(KONIEC) - KONIEC := (KONIEC+1) MOD POJEMNOŚĆ
NAST(DŁUGOŚĆ) - DŁUGOŚĆ := DŁUGOŚĆ+1
CZYTAJ BLOK - PROCEDURA CZYTAJ(BLOK, NR BLOKU)
NAST(POCZĄTEK) - POCZĄTEK := (POCZĄTEK+1) MOD POJEMNOŚĆ
POPRZ (DŁUGOŚĆ) - DŁUGOŚĆ := DŁUGOŚĆ-1
DR - dysk realny

Rysunek 1.7 ilustruje sieć Petriego opisującą współpracę procesu wejściowego i procesu przetwarzania realizowaną na podstawie bufora DW1.

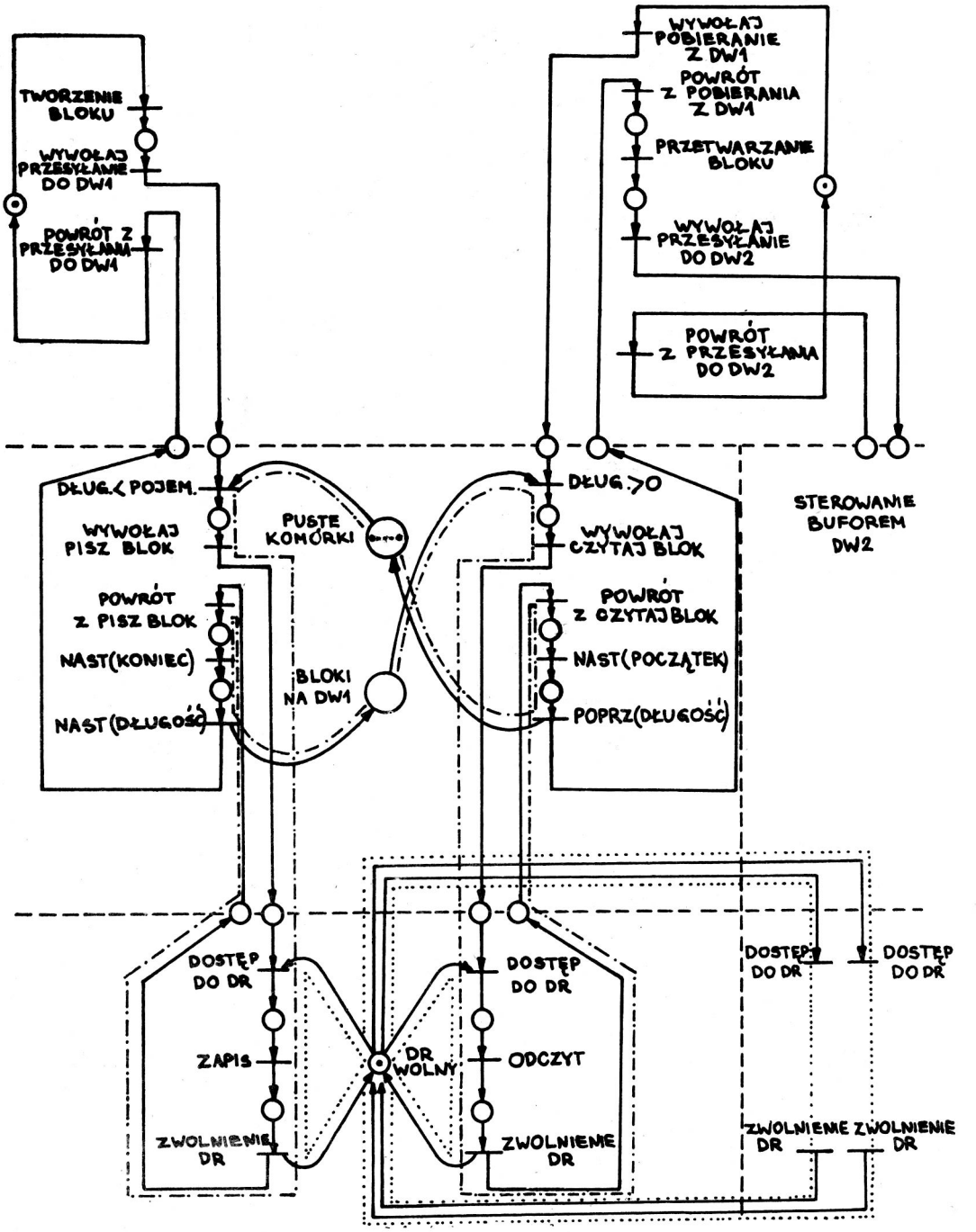
Dla znakowania początkowego w miejscu PUSTE KOMÓRKI znajdują się znaczniki w liczbie równej stałej POJEMNOŚĆ. Liczba znaczników w miejscu BLOKI NA DW1 określa liczbę bloków znajdujących się w danej chwili w buforze DW1. Warunkiem koniecznym zapalenia przejścia DŁUG < POJEM jest istnienie co najmniej jednej pustej komórki w DW1. Warunkiem koniecznym zapalenia przejścia DŁUG > 0 jest istnienie co najmniej jednego bloku w buforze DW1. Do miejsc PUSTE KOMÓRKI i BLOKI NA DW1 są skierowane łuki z ostatnich przejść procedur PRZESYŁANIE, POBIERANIE, ponieważ procedury te wzajemnie wykluczają się w zakresie dostępu do procesora.

Procedury monitora dysku wirtualnego wzajemnie wykluczają się w zakresie dostępu do tego dysku, tzn. nie mogą być one wykonywane jednocześnie. W celu uniknięcia zbyt szczegółowych rozważań - zagadnienie to nie jest odwzorowane w sieci ilustrowanej rys. 1.7.

1.2. Sieci Petriego z czynnikiem czasu

Dodanie czynnika czasu do sieci Petriego umożliwia ocenę wydajności systemów. Czas wykonania instrukcji, operacji itd. może być wyrażony zmienną losową [4],[89] lub za pomocą liczb rzeczywistych [102], [111]. Istnieją dwa podejścia do modelowania czasu za pomocą liczb rzeczywistych. W pierwszym podejściu czas modelowany jest jedną liczbą rzeczywistą [102], natomiast w drugim - czas charakteryzują dwie liczby [85]: pierwsza z nich określa najwcześniejszy, a druga - najpóźniejszy moment wystąpienia zdarzenia. Drugie podejście przydatne jest w opisie mechanizmu time-out'u.

Istnieją dwa sposoby przypisywania czasu elementom sieci Petriego. W pierwszym - czas przyporządkowany jest przejściom [4], [89], [102], natomiast w drugim - miejscom [111]. Analiza równoważności obu sposobów zawarta jest w pracy [113]. W pracy [113] przedstawiono transfor-


TWORZENIE BLOKU
WYWOŁAJ PRZESYŁANIE DO DW1
POWRÓT Z PRZESYŁANIA DO DW1
WYWOŁAJ POBIERANIE Z DW1
POWRÓT Z POBIERANIA Z DW1
PRZETWARZANIE BLOKU
WYWOŁAJ PRZESYŁANIE DO DW2
POWRÓT Z PRZESYŁANIA DO DW2
DŁUG.< POJEM.
WYWOŁAJ PISZ BLOK
POWRÓT Z PISZ BLOK
NAST(KONIEC)
NAST(DŁUGOŚĆ)
PUSTE KOMÓRKI
BLOKI NA DW1
DŁUG.>0
WYWOŁAJ CZYTAJ BLOK
POWRÓT Z CZYTAJ BLOK
NAST(POCZĄTEK)
POPRZ(DŁUGOŚĆ)
STEROWANIE BUFOREM DW2
DOSTĘP DO DR
ZAPIS
ZWOLNIENIE DR
DR WOLNY
DOSTĘP DO DR
ODCZYT
ZWOLNIENIE DR
DOSTĘP DO DR
DOSTĘP DO DR
ZWOLNIENIE DR
ZWOLNIENIE DR

Rys. 1.7

mację sieci z czasem przypisanym miejscom w sieć z czasem przypisanym przejściom oraz transformację odwrotną. W literaturze dominuje przypisywanie czasu przejściom; co wynika z następującego faktu. Ogólna teoria sieci jest oparta na modelu systemów, zwanym siecią warunkowo-zdarzeniową [107]. W modelu tym warunki reprezentowane są miejscami, natomiast zdarzenia - przejściami. Wielu autorów definiuje czas za pomocą zdarzeń [9]. Dlatego przyporządkowanie czasu - przejściom wydaje się bardziej naturalne i jest przyjęte w pracy. Metodologię podobną do zawartej w prezentowanej pracy można zastosować do sieci z czasem przypisanym miejscom.

1.2.1. Deterministyczne sieci Petriego

Wektor palenia sekwencji palenia $\sigma = t_{i_1},\ldots,t_{i_j},\ldots,t_{i_s}$ jest wektorem $f(\sigma)$ o $|T|$ składowych ($|T|$ jest liczbą przejść ZSP) takich, że i-ta składowa $f(t_i)$ jest liczbą paleń przejścia t_i w sekwencji σ.

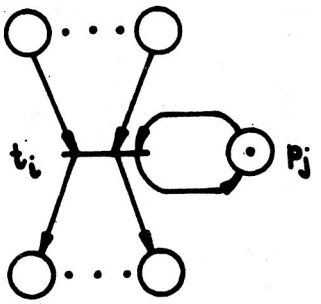
Deterministyczną siecią Petriego (DSP) jest para $\mathcal{M}^t = <\mathcal{M}, \mathcal{T}>$, gdzie $\mathcal{M}$ jest ZSP, $\mathcal{T} : T \longrightarrow R_+^r$ (R_+^r jest zbiorem nieujemnych liczb wymiernych) jest funkcją czasu palenia. W chwili początkowej $\tau = 0$, znakowanie DSP zadane jest znakowaniem początkowym M_0, natomiast żadne z przejść się nie pali.

Gdy przejście t_i jest przygotowane do palenia, wówczas może nastąpić jego palenie. W chwili rozpoczęcia palenia przejścia t_i, z każdego z miejsc $p_j \in {}^\bullet t_i$ usuwanych jest $W(<p_j,t_i>)$ znaczników. Faza palenia przejścia t_i trwa przez czas $\mathcal{T}(t_i)$. W chwili zakończenia palenia przejścia t_i do każdego z miejsc $p_k \in t_i^\bullet$ jest dodawanych $W(<t_i,p_k>)$ znaczników.

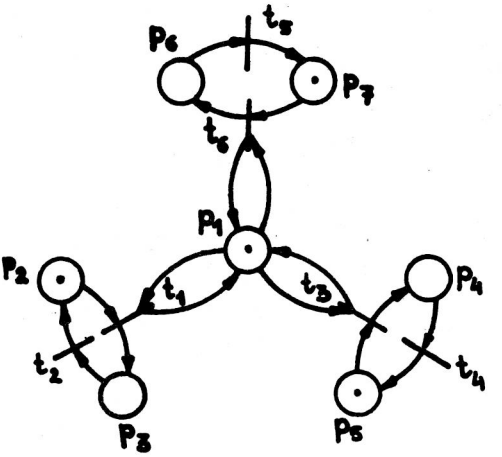
Jeśli w czasie palenia przejścia t_i istnieją warunki do ponownego palenia tego przejścia, zdarzenie to może nastąpić. Jest to przypadek palenia wielokrotnego. Gdy wymagamy, aby przejście t_i w danej chwili było palone tylko jednokrotnie, dobudowujemy pętlę zawierającą przejście t_i oraz miejsce p_j z jednym znacznikiem dla M_0 (rys. 1.8).

Palenie przejść w DSP można wyrazić za pomocą diagramu palenia. Rozważmy sieć z rys. 1.9. Czasy palenia poszczególnych przejść są zadane następująco: $\mathcal{T}(t_1) = 2s$, $\mathcal{T}(t_2) = 4s$, $\mathcal{T}(t_3) = 3s$, $\mathcal{T}(t_4) = 2s$, $\mathcal{T}(t_5) = 1s$, $\mathcal{T}(t_6) = 5s$. Przykładowy przebieg palenia przejść tej sieci obrazuje rys. 1.10. Każdemu z przejść jest przypisana jedna oś na diagramie palenia. Strzałki służą wyrażeniu relacji przyczynowości palenia przejść. Czasową sekwencją palenia x jest zbiór $|T|$ elementowy, którego elementy są parami

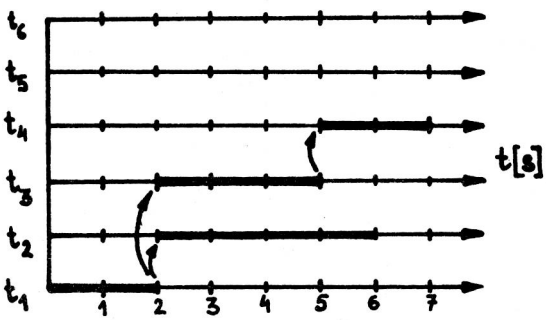
$$<t_i, t_i(1)t_i(2)\ldots t_i(j)\ldots>,$$



Rys. 1.8



Rys. 1.9



Rys. 1.10

gdzie $t_i(1)t_i(2)...t_i(j)...$ jest sekwencją chwil rozpoczęcia kolejnych paleń przejścia t_i, czyli

$$x = \left\{ < t_i, t_i(1)t_i(2)...t_i(j)... > : t_i \in T \right\}.$$

Dogodną formą reprezentacji czasowej sekwencji palenia jest diagram palenia. Wektor palenia czasowej sekwencji palenia x jest wektorem g(x) o |T| składowych (|T| jest liczbą przejść DSP takich, że i-ta składowa $g(t_i)$ jest liczbą paleń przejścia t_i w sekwencji x.

Obecnie przedstawimy kryteria oceny zachowania przejściowego i cyklicznego DSP.

1.2.1.1. Zachowanie przejściowe

Zachowanie przejściowe DSP jest wyrażone za pomocą wektora H o $|T|$ składowych będących liczbami naturalnymi. Symbolem S(H) będziemy oznaczać zbiór takich czasowych sekwencji palenia, których wektor palenia jest równy wektorowi H. Dla czasowej sekwencji palenia x - wielkość $R(x,t_i)$ jest określona następująco:

$$P(x,t_i) = \begin{cases} t_i(1), & \text{jeśli przejście } t_i \text{ pali się w sekwencji} \\ & x \text{ co najmniej jeden raz,} \\ \infty & \text{w przeciwnym przypadku.} \end{cases}$$

Dla sekwencji x - wielkość $Z(x,t_i)$ jest opisana zależnością:

$$Z(x,t_i) = \begin{cases} t_i(s) + \mathcal{T}(t_i), & \text{jeśli przejście } t_i \text{ pali się w sekwencji } x \text{ } s > 0 \text{ razy,} \\ 0, & \text{w przeciwnym przypadku,} \end{cases}$$

$(t_i(s) + \mathcal{T}(t_i))$ jest chwilą zakończenia palenia przejścia t_i po raz ostatni w sekwencji x, jeśli przejście to paliło się co najmniej jeden raz). Czasem wykonania czasowej sekwencji palenia x jest liczba

$$\max_{t_i \in T} Z(x,t_i) - \min_{t_i \in T} R(x,t_i).$$

Czas wykonania czasowej sekwencji palenia $x \in S(H)$ oznaczać będziemy symbolem $\beta(H,x)$.

W praktycznych zagadnieniach oceny wydajności systemów komputerowych interesują nas zagadnienia dwu typów:

- zagadnienie weryfikacji
- zagadnienie optymalizacji.

W pierwszym przypadku sprawdzamy, czy system spełnia ograniczenia nałożone na jego wydajność (np. generuje odpowiedź w przedziale czasu o zadanej długości), natomiast w przypadku drugim - wyznaczamy optymalną wydajność (np. minimum średniego czasu odpowiedzi).

DECYZYJNY PROBLEM CZASU WYKONANIA (DPCW):

Dane:

DSP $\mathcal{M}^t = < \mathcal{M}, \mathcal{T} >$,

wektor $H \in N^{|T|}$

liczba $\beta \in R_+^r$.

Pytanie:

Czy istnieje taka czasowa sekwencja palenia x, że $x \in S(H)$ oraz $\beta(H,x) \leqslant \beta$?

OPTYMALIZACYJNY PROBLEM CZASU WYKONANIA (OPCW):

Dane:

DSP $\mathcal{M}^t = < \mathcal{M}, \mathcal{T} >$,

wektor $H \in N^{|T|}$,

Zadanie:

Wyznaczyć taki czas wykonania $\beta(H,x^*)$ czasowej sekwencji palenia $x^* \in S(H)$, jeśli taka sekwencja istnieje, że

$$\beta(H,x^*) = \min_{x \in S(H)} \beta(H,x).$$

1.2.1.2. Zachowanie cykliczne

Zachowanie cykliczne jest wyrażane za pomocą T-niezmiennika. T-niezmiennik jest wektorem I o |T| składowych, takim że

$$\sum_{t_i \in {}^\bullet p_j} W(<t_i,p_j>)I(t_i) = \sum_{t_i \in p_j^\bullet} W(<p_j,t_i>)I(t_i)$$

dla każdego $p_j \in P$,

gdzie $I(t_i)$ jest składową dla przejścia t_i będącą liczbą naturalną. Liczba $W(<t_i,p_j>)$ jest równa liczbie znaczników dodawanych do miejsca p_j w wyniku palenia przejścia t_i, natomiast liczba $W(<p_j,t_i>)$ określa liczbę znaczników pobieranych z miejsca p_j na rzecz palenia przejścia t_i. Rozważmy sekwencję palenia σ, której wektor palenia $f(\sigma)$ jest równy T-niezmiennikowi I. Liczba $W(<t_i,p_j>)I(t_i)$ zadaje liczbę znaczników dodawanych do miejsca p_j, dzięki paleniu przejścia t_i w sekwencji σ. Liczba $\sum_{t_i \in {}^\bullet p_j} W(<t_i,p_j>)I(t_i)$ określa liczbę znaczników dodawanych do miejsca p_j, dzięki paleniu przejść zawartych w sekwencji σ. Dla T-niezmiennika I liczba ta równa jest liczbie znaczników pobieranych z miejsca p_j na rzecz palenia przejść zawartych w sekwencji σ. Zatem T-niezmiennik równy jest wektorowi palenia dla sekwencji palenia σ, po której paleniu jest osiągane takie same znakowanie jak przed paleniem.

Dla czystych SP^*, w interpretacji macierzowej - T-niezmiennik I jest wektorem kolumnowym o |T| składowych spełniających równanie

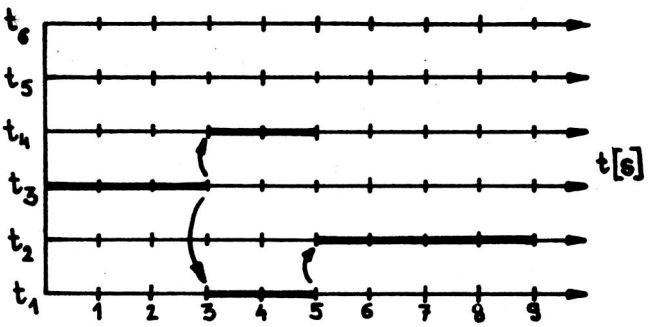
$$C \cdot I = O,$$

gdzie: C - macierz incydencji SP,
$I(t_i) \in N$.

Przestrzeń rozwiązań równania C I zawiera $\rho = |T| - r[C]$ liniowo niezależnych wektorów, gdzie r[C] jest rzędem macierzy C. Niech M będzie znakowaniem, dla którego jest palona sekwencja σ o wektorze palenia równym T-niezmiennikowi I oraz niech M' będzie znakowaniem osiągalnym po tej sekwencji. Dla danych wielkości spełnione są zależności:

$$M' = M + C\,I = M.$$

Obecnie przeanalizujemy takie czasowe sekwencje palenia, które nie zmieniają znakowania DSP. Czasowa sekwencja palenia ilustrowana rys. 1.10 nie zmienia znakowania DSP z rys. 1.9. Wielokrotne powtórzenie tej sekwencji również zachowuje znakowanie. Czas wykonania tej sekwencji równy jest 7s. Przejścia t_1, t_3 muszą palić się w trybie wzajemnego wykluczania. Przy zmianie kolejności palenia tych przejść, możemy uzyskać czasową sekwencję palenia obrazowaną rys. 1.11 (zachowującą znakowanie) o czasie wykonania równym 9s.

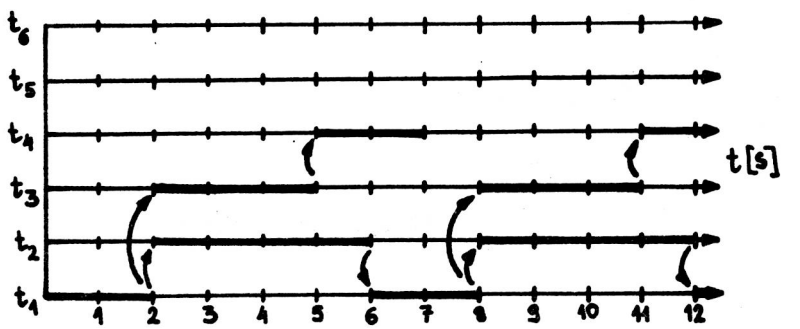


Rys. 1.11

Wektor palenia dla każdej z dwu czasowych sekwencji palenia wyrażonych rys. 1.10, 1.11 równy jest następującemu T-niezmiennikowi I':

$I'(t_1) = 1$, $I'(t_2) = 1$, $I'(t_3) = 1$, $I'(t_4) = 1$, $I'(t_5) = 0$, $I'(t_6) = 0$.

Z kolei przeanalizujemy czasową sekwencję palenia z rys. 1.12.



Rys. 1.12

Dla tej sekwencji znakowania sieci w wybranych chwilach są następujące:

Chwila	p_1	p_2	p_3	p_4	p_5	p_6	p_7
0s	1	1	0	0	1	0	1
2s	1	0	1	0	1	0	1
5s	1	0	0	1	0	0	1
6s	1	1	0	0	0	0	1
7s	0	0	0	0	1	0	1
8s	1	0	1	0	1	0	1

Chwila	p_1	p_2	p_3	p_4	p_5	p_6	p_7
0_+s	0	0	0	0	1	0	1
2_+s	0	0	0	0	0	0	1
5_+s	1	0	0	0	0	0	1
6_+s	0	0	0	0	0	0	1
7_+s	0	0	0	0	1	0	1
8_+s	0	0	0	0	0	0	1

Stanem DSP jest wektor s o $|P| + |T|$ składowych, którego k-ta składowa S_k równa jest $S_i = M(p_i)$ dla $i \in \{1,\ldots,|P|\}$, gdzie $M(p_i)$ jest znakowaniem miejsca p_i oraz $S_{|P|+j} = \tau(t_j)$ dla $j \in \{1,\ldots,|T|\}$, gdzie $\tau(t_j)$ jest wektorem czasów pozostałych do zakończenia palenia przejścia t_j. Składowa $\tau(t_j)$ jest wektorem, ponieważ przejście t_j może w danej chwili palić się wielokrotnie. Stany DSP z rys. 1.9 dla czasowej sekwencji palenia ilustrowanej rys. 1.12 w chwilach 2s i 8s są równe. Możemy zatem, począwszy od chwili 2s, uzyskać odtwarzanie stanu DSP co 6s poprzez powtarzanie zachowania sieci z przedziału [2s,8s[. Jednak nigdy nie zostanie powtórnie osiągnięte znakowanie początkowe M_0. Ponadto w przedziale czasu [2s,8s[liczby paleń poszczególnych przejść są równe przyporządkowanym im składowym T-niezmiennika I'.

Dla SP może istnieć wiele T-niezmienników. Różne T-niezmienniki mogą wyrażać różne tryby pracy systemu modelowanego za pomocą SP.

Podane rozważania stanowią uzasadnienie dla dwu różnych definicji czasu cyklu.

Liczba $\gamma_1(I,x)$ jest czasem cyklu pierwszego rodzaju czasowej sekwencji palenia x dla T-niezmiennika I, jeśli sekwencja x spełnia warunek

$$\gamma_1(I,x) = \lim_{k\to\infty} \frac{t_i(k\ I(t_i))}{k} \text{ dla każdego } t_i \in T,$$

gdzie $t_i(k\ I(t_i))$ jest chwilą, w której rozpoczynane jest $(k\ I(t_i))$-te palenie przejścia t_i, pod warunkiem, że znakowanie sieci w chwilach $k\ \gamma_1(I,x)$ $(k \in \{1,2,\ldots\})$ jest równe znakowaniu początkowemu.

Liczba $\gamma_2(I,x)$ jest czasem cyklu drugiego rodzaju sekwencji palenia x dla T-niezmiennika I, jeśli sekwencja x spełnia warunek

$$\gamma_2(I,x) = \lim_{k\to\infty} \frac{t_i(k\ I(t_i))}{k} \text{ dla każdego } t_i \in T.$$

DECYZYJNY PROBLEM CZASU CYKLU PIERWSZEGO RODZAJU ($DPCC_1$):

Dane:

DSP $\mathcal{M}^t = <\mathcal{M}, \mathcal{T}>$
T-niezmiennik $I \in N^{|T|}$,
liczba $\gamma \in R_+^r$.

Pytanie:

Czy istnieje taka czasowa sekwencja palenia x, że

$$\gamma_1(I,x) \leqslant \gamma ?$$

DECYZYJNY PROBLEM CZASU CYKLU DRUGIEGO RODZAJU ($DPCC_2$) jest podobnie zdefiniowany, z tym że nierówność ma postać $\gamma_2(I,x) \leqslant \gamma$.

OPTYMALIZACYJNY PROBLEM CZASU CYKLU PIERWSZEGO RODZAJU ($OPCC_1$):

Dane:

DSP $\mathcal{M}^t = <\mathcal{M}, \mathcal{T}>$
T-niezmiennik $I \in N^{|T|}$.

Zadanie:

Wyznaczyć taki czas cyklu $\gamma_1(I,x^*)$ czasowej sekwencji palenia $x^* \in S_1(I)$, jeśli taka sekwencja istnieje, że

$$\gamma_1(I,x^*) = \min_{x \in S_1(I)} \gamma_1(I,x),$$

gdzie $S_1(I)$ jest zbiorem czasowych sekwencji palenia, dla których istnieje $\gamma_1(I,x)$.

OPTYMALIZACYJNY PROBLEM CZASU CYKLU DRUGIEGO RODZAJU ($OPCC_2$) jest analogicznie sformułowany.

Często rozwiązanie $OPCC_1$ możemy uzyskać poprzez wykorzystanie wyników teorii szeregowania deterministycznego dla kryterium: minimum czasu wykonania systemu operacji.

Uwaga

Prawdziwa jest relacja

$$\min_{x \in S_2(I)} \gamma_2(I,x) \leqslant \min_{x \in S_1(I)} \gamma_1(I,x).$$

Wartość $\gamma_2(I',x) = 6s$ dla czasowej sekwencji palenia obrazowanej rys. 1.12 jest minimalną dla DSP z rys. 1.9 i T-niezmiennika I', ponieważ w sieci tej istnieje obwód $p_2t_1p_3t_2p_2$ o sumarycznym czasie palenia równym 6s oraz sumarycznej liczbie kropek równej jedności. Czas cyklu dla tego obwodu nie może być mniejszy niż 6s.

W pracy będziemy się zajmowali problemami czasu cyklu drugiego rodzaju. W celu uproszczenia oznaczeń zamiast symboli $DPCC_2$, $OPCC_2$ stosować będziemy symbole DPCC, OPCC.

1.2.2. Stochastyczne sieci Petriego

Sieci z losowym wyborem przejść do palenia wprowadzono w pracy [110]. Sieci z czasem palenia przejścia zadanym za pomocą zmiennej losowej, zwane stochastycznymi sieciami Petriego, zdefiniowano w pracach [90], [92]. W pracy [1] jest zawarta systematyka stochastycznych sieci Petriego z uwzględnieniem różnych rozkładów zmiennych losowych czasów palenia przejść i różnych trybów wyboru przejść do palenia.

W pracy będziemy stosować uogólnione stochastyczne sieci Petriego (USSP*) [4]. Przejścia USSP należą do dwu klas: przejść bezzwłocznych i przejść czasowych. Przejścia bezzwłoczne oznaczone są cienką kreską, natomiast przejścia czasowe - grubą. Przejście bezzwłoczne pali się w zerowym czasie, natomiast czas palenia przejścia czasowego jest wyrażony zmienną losową o rozkładzie wykładniczym. Niech liczba λ_i (parametr rozkładu wykładniczego) będzie intensywnością palenia przejścia czasowego t_i. Jeśli zbiór H przejść przygotowanych do palenia zawiera jedynie przejścia czasowe, to przejście $t_i \in H$ pali się z prawdopodobieństwem

$$\frac{\lambda_i}{\sum_{t_k \in H} \lambda_k} \tag{1.1}$$

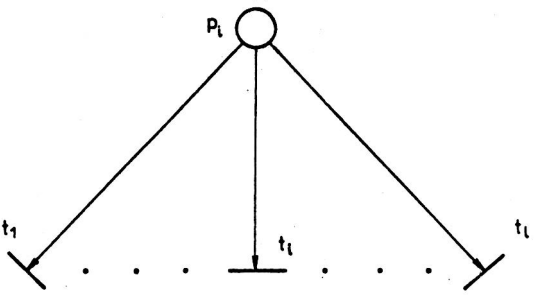
Jeśli zbiór H zawiera przejścia bezzwłoczne i czasowe, to mogą palić się jedynie przejścia bezzwłoczne. Jeśli zbiór H zawiera więcej niż jedno przejście bezzwłoczne, to palone przejście jest wybierane zgodnie z rozkładem prawdopodobieństwa określonym na zbiorze przygotowanych do palenia przejść bezzwłocznych.

Sposób interpretacji sieci działań programu sekwencyjnego za pomocą pojęcia procesu sekwencyjnego przedstawiliśmy w punkcie 1.1. W dalszej części pracy będziemy używać pojęcia procesu sekwencyjnego, a nie sieci działań.

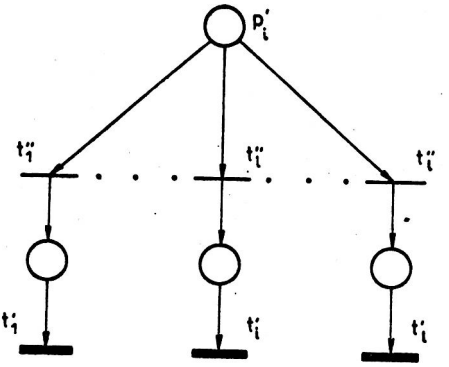
Rozważmy fragment procesu sekwencyjnego związany z wyborem kierunku przepływu sterowania ilustrowany rys. 1.13.

Zakładamy, że na podstawie analizy probabilistycznej procesu sekwencyjnego dane jest prawdopodobieństwo $p(t_i)$ palenia przejścia t_i ($i \in \{1,\ldots,l\}$), jeśli znacznik jest w miejscu p_0. Spełniony musi być warunek $\sum_{i=1}^{l} p(t_i) = 1$. Niech czas palenia przejścia t_i będzie zadany zmienną losową o rozkładzie wykładniczym z parametrem λ_i. Sieć z rys. 1.13 można zinterpretować jako USSP z rys. 1.14.

Przejścia t_i'' ($i \in \{1,\ldots,l\}$) są przejściami bezzwłocznymi, natomiast przejścia t_i' ($i \in \{1,\ldots,l\}$) - przejściami czasowymi. Jeśli



Rys. 1.13



Rys. 1.14

kropka jest w miejscu p_i', to przejście t_i'' ($i \in \{1,\ldots,l\}$) pali się z prawdopodobieństwem $p(t_i'') = p(t_i)$. Czas palenia przejścia t_i' ($i \in \{1,\ldots,l\}$) jest opisany zmienną losową o rozkładzie wykładniczym z parametrem λ_i.

Zachowanie USSP jako funkcja czasu jest równoważne zachowaniu w czasie stochastycznego procesu punktowego (SPP). SPP przebywa niezerowy czas w znakowaniach, dla których są przygotowane jedynie przejścia czasowe oraz przechodzi w czasie zerowym przez znakowania, dla których przygotowane jest co najmniej jedno przejście bezzwłoczne. Pierwsze znakowania są nazywane znakowaniami (stanami) uchwytnymi, natomiast drugie - zanikającymi.

Obecnie rozważymy macierz prawdopodobieństw przejść dla USSP.

W SPP dla USSP można wyeksponować włożony łańcuch Markowa (WŁM). W celu uzyskania WŁM o skończonej liczbie stanów, wymagamy, aby USSP była ograniczona. Zgodnie z pracą [4] macierz U prawdopodobieństw przejść WŁM dla USSP może być wyrażona w postaci:

$$U = A + B = \begin{matrix} \overset{K_v}{\longleftrightarrow} & \overset{K_t}{\longleftrightarrow} \end{matrix} \begin{bmatrix} C & D \\ O & O \end{bmatrix} + \begin{matrix} \overset{K_v}{\longleftrightarrow} & \overset{K_t}{\longleftrightarrow} \end{matrix} \begin{bmatrix} O & O \\ E & F \end{bmatrix} \begin{matrix} \updownarrow K_v \\ \updownarrow K_t \end{matrix} \qquad (1.2)$$

gdzie K_v (K_t) jest liczbą stanów zanikających (uchwytnych) WŁM (jak również SPP).

W celu zmniejszenia złożoności problemu analizy stanów uchwytnych, pragniemy znaleźć macierz prawdopodobieństw przejść między stanami uchwytnymi z uwzględnieniem przejść przez stany zanikające.

Prawdopodobieństwo w_{ij} przejścia ze stanu uchwytnego i do stanu uchwytnego j spełnia zależność

$$w_{ij} = f_{ij} + \sum_{v \in V} e_{iv} p_{vj}, \tag{1.3}$$

gdzie V jest zbiorem stanów zanikających, f_{ij}, e_{iv} są składowymi macierzy F, E, p_{vj} reprezentuje prawdopodobieństwo przejścia SPP ze stanu zanikającego v do stanu uchwytnego j w dowolnej liczbie kroków, ale z przechodzeniem jedynie przez stany zanikające.

Macierz

$$G^k = \sum_{h=0}^{k-1} C^h D \tag{1.4}$$

określa prawdopodobieństwo osiągania dowolnego stanu uchwytnego po trajektorii o długości nie większej niż k kroków i rozpoczynającej się w dowolnym stanie zanikającym, pod warunkiem, że stanami pośrednimi są tylko stany zanikające [4].

W przypadkach systemów procesów sekwencyjnych rozpatrywanych w pracy - nie ma pętli w zbiorze stanów zanikających. Dlatego istnieje taka liczba $k_0 < K_v$, że $C^k = 0$ dla każdego $k > k_0$. Możemy zatem macierz W prawdopodobieństw przejść między stanami uchwytnymi, z uwzględnieniem przejść przez stany zanikające jako pośrednie, wyrazić w postaci

$$W = F + E \sum_{h=0}^{k_0} C^h D. \tag{1.5}$$

Obecnie zajmiemy się analizą czasowej złożoności obliczeniowej algorytmu wyznaczania wyrażenia (w skrócie: złożoności obliczeniowej wyrażenia) $\sum_{h=0}^{k_0} C^h D$. Wyrażenie to określa macierz prawdopodobieństw osiągania dowolnego stanu uchwytnego po trajektorii o dowolnej długości rozpoczynającej się w dowolnym stanie zanikającym, pod warunkiem, że stanami pośrednimi są tylko stany zanikające. Rząd złożoności sumy $\sum_{h=0}^{k_0} C^h D$ jest równy rzędowi złożoności wyrażenia $C^{k_0} D$. Złożoność tego wyrażenia wyznaczona jest poprzez złożoność wyrażeń $C' = C^{k_0}$ i $C'D$. Złożoność obliczeniowa mnożenia macierzy A_{nxm}, B_{mxp} jest równa $O(nmp)$. Dla iloczynu C^{k_0} - złożoność wynosi $O(K_v^3 k_0)$ dla $k_0 > 1$, natomiast dla iloczynu $C'D$ - złożoność równa jest $O(K_v^2 K_t)$.

Po usunięciu stanów zanikających z WŁM otrzymujemy <u>zredukowany włożony łańcuch Markowa</u> (ZWŁM).

Rozważać będziemy dwie sytuacje. W pierwszej, wszystkie stany ZWŁM, z wyłączeniem jedynego stanu końcowego, są stanami przejściowymi

[118]. W drugiej, stan początkowy jest osiągalny z niezerowym prawdopodobieństwem z każdego stanu osiągalnego ze stanu początkowego. Podane sytuacje są nazywane odpowiednio: zachowaniem przejściowym i zachowaniem cyklicznym USSP.

1.2.2.1. Zachowanie przejściowe

Macierz uzyskaną z macierzy W, poprzez usunięcie wiersza i kolumny przyporządkowanych stanowi końcowemu, oznaczymy symbolem $\bar{W}$. Składowa $(\bar{W}^n)_{ij}$ macierzy $\bar{W}^n$ wyraża średnią liczbę wystąpień stanu j ZWŁM na (n+1)-szej pozycji w sekwencji n+1 stanów pod warunkiem, że stanem początkowym jest stan i. Niech składowa z_{ij} macierzy Z opisuje średnią liczbę wystąpień stanu j na wszystkich pozycjach sekwencji rozpoczynającej się ze stanu i. Zatem

$$Z = \sum_{n=0}^{\infty} \bar{W}^n,$$

gdzie

$$(\bar{W}^0)_{ij} = \begin{cases} 1, & \text{jeśli } i=j, \\ 0, & \text{jeśli } i \neq j. \end{cases}$$

Po rozwinięciu powyższej równości macierzowej otrzymujemy

$$Z = I + \bar{W} + \bar{W}^2 + \ldots,$$

ponieważ $\bar{W}^0 = I$. Ten szereg macierzowy jest zbieżny do $(I - \bar{W})^{-1}$ [118]. Zatem

$$Z = (I - \bar{W})^{-1}.$$

Rozważany ZWŁM ma jeden stan końcowy. Średni czas osiągania stanu końcowego t przy starcie ze stanu początkowego i jest dany zależnością

$$\tau = \sum_{\substack{k=1 \\ k \neq t}}^{K_t} z_{ik}\, ST_k,$$

gdzie K_t - liczba stanów ZWŁM (liczba stanów uchwytnych WŁM),
ST_k - średni czas pobytu ZWŁM w stanie k równy

$$ST_k = \frac{1}{\sum_{t_r \in H_k} \lambda_r}$$

H_k - zbiór przejść czasowych przygotowanych do palenia dla znakowania reprezentowanego stanem k.

1.2.2.2. Zachowanie cykliczne

W celu wyznaczenia rozkładu prawdopodobieństwa stanów ZWŁM w stanie stacjonarnym rozwiązujemy równanie:

$$Y = Y W.$$

Średni ułamek czasu pobytu [4] w stanie uchwytnym i jest opisany równością

$$v_i = \frac{ST_i \, y_i}{\sum_{j \in U} ST_j \, y_j}$$

gdzie ST_i - średni czas pobytu ZWŁM w stanie i,
U - zbiór stanów uchwytnych,
y_i, y_j - składowe wektora Y.

W celu wyznaczenia średniego czasu cyklu systemu, wybierzmy jeden ze stanów, a mianowicie stan k jako stan odniesienia. Iloraz

$$q_{jk} = \frac{y_j}{y_k}$$

określa średnią liczbę przejść przez stan j między dwoma kolejnymi przejściami przez stan k. Średni czas cyklu systemu określa formuła

$$\tau_{śr} = \sum_{j \in U} q_{jk} \, ST_j.$$

1.2.3. Mieszane sieci Petriego

Dla mieszanej sieci Petriego (MSP) czas palenia przejścia t_i jest zadany nieujemną liczbą wymierną $\mathcal{T}(t_i)$. Wybór przejść do palenia spośród zbioru przejść przygotowanych do palenia jest wyrażony dyskretnym rozkładem prawdopodobieństwa. Stan mieszanej sieci Petriego jest wyrażony znakowaniem wraz z czasami pozostałymi do zakończenia palenia przejść palonych w danej chwili. Chwile, w których następuje zmiana znakowania tworzą włożony łańcuch Markowa z czasem dyskretnym. Wymagamy, aby łańcuch ten miał skończony zbiór stanów. Dlatego możemy MSP* analizować analogicznie do USSP* - badając podobne charakterystyki.

1.3. Złożoność obliczeniowa problemów kombinatorycznych

Obecnie przedstawimy podstawowe pojęcia z zakresu złożoności obliczeniowej problemów kombinatorycznych - na podstawie [16]. Pojęcia te definiuje się dla klasy tak zwanych problemów decyzyjnych. Problemy te są formułowane w postaci pytania, na które odpowiedź brzmi "tak" lub

"nie". Oprócz klasy problemów decyzyjnych ważna jest klasa problemów optymalizacyjnych dotyczących ekstremalizacji funkcji celu. Złożoność obliczeniowa obu klas problemów może być badana w podobny sposób. Wynika to z możliwości kojarzenia problemu decyzyjnego z danym problemem optymalizacyjnym. Na przykład OPTYMALIZACYJNEMU PROBLEMOWI CZASU CYKLU (pkt. 1.2.1.2) możemy przyporządkować DECYZYJNY PROBLEM CZASU CYKLU.

Problem decyzyjny nie jest obliczeniowo trudniejszy niż odpowiadający mu pierwotny problem optymalizacyjny, jeżeli dla danego rozwiązania problemu optymalizacyjnego można relatywnie prosto obliczyć wartość funkcji celu. Określenie relatywnie prosto oznacza tu złożoność obliczeniową nie większą niż dla problemu optymalizacyjnego, które to wymaganie jest zwykle spełnione. Zatem w celu wykazania "łatwości" problemu decyzyjnego wystarczy dowieść "łatwość" problemu optymalizacyjnego. Natomiast z "trudności" problemu decyzyjnego wynika "trudność" problemu optymalizacyjnego. Intuicyjne pojęcia "łatwości" i "trudności" problemu zostaną sprecyzowane w dalszej części tego punktu.

Ponieważ podstawowe pojęcia teorii złożoności obliczeniowej są zdefiniowane dla problemu decyzyjnego, a więc teraz pojęcie to scharakteryzujemy dokładniej.

Przez problem decyzyjny Π rozumiemy zbiór parametrów, które nie muszą mieć nadanych wartości oraz pytanie, na które odpowiedź brzmi "tak" lub "nie". Po ustaleniu wartości wszystkich parametrów danego problemu Π otrzymujemy instancję (konkretny problem), którą oznaczamy symbolem I. Zbiór instancji problemu Π oznaczać będziemy symbolem D_Π.

Dane instancji $I \in D_\Pi$ zapisuje się (koduje) za pomocą skończonego łańcucha $x(I)$ symboli zgodnie z ustaloną regułą kodowania. Rozmiar instancji I jest długością łańcucha danych oznaczoną symbolem $N(I)$. Reguły kodowania powinny być jednoznaczne i zwięzłe, co oznacza, że nie powinny powodować wykładniczego wzrostu rozmiaru kodowanej instancji w stosunku do innych reguł kodowania.

W praktyce korzystnie jest wyrazić rozmiar instancji za pomocą jednego, dwu, rzadko kiedy większej liczby parametrów, określających liczbę elementów zbioru istotnego dla danego problemu. W przypadkach rozpatrywanych w pracy, parametry te są liczbą miejsc, przejść sieci, liczbą procesów sekwencyjnych.

Ponieważ zależy nam na algorytmach najszybszych, a więc badać będziemy złożoność czasową algorytmów.

Funkcją czasowej złożoności obliczeniowej (złożonością obliczeniową) algorytmu A rozwiązującego problem Π jest funkcja przyporządkowująca każdej wartości rozmiaru $N(I)$ instancji $I \in D_\Pi$ maksymalną liczbę elementarnych kroków maszyny cyfrowej potrzebną do rozwiązania instancji o tym rozmiarze za pomocą algorytmu A.

Przyjmujemy jednorodne koszty wykonania operacji elementarnych.

Funkcja f(k) jest rzędu g(k), co zapisujemy O(g(k)), jeśli istnieje stała c taka, że $|f(k)| \leq c|g(k)|$ dla prawie wszystkich wartości k.

Algorytmem wielomianowym nazywamy algorytm, którego złożoność obliczeniowa jest O(p(k)), gdzie p jest pewnym wielomianem, k - rozmiarem rozwiązywanej instancji. Każdy algorytm, którego złożoność obliczeniowa nie może być tak ograniczona - nazywany jest wykładniczym.

Algorytmy wielomianowe są traktowane jako efektywne, natomiast wykładnicze - jako nieefektywne.

Pojęcia teorii złożoności obliczeniowej są formułowane za pomocą maszyn Turinga. Definicja deterministycznej jednotaśmowej maszyny Turinga (DMT) jest zawarta w pracy [39].

Niedeterministyczna jednotaśmowa maszyna Turinga (NDMT) składa się z DMT i modułu generującego. Program NDMT jest określony tak samo jak DMT, lecz jego wykonanie dla łańcucha danych x(I) instancji I jest inaczej przeprowadzane niż dla DMT, przebiega bowiem w dwu etapach. W etapie pierwszym moduł generujący zapisuje w dowolny sposób na taśmie łańcuch S symboli ze skończonego zbioru symboli taśmy. W drugim etapie NDMT sprawdza, tak jak wykonywany jest program przez DMT, czy wygenerowany łańcuch S spełnia warunki określone w pytaniu instancji I. Dla jednej instancji I może istnieć wiele łańcuchów. Twierdzimy, że NDMT rozwiązuje problem decyzyjny Π, jeśli dla każdej instancji $I \in D_\Pi$ są spełnione dwa warunki:

1. Jeśli odpowiedź dla I brzmi "tak", to zostanie wygenerowany łańcuch S, który wraz z x(I) spowoduje, że po wykonaniu programu przez NDMT maszyna ta osiągnie stan końcowy q_{tak}.

2. Jeśli odpowiedź brzmi "nie", to dla każdego wygenerowanego łańcucha S albo NDMT osiągnie stan q_{nie}, albo etap sprawdzania nie zostanie zakończony w skończonym czasie.

Twierdzimy, że NDMT rozwiązuje problem decyzyjny Π w (co najwyżej) wielomianowym czasie, jeśli dla każdej instancji $I \in D_\Pi$, dla której odpowiedź brzmi "tak", zostanie wygenerowany taki łańcuch S, że czas wykonania etapu generacji i etapu sprawdzania zakończonego odpowiedzią "tak" przez NDMT (dla I oraz S) jest O(p(N(I))) dla pewnego wielomianu p.

Klasę P tworzą wszystkie problemy decyzyjne, które w co najwyżej wielomianowym czasie rozwiązuje DMT. Klasa NP zawiera wszystkie problemy decyzyjne, które w co najwyżej wielomianowym czasie rozwiązuje NDMT. Prawdziwa jest zależność $P \subseteq NP$. Najprawdopodobniej klasa P jest właściwą podklasą klasy NP, lecz jest to problem nadal otwarty.

Transformacją wielomianową problemu Π_2 do problemu Π_1 ($\Pi_2 \propto \Pi_1$) jest funkcja $f : D_{\Pi_2} \longrightarrow D_{\Pi_1}$, która spełnia warunki:

1. Dla każdej instancji $I_2 \in D_{\Pi_2}$ odpowiedź brzmi "tak", wtedy i tylko wtedy, gdy dla instancji $f(I_2)$ odpowiedź brzmi również "tak".

2. Czas obliczania funkcji f przez DMT dla każdej instancji $I_2 \in D_{\Pi_2}$ jest ograniczony od góry przez wielomian od $N(I_2)$.

Problem decyzyjny Π_1 nazywamy NP-zupełnym, jeśli $\Pi_1 \in NP$ i dla każdego innego problemu decyzyjnego $\Pi_2 \in NP$, $\Pi_2 \propto \Pi_1$.

Zatem, jeśli istniałby algorytm wielomianowy do rozwiązywania jakiegokolwiek problemu NP-zupełnego, to każdy problem z klasy NP (w tym również problemy NP-zupełne) mógłby być rozwiązany za pomocą algorytmu wielomianowego. Z bezskuteczności poszukiwań algorytmu wielomianowego dla któregokolwiek problemu NP-zupełnego wynika, że prawdopodobnie wszystkie problemy NP-zupełne można rozwiązać tylko przy użyciu algorytmów wykładniczych.

Z danych definicji wynika, że dla udowodnienia NP-zupełności problemu decyzyjnego Π wystarczy dowieść, że $\Pi \in NP$ oraz przetransformować wielomianowo do problemu Π dowolny znany problem NP-zupełny.

W celu zbadania danego problemu z punktu widzenia złożoności obliczeniowej, staramy się znaleźć dla niego optymalny deterministyczny algorytm wielomianowy lub wykazać trudność tego problemu. Aby wykazać trudność, wystarczy wielomianowo przetransformować pewien problem NP-zupełny do decyzyjnej wersji danego problemu.

Do klasy problemów NP-zupełnych należą najtrudniejsze problemy klasy NP. Obecnie klasę tę scharakteryzujemy dokładniej.

Niektóre problemy NP-zupełne można, dla spotykanych w praktyce danych, rozwiązać stosunkowo niewielkim nakładem czasu. Pewne problemy NP-zupełne można rozwiązać za pomocą tzw. algorytmów pseudowielomianowych. Złożoność obliczeniowa algorytmów pseudowielomianowych jest ograniczona od góry poprzez wielomian zależny od rozmiaru instancji oraz od maksymalnej wartości występujących w tym problemie liczb $Max(I)$. Ponieważ w praktyce liczba $Max(I)$ przyjmuje skończone wartości, a więc algorytmy te mają stosunkowo korzystne własności złożoności obliczeniowej. Nie są to jednak algorytmy wielomianowe. Algorytmy pseudowielomianowe można ewentualnie zbudować tylko dla problemów liczbowych Π, czyli takich, dla których nie istnieje wielomian p taki, że $Max(I) \leqslant p(N(I))$ dla każdego $I \in D_{\Pi}$.

Gdy uwzględni się powyższe rozważania, można w klasie problemów NP-zupełnych wyodrębnić problemy silnie NP-zupełne.

Dla dowolnego problemu decyzyjnego Π i dowolnego wielomianu p, określonego dla liczb całkowitych, niech Π_p oznacza podproblem problemu Π otrzymany przez ograniczenie D_{Π} tylko do tych instancji, dla których $Max(I) \leqslant p(N(I))$. Zatem Π_p nie jest problemem liczbowym.

Problem decyzyjny Π jest silnie NP-zupełny, jeśli należy do NP i istnieje wielomian p określony dla liczb całkowitych, dla którego Π_p jest NP-zupełny.

Z definicji tej wynika, że jeśli problem Π jest NP-zupełny i nie jest problemem liczbowym, to jest silnie NP-zupełny, ponadto jeśli problem Π jest silnie NP-zupełny, to istnienie dla niego algorytmu pseudowielomianowego jest równoważne istnieniu algorytmów wielomianowych dla wszystkich problemów NP-zupełnych, czyli równości P = NP, co jest bardzo mało prawdopodobne.

Dowód silnej NP-zupełności na podstawie definicji jest trudny. Dla uproszczenia dowodu wprowadzono pojęcie transformacji pseudowielomianowej.

Pseudowielomianową transformacją problemu decyzyjnego Π_2 do problemu decyzyjnego Π_1 nazywamy funkcję $f : D_{\Pi_2} \longrightarrow D_{\Pi_1}$, taką, że:

1. Dla każdej instancji $I_2 \in D_{\Pi_2}$ odpowiedź brzmi "tak", wtedy i tylko wtedy, gdy dla $f(I_2)$ brzmi także "tak",

2. Funkcja f może być obliczona przez DMT w czasie ograniczonym od góry przez wielomian zależny od dwóch zmiennych: $N_2(I_2)$ i $Max(I_2)$.

3. Istnieje wielomian q_1 taki, że dla każdej $I_2 \in D_{\Pi_2}$ $q_1(N_1(f(I_2))) \geq N_2(I_2)$,

4. Istnieje wielomian q_2 taki, że dla każdej $I_2 \in D_{\Pi_2}$ $Max_1(f(I_2)) \leq q_2(Max_2(I_2), N_2(I_2))$.

W porównaniu z definicją transformacji wielomianowej osłabiony jest warunek wielomianowego czasu realizacji algorytmu obliczającego funkcję f. Z drugiej strony wymaga się, aby rozmiar instancji nie zmalał wykładniczo (3), a największa wartość liczbowa zawarta w danych nie wzrosła wykładniczo (4).

W pracy [39] wykazano, że jeśli problem Π_2 jest silnie NP-zupełny i $\Pi_1 \in$ NP oraz istnieje pseudowielomianowa transformacja problemu Π_2 do Π_1, to problem Π_1 jest silnie NP-zupełny. Przykładem problemu silnie NP-zupełnego, jest PROBLEM TRÓJPODZIAŁU. Problem ten będziemy stosować w dowodach trudności problemów rozważanych w pracy - transformując go pseudowielomianowo do tych problemów.

Obecnie przedstawimy PROBLEM TRÓJPODZIAŁU [39].

Dane:

X - zbiór zawierający 3k elementów oznaczonych symbolem x_i,

$x_i \in N \setminus \{0\}$ (zbiór dodatnich liczb naturalnych),

$B \in N \setminus \{0\}$,

elementy $x_i \in X$ spełniają relacje

$B/4 < x_i < B/2$ oraz $\sum_{x_i \in X} x_i = k\,B$.

Pytanie:

Czy zbiór X może być podzielony na k podzbiorów $S_1,\ldots,S_k$ takich, że jeśli $i \neq j$, to $S_i \cap S_j = \emptyset$ oraz dla $1 \leqslant i \leqslant k$ $\sum_{x_j \in S_i} x_j = B$?

Obecnie przejdziemy do przedstawienia definicji problemu NP-trudnego i silnie NP-trudnego. W pracy pojęcia te będziemy stosowali w odniesieniu do problemów decyzyjnych.

Problemem przeszukiwania Π nazywamy zbiór instancji D_Π i zdefiniowany dla każdej instancji $I \in D_\Pi$ zbiór rozwiązań $Z_\Pi(I)$. Twierdzimy, że algorytm rozwiązuje problem przeszukiwania Π, jeśli dla każdej instancji $I \in D_\Pi$ odpowiedź brzmi "nie", jeśli $Z_\Pi(I)$ jest pusty, natomiast w przeciwnym razie algorytm znajduje jedno z rozwiązań należących do zbioru $Z_\Pi(I)$.

Jako problemy przeszukiwania możemy przedstawić zarówno problemy optymalizacyjne, jak i decyzyjne. W przypadku problemu decyzyjnego - zbiór $Z_\Pi(I) = \{\text{"tak"}\}$, jeśli odpowiedź dla instancji I brzmi "tak", natomiast $Z_\Pi(I)$ jest pusty w przeciwnym razie.

Pojęcie wielomianowej transformacji Turinga jest uogólnieniem pojęcia transformacji wielomianowej. Wielomianową transformacją Turinga problemu przeszukiwania Π_1 do problemu przeszukiwania Π_2 (oznaczenie $\Pi_1 \propto_T \Pi_2$) nazywamy algorytm A rozwiązujący problem Π_1 przy użyciu hipotetycznej procedury P rozwiązującej problem Π_2, przy czym czas wykonania algorytmu A przez DMT jest wielomianowy, jeśli procedura P może być wykonana przez DMT w czasie wielomianowym.

Problem przeszukiwania Π_2 jest NP-trudny, jeśli istnieje taki NP-zupełny problem decyzyjny Π_1, dla którego zachodzi $\Pi_1 \propto_T \Pi_2$.

Problemy NP-trudne (optymalizacyjne czy decyzyjne) można badać w podobny sposób jak problemy NP-zupełne. Niektóre liczbowe problemy NP-trudne można rozwiązać za pomocą algorytmów pseudowielomianowych. Podobnie jak dla problemów NP-zupełnych jest definiowana silna NP-trudność. Problem przeszukiwania jest silnie NP-trudny, jeśli jego podproblem spełniający dla pewnego wielomianu ograniczenie $p(N(I)) \geqslant Max(I)$ jest NP-trudny.

2. ZŁOŻONOŚĆ OBLICZENIOWA PROBLEMU CZASU CYKLU DLA WYBRANYCH KLAS DETERMINISTYCZNYCH SIECI PETRIEGO

Deterministyczne sieci Petriego (DSP^*) można zastosować w projektowaniu komputerów równoległych, takich jak: komputery z bardzo długim słowem maszynowym, komputery potokowe, macierzowe, systoliczne, stero-

wane przepływem danych. Prace [45], [46] zawierają rezultaty z zakresu optymalizacji komputera potokowego z zastosowaniem tych sieci.

DSP^* są przydatne w projektowaniu systemów czasu rzeczywistego.

Rozdział ten stanowi syntezę następujących prac autora [73], [75], [78], [79].

Badać będziemy żywe i ograniczone ZSP^*. Jeśli ZSP nie jest żywa, to pewna jej część może zostać zablokowana. W praktyce projektant pragnie budować systemy pozbawione blokad. Jeśli ZSP nie jest ograniczona, to nie może być zrealizowana fizycznie - ze względu na potrzebę nieograniczonej liczby zasobów. Dodatkowo wyróżniać będziemy bezpieczne ZSP^*. Znaczenie bezpiecznych ZSP^* wynika z następujących faktów. W przypadkach rzeczywistych, często pojemność buforów jest równa jedności, jednostki sprzętowe mogą wykonywać w danej chwili tylko jedną operację, istnieje tylko jedna kopia programu. Elementami składowymi systemów procesów są procesy sekwencyjne, z których każdy zawiera dokładnie jedną kropkę.

Rozważać będziemy spójne SP^*. Jeśli sieć składa się z podsieci izolowanych, to każda podsieć może być analizowana odrębnie. Prawdziwe jest następujące twierdzenie.

Twierdzenie 2.1 [102]

Jeśli dla spójnej SP $\mathcal{N}$ istnieje takie znakowanie M_0, że ZSP $\mathcal{M} = <\mathcal{N}, M_0>$ jest żywa i ograniczona, to SP $\mathcal{N}$ jest silnie spójna.

Dlatego analizować będziemy jedynie silnie spójne SP^*.

W pierwszych dwu punktach rozdziału przedstawiamy definicje klas sieci wraz z relacjami między poszczególnymi klasami. W punkcie trzecim badamy rozstrzygalność problemu czasu cyklu, w punkcie czwartym - złożoność obliczeniową problemu czasu cyklu, w punkcie piątym - oszacowanie minimalnego czasu cyklu. Rozdział zakończony jest podsumowaniem.

2.1. Wybrane klasy sieci

SP^* można podzielić na dwie wzajemnie dopełniające się klasy, a mianowicie na SP^* P-niezmiennicze i nie P-niezmiennicze.

Wektor J o $|P|$ składowych jest nazywany P-niezmiennikiem, jeśli dla każdego przejścia t_j jest spełniony warunek

$$\sum_{p_i \in {}^\bullet t_j} J(p_i)\, W(<p_i, t_j>) = \sum_{p_i \in t_j^\bullet} J(p_i)\, W(<t_j, p_i>),$$

gdzie $J(p_i)$ jest nieujemną składową całkowitą dla miejsca p_i. Składowa $J(p_i)$ określa wagę kropki w miejscu p_i. Dla P-niezmiennika J suma ważona wszystkich kropek w ZSP równa $\sum_{p_i \in P} J(p_i)\, M(p_i)$ jest stała nieza-

leżnie od znakowania M, osiągalnego ze znakowania początkowego. Własność ta jest przydatna w badaniu ZSP* modelujących systemy rzeczywiste.

Dla czystych SP*, w interpretacji macierzowej - P-niezmiennik jest wektorem kolumnowym o |P| składowych spełniających równanie

$$J^T C = 0,$$

gdzie: C - macierz incydencji SP,
J^T - wektor transponowany wektora J,
$J(p_i) \in N$.

Niech M będzie dowolnym znakowaniem osiągalnym ze znakowania początkowego M_0. Dla danych wielkości jest spełniona równość

$$J^T M = J^T M_0.$$

SP jest <u>siecią P-niezmienniczą</u>, jeśli istnieje P-niezmiennik J ze wszystkimi składowymi dodatnimi. W przeciwnym razie SP nazywamy nie P--niezmienniczą. W SP możemy wyróżniać podsieci będące sieciami P-niezmienniczymi. Zbiór miejsc podsieci P-niezmienniczych zawiera te miejsca, dla których składowe pewnego P-niezmiennika są dodatnie.

Ranga P-niezmienniczych SP* jest większa niż SP* nie P-niezmienniczych - ze względu na znaczenie w praktyce. Sieci P-niezmiennicze można zdekomponować na podsieci P-niezmiennicze. Podsieci P-niezmiennicze mogą wyrażać cykliczne procesy sekwencyjne i równoległe (procesy cykliczne często występują w systemach operacyjnych), podsystemy zarządzania zasobami odnawialnymi (zasoby przydzielane i zwracane), komunikację między procesami czy procesorami, której protokół wymaga potwierdzenia odbioru komunikatu, podsystemy wykrywania i naprawy uszkodzeń.

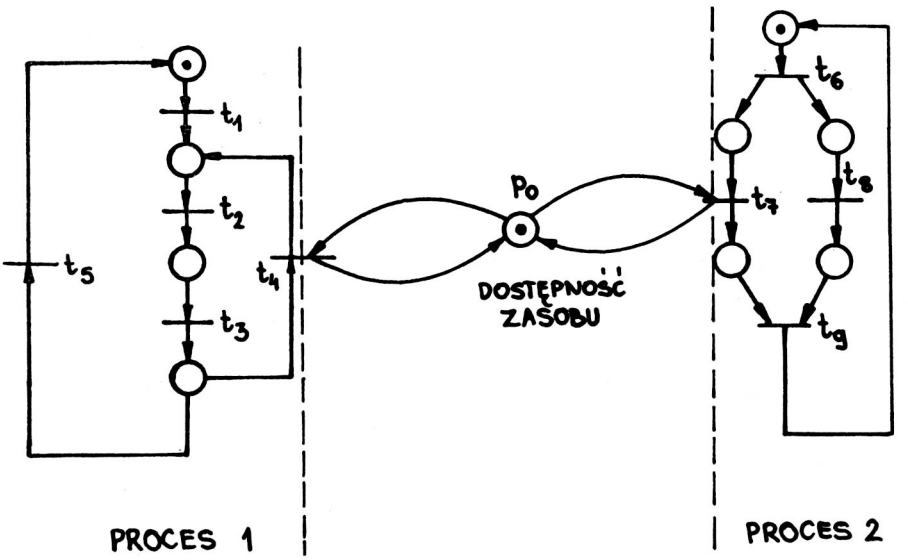
P r z y k ł a d 2.1

Przeanalizujemy system dwu procesów ubiegających się o dzielony zasób z rys. 2.1.

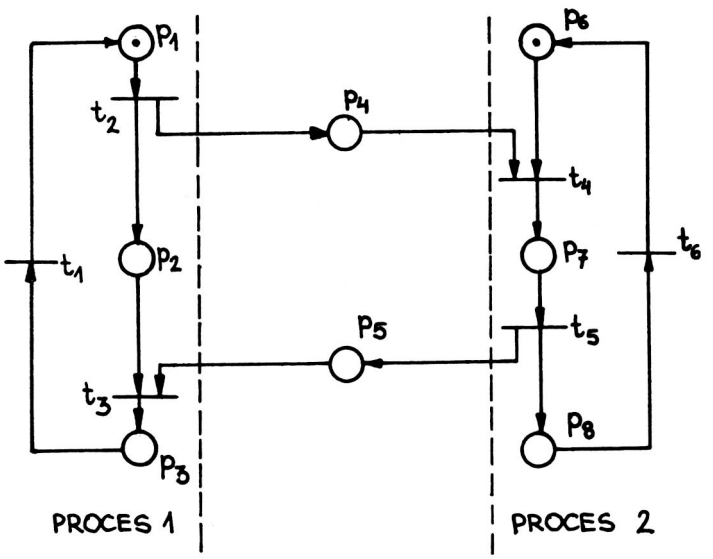
Proces 1 jest procesem sekwencyjnym, natomiast Proces 2 - procesem równoległym ze względu na możliwość jednoczesnego palenia przejść t_7, t_8. Do realizacji operacji odwzorowywanych przejściami t_4, t_7 niezbędny jest pewien zasób. Jeśli w miejscu p_0 znajduje się znacznik, to znaczy że zasób ten jest dostępny. Powyższa SP jest P-niezmiennicza. Wśród jej podsieci P-niezmienniczych na szczególną uwagę zasługują: sieci procesu sekwencyjnego i równoległego oraz sieć opisująca użytkowanie zasobu, która zawiera miejsce p_0, przejścia t_4, t_7 i łuki między miejscem p_0 a przejściami t_4, t_7.

P r z y k ł a d 2.2

Rys. 2.2 ilustruje komunikację między dwoma procesami z potwierdzaniem odbioru komunikatu.



Rys. 2.1



Rys. 2.2

Przejścia sieci reprezentują:

t_1 - produkcję komunikatu,

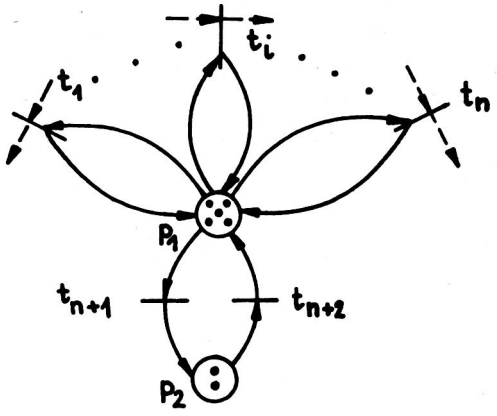
t_2 - wysłanie komunikatu,
t_3 - odbiór potwierdzenia,
t_4 - odbiór komunikatu,
t_5 - wysłanie potwierdzenia,
t_6 - konsumpcję komunikatu.

Znacznik w miejscu p_4 wyraża komunikat wysłany, ale jeszcze nie odebrany, natomiast znacznik w miejscu p_5 oznacza wysłane, ale jeszcze nie odebrane potwierdzenie.

Jedną z podsieci P-niezmienniczych jest obwód
$p_1, t_2, p_4, t_4, p_7, t_5, p_5, t_3, p_3, t_1, p_1$.

P r z y k ł a d 2.3

Rys. 2.3 przedstawia sieciową reprezentację podsystemu przydzielania i naprawy zasobów jednego typu.



Rys. 2.3

Przejścia $t_1, \ldots, t_n$ wyrażają użytkowanie sprawnego i dostępnego zasobu na rzecz realizowanych procesów. Przejście t_{n+1} odwzorowuje uszkodzenie zasobu, natomiast przejście t_{n+2} - naprawę. Liczba znaczników w miejscu p_1 równa jest liczbie sprawnych zasobów, natomiast liczba znaczników w miejscu p_2 wskazuje na liczbę uszkodzonych zasobów.

W SP z rys. 1.7, wyrażającej system operacyjny systemu czasu rzeczywistego, istnieją następujące podsieci P-niezmiennicze:

- podsieć procesu wejściowego,
- podsieć procesu przetwarzania,
- podsieć dysku wirtualnego DW1 (linia —·—·—),
- podsieć dysku realnego (linia ·····).

Dodatkowo rangę P-niezmienniczych $SP^{\mathbb{N}}$ podkreślają dwa następujące twierdzenia.

Twierdzenie 2.2 [112]
Jeśli SP jest strukturalnie żywa i strukturalnie ograniczona, to jest P-niezmiennicza.

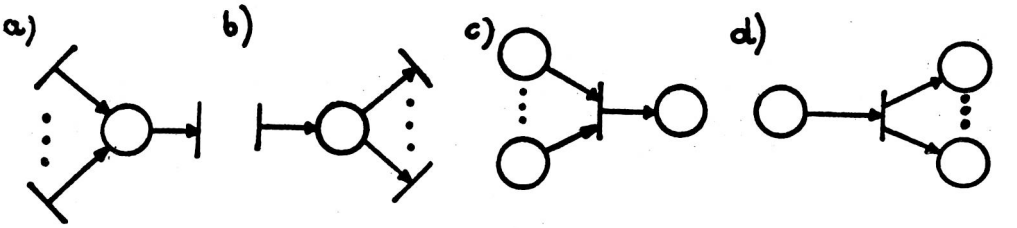
W przypadku wielu ważnych klas $SP^{\mathbb{N}}$ i $ZSP^{\mathbb{N}}$ krotności łuków są równe jedności. Dlatego SP dla tych klas może być zdefiniowana jako trójka $\mathcal{N} = <P,T,F>$.

Twierdzenie 2.3
Silnie spójna SP o krotności łuków równej jedności jest P-niezmiennicza.

Dowód
Miejscu p_i przypiszmy liczbę $J(p_i)$ równą liczbie obwodów, do których miejsce to należy. Ze względu na silną spójność SP $\mathcal{N}$ - każde z jej miejsc należy do co najmniej jednego obwodu. Zatem $J(p_i) > 0$ dla każdego miejsca $p_i \in P$. Niech $p_i \in {}^\bullet t_j$, a zatem wśród miejsc wyjściowych przejścia t_j musi być miejsce należące do tego samego obwodu co miejsce p_i. Na podstawie poprzedniego zdania i definicji liczby $J(p_i)$ otrzymujemy równość $\sum_{p_i \in {}^\bullet t_j} J(p_i) = \sum_{p_k \in t_j^\bullet} J(p_k)$. Ze względu na tę równość i relację $J(p_i) > 0$ dla każdego miejsca $p_i \in P$, stwierdzamy, że SP $\mathcal{N}$ jest P-niezmiennicza. ∎

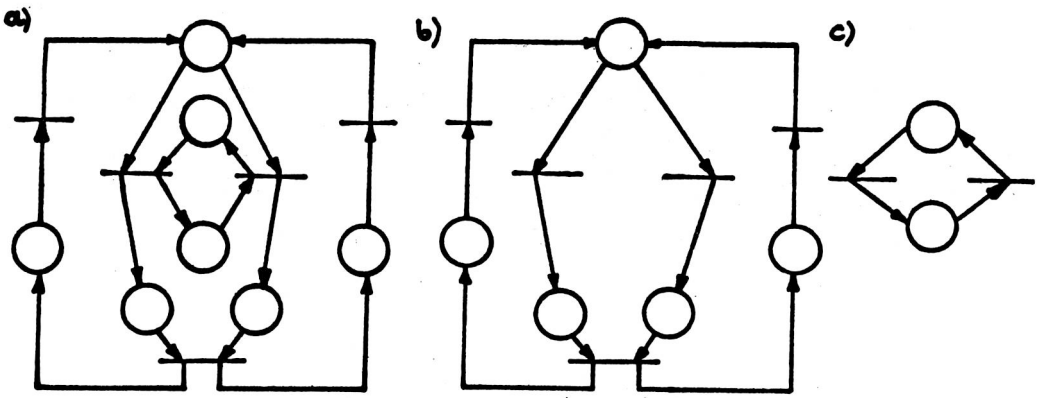
SP $\mathcal{N}$ nazywamy grafem synchronizacji [52], jeśli dla każdego miejsca p prawdziwa jest zależność $|{}^\bullet p| = |p^\bullet| = 1$. ZSP $\mathcal{M} = <\mathcal{N}, M_0>$, której SP $\mathcal{N}$ jest grafem synchronizacji, zwana jest grafem znakowanym. SP jest automatem [52], jeśli $|{}^\bullet t| = |t^\bullet| = 1$ dla każdego przejścia t. SP nazywamy SP bez konfliktów w przód (w tył), jeśli dla każdego miejsca p prawdziwa jest zależność $|p^\bullet| = 1$ ($|{}^\bullet p| = 1$). SP nazywamy SP bez współbieżności w przód (w tył), jeśli dla każdego przejścia t prawdziwa jest równość $|t^\bullet| = 1$ ($|{}^\bullet t| = 1$). W SP a) bez konfliktów w przód, b) bez konfliktów w tył, c) bez współbieżności w przód, d) bez współbieżności w tył występują fragmenty przedstawione na odpowiednich częściach rys. 2.4.

Dla SP $\mathcal{N} = <P,T,F>$ i zbioru $Y \subset X = P \cup T$ wprowadzamy oznaczenia ${}^\bullet Y = \{x \in X : (\exists y \in Y)(<x,y> \in F)\}$, $Y^\bullet = \{x \in X : (\exists y \in Y)(<y,x> \in F)\}$. Podsiecią zamkniętą sieci $\mathcal{N} = <P,T,F>$ nazywamy silnie spójną SP $\mathcal{N}' = <P',T',F'>$, gdzie $P' \subset P, T' \subset T$, ${}^\bullet P' = P'^\bullet = T'$ (dla $p_i \in P'$ jeśli $<t_j,p_i> \in F$, lub $<p_i,t_j> \in F$, to $t_j \in T'$), $F' = F \cap [(P' \times T') \cup (T' \times P')]$



Rys. 2.4

Stąd podsieć zamknięta jest jednoznacznie określona przez zbiór miejsc $P' \subset P$. Dla sieci z rys. 2.5a, przykładami podsieci zamkniętych są podsieci reprezentowane rys. 2.5b oraz 2.5c.

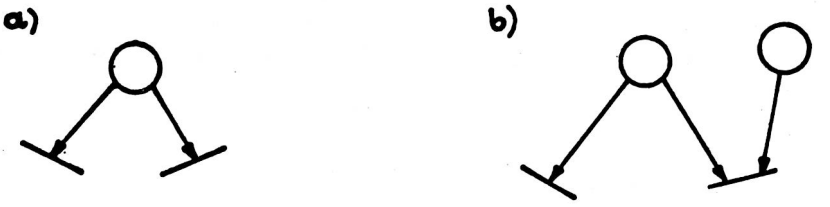


Rys. 2.5

Podsieć zamknięta jest <u>minimalną podsiecią zamkniętą</u>, jeśli usunięcie dowolnego fragmentu tej podsieci powoduje, że wynikowa podsieć nie jest podsiecią zamkniętą. <u>SP $\mathcal{N}$ jest pokryta przez zbiór zamkniętych podsieci</u> $\mathcal{N}_i = \langle P_i, T_i, F_i \rangle$, $i \in K$, jeśli $\mathcal{N} = \langle \bigcup_{i \in K} P_i, \bigcup_{i \in K} T_i, \bigcup_{i \in K} F_i \rangle$. SP $\mathcal{N}$ nazywamy <u>SP dekomponowalną na automaty</u> [102], jeśli każda jej minimalna podsieć zamknięta jest automatem skończonym oraz istnieje zbiór automatów $\mathcal{N}_i$, $i \in K$, który pokrywa sieć $\mathcal{N}$. SP $\mathcal{N} = \langle P, T, F \rangle$ jest <u>SP swobodnego wyboru</u> [44], jeśli $(\forall p \in P)(\forall t \in T)$ $(\langle p,t \rangle \in F \Longrightarrow (p^\bullet = \{t\}$ lub $^\bullet t = \{p\}))$, tzn. łuk skierowany z miejsca p do przejścia t jest albo jedynym łukiem wychodzącym z miejsca p lub jedynym łukiem skierowanym do przejścia t.

W SP swobodnego wyboru dopuszczalne są struktury o postaci przedstawionej na rys. 2.6a, natomiast nie są dopuszczalne fragmenty ilus-

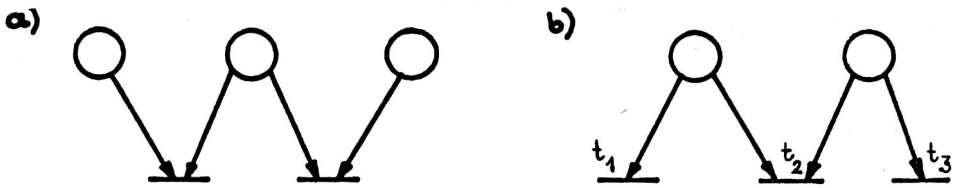
trowane rys. 2.6b.



Rys. 2.6

SP* swobodnego wyboru są przydatne w opisie architektury komputerów jednoprocesorowych [127], komputerów sterowanych przepływem danych [58], procesów sekwencyjnych.

SP jest <u>prostą SP</u> [44], jeśli każde przejście ma najwyżej jedno dzielone miejsce wejściowe. W prostej SP mogą wystąpić fragmenty reprezentowane rys. 2.7a, lecz nie mogą pojawić się struktury przedstawione na rys. 2.7b, ponieważ przejście t_2 ma dwa dzielone miejsca wejściowe.



Rys. 2.7

Miejsce p oraz przejście t tworzą pętlę, jeśli $p \in {}^\bullet t$ oraz $p \in t^\bullet$. Przykład pętli ilustruje rys. 2.8.

Rys. 2.8

<u>SP</u> jest <u>bezkonfliktowa</u> [66], jeśli każde miejsce, które jest miejscem wejściowym więcej niż jednego przejścia, tworzy z tym przejściem pętlę. W bezkonfliktowej SP są dopuszczalne konflikty w przód i w tył.

<u>Bezkonfliktowa SP bez pętli</u> [31] spełnia warunki:

1. Dla każdego miejsca p prawdziwa jest równość $|p^\bullet| = 1$,
2. W SP nie istnieją pętle.

Jeśli SP $\mathcal{N}$ ma pewną własność strukturalną (np. $\mathcal{N}$ jest SP swobodnego wyboru), to tę samą własność przypisujemy ZSP $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$ (np.

$\mathcal{N}$ jest ZSP swobodnego wyboru).

Powyższe klasy sieci są wyznaczane przez podane własności strukturalne - niezależnie od znakowania początkowego. Istnieją klasy ZSP*, których zdefiniowanie wymaga odwołania się do znakowania początkowego. Taką klasą jest klasa sieci trwałych.

Niech symbol $M \xrightarrow{t}$ oznacza, że przejście t jest przygotowane do palenia dla znakowania M. ZSP jest trwałą [66], jeśli dla każdych dwu przejść t_1, t_2, $t_1 \neq t_2$ oraz każdego znakowania M osiągalnego ze znakowania początkowego, jeśli $M \xrightarrow{t1}$ i $M \xrightarrow{t_2}$, to $M \xrightarrow{t_1 t_2}$, tzn. jeśli przejścia t_1 oraz t_2 mogą być palone dla znakowania M, to palenie jednego z tych przejść nie powoduje niemożności palenia drugiego z przejść.

2.2. Zależności między klasami sieci

W pierwszej kolejności przebadamy zależności między sieciami bez konfliktów w przód oraz sieciami bez konfliktów w tył a grafami synchronizacji (graf synchronizacji wraz ze znakowaniem tworzy graf znakowany).

Twierdzenie 2.4 [79]

Jeśli dla silnie spójnej sieci bez konfliktów w przód $\mathcal{N}$ istnieje znakowanie M_0 takie, że ZSP $\mathcal{N} = <\mathcal{N}, M_0>$ jest żywa i ograniczona, to SP $\mathcal{N}$ jest grafem synchronizacji.

Dowód

Ponieważ sieć $\mathcal{N}$ jest żywa, zatem musi istnieć sekwencja palenia $\sigma = t_a, \ldots, t_n$, która zawiera każde przejście co najmniej jeden raz, a ponadto nie powoduje zmniejszania liczby znaczników w żadnym z miejsc sieci. Niech $f(t_k)$ oznacza liczbę wystąpień przejścia t_k w sekwencji σ. Niech sieć zawiera fragment ilustrowany rys. 2.9.

Sekwencja σ spełnia warunek

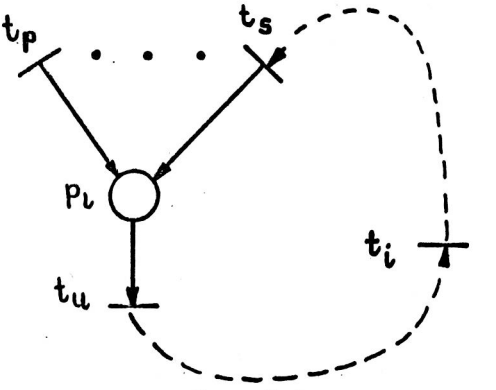
$$(\forall t_k \in T)(f(t_k) \geq 1).$$

Ponieważ sieć $\mathcal{N}$ jest ograniczona, więc dodatkowo spełniona musi być zależność

$$\sum_{t_r \in \cdot p_i} f(t_r) = f(t_u).$$

Z powyższych wymagań wynika relacja

$$f(t_s) < f(t_u).$$



Rys. 2.9

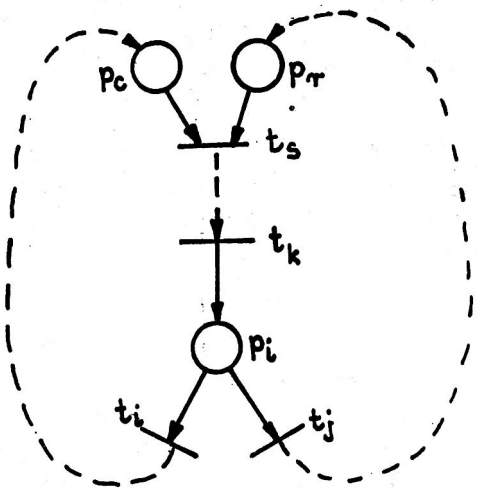
Sieć jest silnie spójna, a więc istnieje droga skierowana z przejścia t_u do przejścia t_s. Niech droga ta zawiera przejście t_i. Rozumując podobnie jak poprzednio stwierdzamy, że $f(t_s) \geq f(t_i)$, $f(t_i) \geq f(t_u)$, a więc $f(t_s) \geq f(t_u)$. Otrzymaliśmy zatem sprzeczność. Stąd spełniony musi być warunek $f(t_s) = f(t_u)$, a więc do miejsca p_i może być skierowany dokładnie jeden łuk, czyli sieć $\mathcal{N}$ musi być grafem synchronizacji. ∎

Twierdzenie 2.5 [79]

Jeśli dla silnie spójnej sieci bez konfliktów w tył $\mathcal{N}$ istnieje znakowanie M_0 takie, że ZSP $\mathcal{M} = <\mathcal{N}, M_0>$ jest żywa i ograniczona, to SP $\mathcal{N}$ jest grafem synchronizacji.

Dowód

Rozważmy fragment sieci $\mathcal{N}$ przedstawiony na rys. 2.10.



Rys. 2.10

Sieć jest silnie spójna, a więc musi istnieć droga skierowana z przejścia t_i do przejścia t_k oraz droga z przejścia t_j do przejścia t_k. Ponieważ sieć jest pozbawiona konfliktów wstecz, a więc pierwszym wspólnym wierzchołkiem obu dróg nie może być miejsce. Niech pierwszym wspólnym wierzchołkiem obu dróg będzie przejście t_s. Dla jednokrotnego palenia przejścia t_s wymagany jest co najmniej jeden znacznik w każdym z miejsc p_c,p_r. Aby miejsca te mogły zawierać co najmniej po jednym znaczniku, każde z przejść t_i, t_j musi palić się co najmniej jednokrotnie. W tym celu z miejsca p_i muszą być pobrane co najmniej dwa znaczniki. Każdorazowe palenie przejścia t_k dostarcza tylko jeden znacznik do

miejsca p_i, a więc sieć nie jest żywa. Aby sieć $\mathcal{N}$ była żywa - z miejsca p_i musi być skierowany dokładnie jeden łuk, czyli sieć $\mathcal{S}$ musi być grafem sychronizacji. ∎

Obecnie przeanalizujemy relacje między sieciami bez współbieżności w przód oraz sieciami bez współbieżności w tył a automatami.

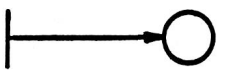
Twierdzenie 2.6 [79]

Jeśli dla silnie spójnej sieci bez współbieżności w przód $\mathcal{S}$ istnieje znakowanie M_0 takie, że ZSP $\mathcal{N} = <\mathcal{S}, M_0>$ jest żywa i ograniczona, to SP $\mathcal{S}$ jest automatem.

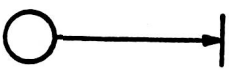
Dowód

Niech sieć $\mathcal{S}$ zawiera przejście t_r takie, że $|{}^\bullet t_r| > 1$. Ponieważ $|t_r^\bullet| = 1$, a więc każde palenie tego przejścia zmniejsza sumaryczną liczbę znaczników w sieci $\mathcal{N}$.

Ze względu na silną spójność sieć nie może zawierać przejścia generującego znaczniki (pozbawionego miejsc wejściowych) ilustrowanego rys. 2.11.



Rys. 2.11



Rys. 2.12

Aby sieć $\mathcal{N}$ była żywa, musi istnieć sekwencja palenia zawierająca nieograniczoną liczbę razy przejście t_r. Sekwencja taka nie istnieje, ponieważ każde palenie przejścia t_r zmniejsza sumaryczną liczbę znaczników w sieci, a żadne z pozostałych przejść nie zwiększa tej liczby. Stąd każde przejście musi zawierać jedno miejsce wejściowe, a więc sieć $\mathcal{S}$ musi być automatem. ∎

Twierdzenie 2.7 [79]

Jeśli dla silnie spójnej sieci bez współbieżności w tył $\mathcal{S}$ istnieje znakowanie M_0 takie, że ZSP $\mathcal{N} = <\mathcal{S}, M_0>$ jest żywa i ograniczona, to SP $\mathcal{S}$ jest automatem.

Dowód

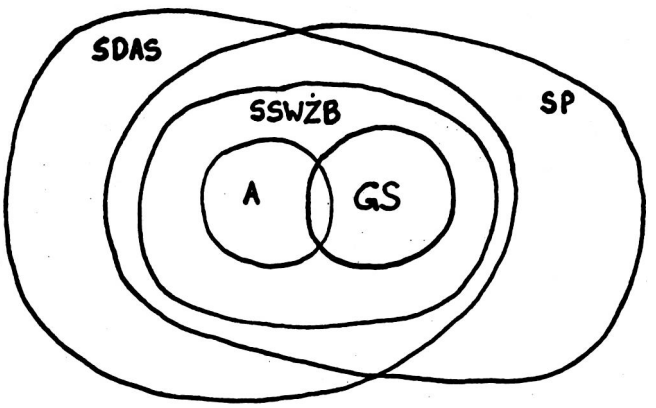
Niech sieć zawiera przejście t_r takie, że $|t_r^\bullet| > 1$. Palenie tego przejścia zwiększa liczbę znaczników w sieci $\mathcal{N}$. Ze względu na silną spójność, sieć nie może zawierać przejścia pochłaniającego znaczniki (pozbawionego miejsc wyjściowych) ilustrowanego rys. 2.12.

Rozumując analogicznie do dowodu poprzedniego twierdzenia stwierdzamy, że sieć $\mathcal{N}$ jest automatem. ∎

Ze względów praktycznych badamy żywe i ograniczone ZSP^*. Zatem z twierdzeń 2.4, 2.5, 2.6, 2.7 wynika, że analiza sieci bez konfliktów w przód, bez konfliktów w tył, bez współbieżności w przód, bez współbieżności w tył jest zbędna, jeśli przebadamy grafy synchronizacji i automaty.

Jeśli dla SP $\mathcal{N}$ istnieje takie znakowanie M, że ZSP $\mathcal{M} = <\mathcal{N}, M>$ jest żywa i bezpieczna (ograniczona), to mówimy, że SP $\mathcal{N}$ ma żywe i bezpieczne (ograniczone) znakowanie. Klasa SP^* mających żywe i bezpieczne znakowanie jest zawarta w klasie SP^* mających żywe i ograniczone znakowanie.

Relacje między niektórymi klasami SP^* wyraża rys. 2.13.



Rys. 2.13

Interpretacja skrótów z rys. 2.13 jest następująca:

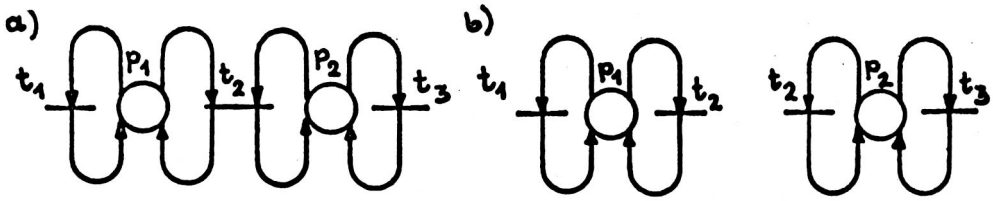
A - klasa automatów,
GS - klasa grafów synchronizacji,
SSWŻB - klasa SP^* swobodnego wyboru mających żywe i bezpieczne znakowanie,
SDAS - klasa SP^* dekomponowalnych na automaty,
SP - klasa prostych SP^*.

Większość relacji z tego rysunku wynika z prac [44], [102].

Uzasadnimy zależność, która w tych pracach nie jest zawarta. Klasa prostych SP^* nie jest zawarta w klasie SP^* dekomponowalnych na automaty. Sieć ilustrowana rys. 2.5a, jest prostą SP. Sieć ta nie jest

SP dekomponowalną na automaty, ponieważ minimalna podsieć zamknięta z rys. 2.5b nie jest automatem.

Klasa SP^{*} dekomponowalnych na automaty nie jest zawarta w klasie prostych SP^{*}. Sieć z rys. 2.14a jest SP dekomponowalną na automaty, dwie bowiem jej minimalne podsieci zamknięte, reprezentowane rys. 2.14b, są automatami. Sieć ta nie jest prostą SP, przejście t_2 bowiem ma dwa dzielone miejsca wejściowe p_1, p_2.



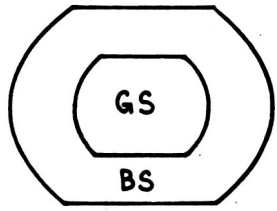
Rys. 2.14

Relację między klasą grafów synchronizacji (GS) a klasą bezkonfliktowych SP^{*} (BS) obrazuje rys. 2.15.

Powyższa relacja wynika bezpośrednio z definicji obu klas.

Klasa grafów synchronizacji nie jest zawarta w klasie bezkonfliktowych SP^{*} bez pętli, ponieważ w grafach synchronizacji dopuszczalne są pętle.

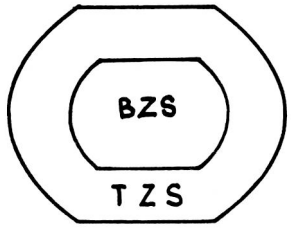
Klasa bezkonfliktowych SP^{*} bez pętli jest zawarta w klasie SP^{*} bez konfliktów w przód (dopuszczających pętle).



Rys. 2.15

Relację, wynikającą w prosty sposób z definicji, między klasą bezkonfliktowych ZSP^{*} (BZS) a trwałymi ZSP^{*} (TZS) ilustruje rys. 2.16.

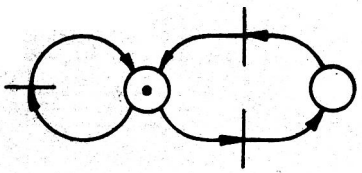
Klasa żywych bezpiecznych bezkonfliktowych ZSP^{*} nie zawiera klasy żywych bezpiecznych ZSP^{*} $\mathcal{M} = < \mathcal{N}, M_0 >$, których SP $\mathcal{N}$ jest automatem. ZSP z rys. 2.17 jest żywa i bezpieczna, natomiast jej SP jest automatem, ale nie jest bezkonfliktowa.



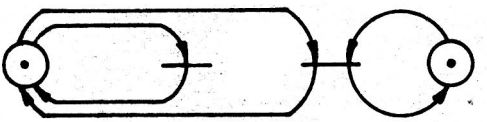
Rys. 2.16

Jednak klasa żywych bezpiecznych bezkonfliktowych ZSP^{*} nie jest zawarta w klasie ży-

wych bezpiecznych ZSP* swobodnego wyboru. ZSP z rys. 2.18 jest żywa bezpieczna bezkonfliktowa, ale nie jest siecią swobodnego wyboru.



Rys. 2.17



Rys. 2.18

2.3. Rozstrzygalność problemu czasu cyklu

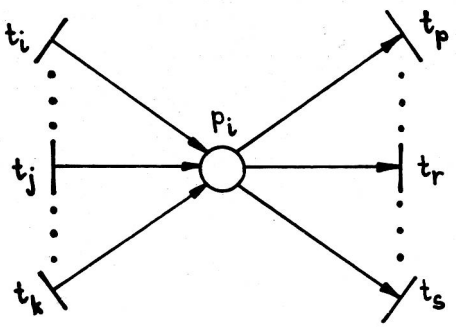
Twierdzenie 2.8

DPCC i OPCC dla DSP* są rozstrzygalne.

Dowód

Pierwsza - zasadnicza część dowodu jest identyczna dla DPCC i OPCC. Stąd stosować będziemy pojęcie PCC.

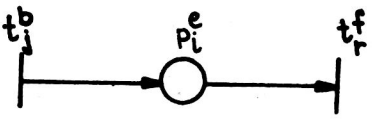
Dla ograniczonej ZSP - zbiór znakowań osiągalnych ze znakowania początkowego M_0 jest skończony. Dla każdego znakowania M osiągalnego z M_0 przeprowadzamy następujące rozważania. Badamy wszystkie takie sekwencje palenia $\sigma = t_{i_1},\ldots,t_{i_s}$, których wektor palenia równy jest T--niezmiennikowi I, a które mogą być palone dla znakowania M. Dla znakowania M i sekwencji σ spełniającej powyższe wymagania konstruujemy graf znakowany zgodnie z następującym algorytmem (graf znakowany jest ZSP spełniającą warunek $|{}^{\bullet}p| = |p^{\bullet}| = 1$ dla każdego miejsca p). Przeanalizujmy fragment ZSP ilustrowany rys. 2.19.



Rys. 2.19

Dla miejsca p_i oraz T-niezmiennika I jest spełniony warunek $\sum_{t_j \in {}^{\bullet}p_i} W(<t_j,p_i>)\, I(t_j) = \sum_{t_r \in p_i^{\bullet}} W(<p_i,t_r>)\, I(t_r)$. Wartość każdej z tych dwu sum oznaczmy symbolem K_i. Stąd podczas palenia sekwencji σ, której wektor palenia równy jest T-niezmiennikowi I, przez miejsce p_i przechodzi K_i znaczników. Niech dany znacznik będzie dodawany do miejsca p_i w wyniku palenia przejścia t_j oraz po-

bierany na rzecz palenia przejścia t_r. W tym przypadku do grafu znakowanego włączamy fragment z rys. 2.20. Stąd w uzyskanym grafie znakowanym przejście t_j występuje $I(t_j)$ razy, natomiast miejsce p_i - K_i razy tzn. w symbolu t_j^b indeks $b \in \{1,\ldots,I(t_j)\}$ oraz w symbolu p_i^e indeks $e \in \{1,\ldots,K_i\}$. Indeks b równy jest liczbie wskazującej po raz który przejście t_j wystąpiło w sekwencji σ.



Rys. 2.20

Na podstawie sekwencji σ możemy określić na rzecz palenia jakich przejść t_u są wykorzystywane znaczniki zadane znakowaniem początkowym. Jeśli pewien znacznik znakowania początkowego z miejsca p_r służy paleniu przejścia t_s po raz pierwszy w sekwencji σ, to w grafie znakowanym lokujemy znacznik w tym miejscu p_r^j, z którego jest skierowany łuk do przejścia t_s^1.

Otrzymany graf znakowany jest bezpieczny i żywy. Dla DSP, której ZSP jest grafem znakowanym istnieją algorytmy wielomianowe dla DPCC [101] i OPCC [73].

Zbiór sekwencji palenia σ, których wektor palenia równy jest T-niezmiennikowi, a które mogą być palone dla znakowania M jest skończony. Ponadto dla ograniczonej ZSP - zbiór znakowań M osiągalnych ze znakowania początkowego M_0 jest skończony. Zatem DPCC i OPCC są rozstrzygalne. ■

2.4. Złożoność obliczeniowa problemu czasu cyklu

Relacja zawierania się jednej klasy sieci w drugiej jest relacją częściowego porządku. Mówimy, że klasa X_1 jest mniejsza od klasy X_2, jeśli $X_1 \subset X_2$. Zależy nam na znalezieniu granicy między klasami deterministycznych sieci Petriego (DSP^*), dla których istnieje algorytm wielomianowy dla problemu czasu cyklu (PCC) a klasami DSP^*, dla których PCC jest NP-trudny. Ponadto, jeśli istnieje algorytm wielomianowy dla pewnej klasy, to istnieje algorytm wielomianowy dla każdej jej podklasy. Jeśli PCC dla pewnej klasy jest NP-trudny, to jest NP-trudny dla każdej jej nadklasy. W przypadku udowodnienia NP-trudności PCC dla pewnej klasy sieci - znalezienie algorytmu wielomianowego dla tej klasy jest mało prawdopodobne. Stąd w tym przypadku - w celu znalezienia dokładnego rozwiązania PCC możemy stosować takie metody, jak metoda podziału i ograniczeń, programowanie dynamiczne.

2.4.1. Sieci P-niezmiennicze

W pierwszej kolejności rozważać będziemy DSP^*, których SP jest grafem synchronizacji (ZSP jest grafem znakowanym).

Dla czystych grafów synchronizacji (bez pętli ilustrowanych rys. 2.8), T-niezmienniki wyrażające zachowanie cykliczne otrzymujemy poprzez rozwiązanie równania macierzowego $C\,I = 0$.

Dla czystych grafów synchronizacji przestrzeń rozwiązań równania $C\,I = 0$ jest wymiaru 1, natomiast wektor $I^T = <11...1>$ jest bazą tej przestrzeni. Wektor ten jest również T-niezmiennikiem dla grafów synchronizacji, w których mogą istnieć pętle.

Minimalny czas cyklu DSP, której SP jest grafem synchronizacji - dla T-niezmiennika I_1 takiego, że $I_1(t_i) = s$ dla każdego przejścia $t_i \in T$ (s - liczba naturalna), jest s-krotnie większy niż minimalny czas cyklu dla T-niezmiennika I_2 spełniającego równość $I_2(t_i) = 1$ dla każdego przejścia $t_i \in T$ [102]. Stąd analizować będziemy tylko zachowanie cykliczne DSP dla drugiego z wymienionych T-niezmienników.

DPCC dla DSP, której SP jest grafem synchronizacji może być rozwiązany za pomocą algorytmu zawartego w pracy [101]. Złożoność tego algorytmu równa jest $O(m^3)$, gdzie $m = |P|$.

OPCC może być rozwiązany na podstawie twierdzenia 2.9.

Twierdzenie 2.9 [101]

Minimalny czas cyklu DSP, której SP jest grafem synchronizacji, dla T-niezmiennika $I^T = <11...1>$ równy jest

$$\gamma(I,x^*) = \max_{C_j \in C^*} \frac{T_j}{K_j} \; (= \gamma_{min}),$$

przy czym chwile rozpoczęcia kolejnych paleń poszczególnych przejść zadane są równościami

$$t_i(k) = \tau_i + (k-1)\,\gamma_{min},$$

gdzie: C^* - zbiór wszystkich obwodów SP,

$T_j = \sum_{t_i \in T(C_j)} \tau(t_i)$ - suma czasów palenia przejść obwodu C_j,

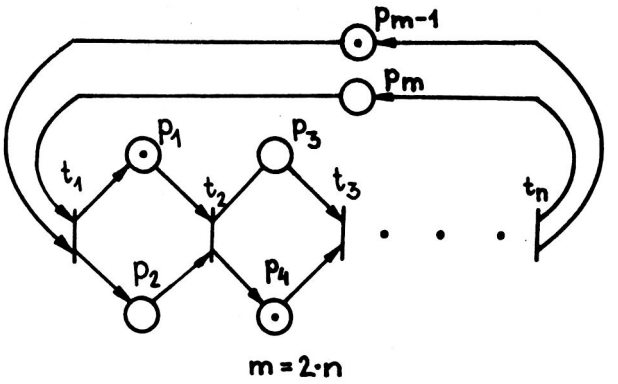
$K_j = \sum_{p_i \in P(C_j)} M_0(p_i)$ - liczba znaczników w miejscach obwodu C_j,

$t_i(k)$ - chwila rozpoczęcia k-tego palenia przejścia t_i,

τ_i - chwila rozpoczęcia pierwszego palenia przejścia t_i.

Dowód tego twierdzenia zawiera metodę wyznaczania liczb τ_i.

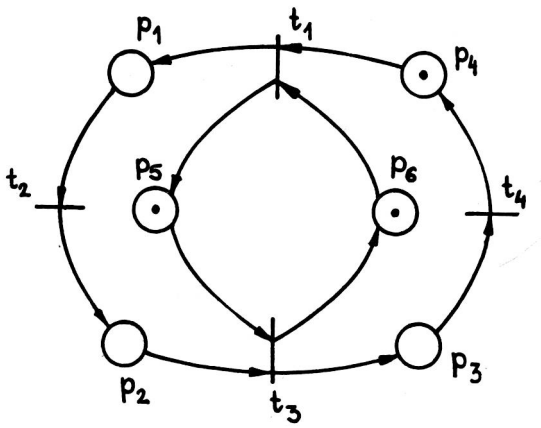
Istnieją grafy synchronizacji o wykładniczej liczbie obwodów. Graf synchronizacji z rys. 2.21 zawiera 2^n obwodów. Zatem metoda oparta na tw. 2.9 wymaga w tym przypadku przebadania wykładniczej liczby obwodów.



Rys. 2.21

W pracy [111], w zakresie badania minimalnego czasu cyklu sieci, której SP jest grafem synchronizacji z czasem przypisanym miejscom są rozważane jedynie obwody z bazy obwodów.Baza obwodów grafu synchronizacji zawiera $|P|-|T|+1$ elementów. W wyniku transformacji(opisanej w pracy [113]) DSP, której SP jest grafem synchronizacji w sieć, której SP jest grafem synchronizacji z czasem przypisanym miejscom - liczba obwodów bazy pozostaje taka sama. Zatem metoda oparta na powyższej transformacji jest złożoności wielomianowej. Jednak w ten sposób można uzyskać jedynie dolne oszacowanie minimalnego czasu cyklu, co zostanie zilustrowane przykładem.

Rozważmy DSP, której SP jest grafem synchronizacji, przedstawioną na rys. 2.22.

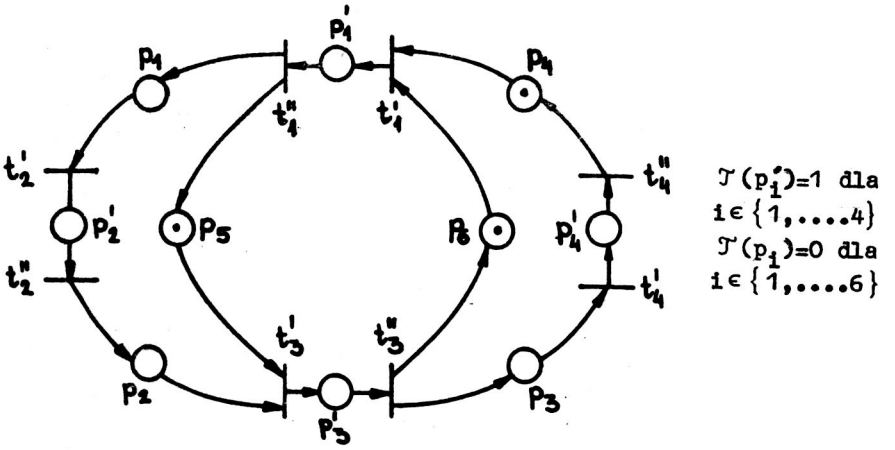


$\mathcal{T}(t_i) = 1$ dla $i \in \{1,.....4\}$

Rys. 2.22

W wyniku przetransformowania DSP z rys. 2.22 w sieć z czasem przypisanym miejscom - uzyskujemy sieć ilustrowaną rys. 2.23.

Wielkość $\mathcal{T}(p)$ jest minimalnym czasem przez jaki znacznik musi znajdować się w miejscu p, aby mógł być wykorzystany na rzecz palenia przejścia $t \in p^{\bullet}$. Palenie przejścia jest zdarzeniem o zerowym czasie trwania.



Rys. 2.23

Dla sieci z rys. 2.23 - przykładowa baza obwodów C^b zawiera następujące obwody:

$C_1 = p_5 t_3' p_3' t_3'' p_6 t_1' p_1' t_1'' p_5$

$C_2 = p_5 t_3' p_3' t_3'' p_3 t_4' p_4' t_4'' p_4 t_1' p_1' t_1'' p_5$

$C_3 = p_1 t_2' p_2' t_2'' p_2 t_3' p_3' t_3'' p_6 t_1' p_1' t_1'' p_1$.

Oszacowanie $\underline{\gamma}$ minimalnego czasu cyklu uzyskane na podstawie bazy obwodów C^b jest określone równością

$$\underline{\gamma} = \max_{C_i \in C^b} \frac{T_i'}{K_i'},$$

gdzie: $T_i' = \sum_{p_j \in P(C_i)} \mathcal{T}(p_j)$ - suma czasów dla miejsc obwodu C_i,

$K_i' = \sum_{p_j \in P(C_i)} M_0(p_j)$ - liczba znaczników w miejscach obwodu C_i.

Zatem w rozpatrywanym przykładzie otrzymujemy

$$\underline{\gamma} = \max \left\{ \frac{2}{2}, \frac{3}{2}, \frac{3}{1} \right\} = 3.$$

Jednakże dla obwodu

$$C_4 = p_1 t_2' p_2' t_2'' p_2 t_3' p_3' t_3'' p_3 t_4' p_4' t_4'' p_4 t_1' p_1' t_1'' p_1$$

mamy $T_4/K_4 = \frac{4}{1} = 4$. Stąd γ jest jedynie dolnym ograniczeniem dla $\gamma(<1\ldots1>, x^*) = 4$.

OPCC dla DSP, której SP jest grafem synchronizacji, rozwiązywać będziemy na podstawie programowania liniowego.

PROBLEM PROGRAMOWANIA LINIOWEGO:

$$\min \sum_{j=1}^{p} c_j x_j$$

$$b_i \leqslant \sum_{j=1}^{p} a_{ij} x_j, \quad i = 1,\ldots,r,$$

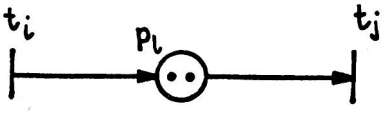
gdzie $p,r \in N$, a_{ij}, b_i, c_j są liczbami wymiernymi.

Dla DSP $\mathcal{M}^t = <\mathcal{M}, \mathcal{T}>$, której SP $\mathcal{M}$ jest grafem synchronizacji, czas cyklu γ spełnia zależność [101], [102]

$$t_i(k) = \tau_i + (k-1)\,\gamma,$$

gdzie: $t_i(k)$ - chwila rozpoczęcia k-tego palenia przejścia t_i,
τ_i - chwila rozpoczęcia pierwszego palenia przejścia t_i.

Obecnie przedstawimy algorytm wielomianowy [73] dla wyznaczania minimalnego czasu cyklu badanej DSP. Rozważmy fragment sieci ilustrowany rys. 2.24.



Rys. 2.24

Dla czasu cyklu γ spełnione muszą być relacje:
chwila zakończenia k-tego palenia przejścia $t_i \leqslant$ chwila rozpoczęcia $(k+M_0(p_l))$-tego palenia przejścia t_j,

$$t_i(k) + \mathcal{T}(t_i) \leqslant t_j(k + M_0(p_l)),$$

$$\tau_i + (k-1)\,\gamma + \mathcal{T}(t_i) \leqslant \tau_j + (k-1+M_0(p_l))\,\gamma,$$

$$\mathcal{T}(t_i) \leqslant \tau_j - \tau_i + M_0(p_l)\,\gamma.$$

W celu wyznaczenia minimalnego czasu cyklu, formułujemy problem programowania liniowego:

$$\min \gamma$$

$$\mathcal{T}(t_i) \leqslant \tau_j - \tau_i + M_0(p_l)\,\gamma \quad \text{dla każdego } p_l \in P,$$

$$0 \leqslant \gamma,$$

gdzie $\tau_1,\ldots,\tau_n$ $(n = |T|)$, γ są zmiennymi.

Algorytm wielomianowy dla zagadnienia programowania liniowego, efektywniejszy niż algorytm Khachiana, zawarty jest w [57]. Złożoność tego algorytmu wynosi $O(k^{3,5}L)$, gdzie k jest liczbą zmiennych, natomiast L - liczbą zależną od długości danych wejściowych. W przypadku rozpatrywanego przez nas zagadnienia programowania liniowego liczba zmiennych wynosi n+1.

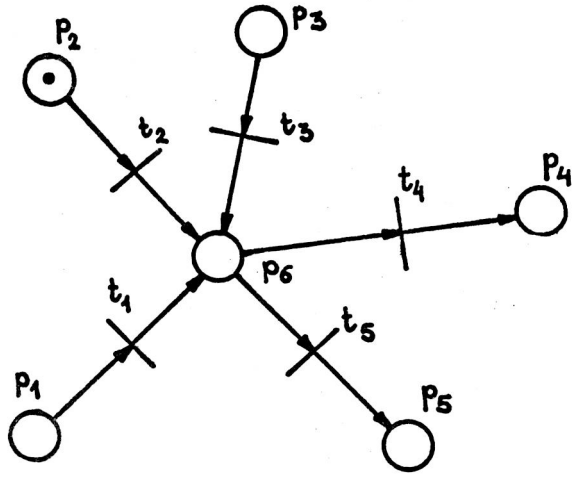
Twierdzenie 2.10 [73]

Dla DSP, której SP jest grafem synchronizacji - dla OPCC istnieje algorytm o złożoności $O(n^{3,5}L)$, gdzie $n = |T|$, L - liczba zależna od długości danych wejściowych. ∎

Obecnie przeanalizujemy OPCC dla DSP, której SP jest automatem. Zakładamy, że $I(t_i) > 0$ dla każdego przejścia $t_i \in T$. Jeśli dla pewnych przejść prawdziwa byłaby relacja $I(t_i) = 0$, to przejścia te wraz z incydentnymi z nimi łukami moglibyśmy wyeliminować. Następnie, każdą silnie spójną podsieć, uzyskaną w wyniku powyższej eliminacji, można badać odrębnie zgodnie z algorytmem dla przypadku: $I(t_i) > 0$ dla wszystkich przejść.

Efektywny algorytm dla OPCC dla DSP, której SP jest automatem - przy założeniu $I(t_i) > 0$ dla każdego przejścia zawarty jest w pracy [102].

OPCC dla rozważanego przypadku można rozwiązać na podstawie zależności z pracy [102]. Dla uzasadnienia tej zależności rozważmy fragment sieci przedstawiony na rys. 2.25.



Rys. 2.25

Niech palenie przejścia t_2 rozpoczyna się w chwili τ'. W chwili $\tau'' = \tau' + \mathcal{T}(t_2)$ znacznik jest dodawany do miejsca p_6. Znacznik ten może być natychmiast wykorzystany do palenia jednego z przejść t_4 lub t_5. Dla danego T-niezmiennika I istnieje taka czasowa sekwencja palenia, że każdy znacznik w każdej chwili może być wykorzystany do palenia jednego z przejść. Stąd minimalny czas cyklu zadany jest wzorem

$$\gamma(I,x^*) = \frac{\sum_{t_i \in T} I(t_i)\,\mathcal{T}(t_i)}{\sum_{p_i \in P} M_0(p_i)} \quad [102].$$

Dla silnie spójnego automatu liczby $m = |P|$, $n = |T|$ spełniają zależność $m \leq n$. Zatem złożoność obliczeniowa algorytmu wyznaczania $\gamma(I,x^*)$ wynosi $O(n)$.

Twierdzenie 2.11

OPCC dla DSP, której SP jest automatem, przy założeniu: T - niezmiennik I równy jest $I(t_i) > 0$ dla każdego przejścia $t_i \in T$, może być rozwiązany za pomocą algorytmu o złożoności $O(n)$, gdzie $n = |T|$. ∎

Na podstawie powyższego twierdzenia dla problemu decyzyjnego formułujemy wniosek.

Wniosek 2.1

DPCC dla DSP, której SP jest automatem, przy założeniu: T - niezmiennik I równy jest $I(t_i) > 0$ dla każdego przejścia $t_i \in T$, może być rozwiązany za pomocą algorytmu o złożoności $O(n)$, gdzie $n = |T|$.

Twierdzenie 2.12

DPCC dla DSP, której ZSP jest bezpieczną siecią swobodnego wyboru jest silnie NP-trudny.

Dowód

Dokonamy pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU Π_2 do DPCC Π_1 dla DSP, której ZSP ilustrowana jest rys.2.26.

Badana ZSP jest bezpieczną siecią swobodnego wyboru.

Dla każdej instancji $I_2 \in D_{\Pi_2}$ - dane odpowiadającej jej instancji $I_1 \in D_{\Pi_1}$ są następujące:

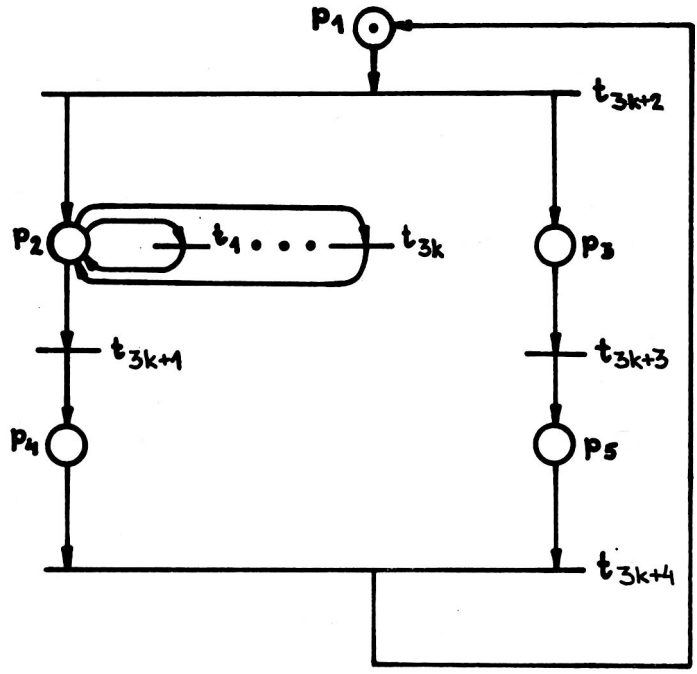
1. Funkcja czasu palenia:

$\mathcal{T}(t_i) = x_i$ dla $i \in \{1,\ldots,3k\}$,

$\mathcal{T}(t_{3k+3}) = B$,

$\mathcal{T}(t_{3k+1}) = \mathcal{T}(t_{3k+2}) = \mathcal{T}(t_{3k+4}) = 0$,

2. T-niezmiennik:
$I(t_i) = 1$ dla $i \in \{1,\ldots,3k\}$,
$I(t_j) = k$ dla pozostałych przejść,
3. Wartość funkcji kryterialnej:
$\gamma = k\,B$.



Rys. 2.26

Załóżmy, że odpowiedź dla instancji $I_2 \in D_{\Pi 2}$ brzmi "tak". Zatem istnieje podział zbioru X na k rozłącznych podzbiorów:

$S_1 = \{x_{11},x_{12},x_{13}\},\ldots,S_j = \{x_{j1},x_{j2},x_{j3}\},\ldots,S_k = \{x_{k1},x_{k2},x_{k3}\}$.

Na podstawie powyższego podziału dokonamy podziału zbioru przejść $T' = \{t_i : i \in \{1,\ldots,3k\}\}$ na k rozłącznych podzbiorów trójelementowych:

$T_1 = \{t_{11},t_{12},t_{13}\},\ldots,T_j = \{t_{j1},t_{j2},t_{j3}\},\ldots,T_k = \{t_{k1},t_{k2},t_{k3}\}$.

Suma czasów palenia przejść z podzbioru T_j, $j \in \{1,\ldots,k\}$ równa jest B.

Wybierzmy następującą czasową sekwencję palenia:

$<t_{j1},(j-1)\,B>$ dla $j \in \{1,\ldots,k\}$,
$<t_{j2},(j-1)\,B + \tau(t_{j1})>$ dla $j \in \{1,\ldots,k\}$,

$<t_{j3},(j-1)\ B + \mathcal{T}(t_{j1}) + \mathcal{T}(t_{j2})>$ dla $j \in \{1,\ldots,k\}$,
$<t_{3k+1},B\ldots j\ B\ldots k\ B>$,
$<t_{3k+2},0\ldots(j-1)\ B\ldots(k-1)\ B>$,
$<t_{3k+3},0\ldots(j-1)\ B\ldots(k-1)B>$,
$<t_{3k+4},B\ldots j\ B\ldots k\ B>$.

Czas wykonania powyższej czasowej sekwencji palenia równy jest k B. Powtarzając powyższą sekwencję (z uwzględnieniem przesunięć w czasie), otrzymujemy zachowanie cykliczne o czasie cyklu k B.

Obecnie przeprowadzimy dowód warunku koniecznego transformacji.

Załóżmy, że dla instancji $I_2 \in D_{\Pi_2}$ odpowiedź brzmi "nie". Stąd w podziale zbioru X na podzbiory trójelementowe istnieje taki podzbiór $S_j = \{x_{j1},x_{j2},x_{j3}\}$, że suma czasów palenia przejść z podzbioru $T_j = \{t_{j1},t_{j2},t_{j3}\}$ jest większa od B. Zatem długość przedziału od chwili rozpoczęcia palenia przejścia t_{3k+2}, bezpośrednio przed paleniem przejść ze zbioru T_j, do chwili zakończenia palenia przejścia t_{3k+4}, bezpośrednio po paleniu przejść ze zbioru T_j, jest większa od B. Ponieważ $\mathcal{T}(t_{3k+3}) = B$, a więc długość przedziału od chwili rozpoczęcia palenia przejścia t_{3k+2} po raz 1-ty do chwili zakończenia palenia przejścia t_{3k+4} po raz 1-ty jest nie mniejsza niż B. Zatem stwierdzamy, że nie istnieje taka czasowa sekwencja palenia x, dla której byłaby spełniona nierówność $\gamma(I,x) \leqslant k\ B$.

Stąd dla dowolnej instancji $I_2 \in D_{\Pi_2}$ odpowiedź brzmi "tak" wtedy i tylko wtedy, gdy dla odpowiadającej jej instancji $I_1 \in D_{\Pi_1}$ istnieje czasowa sekwencja palenia x o czasie cyklu $\gamma(I,x) \leqslant k\ B$.

Przeanalizujmy złożoność obliczeniową zdefiniowanej transformacji. Rozmiar instancji $I_2 \in D_{\Pi_2}$ spełnia zależność $N(I_2) = O(k \log B)$, natomiast maksymalna liczba w danych - $Max(I_2) = B$. Liczba kroków potrzebnych do zapisania danych instancji $I_1 \in D_{\Pi_1}$ jest $O(k \log B)$. Zatem badana transformacja jest transformacją wielomianową, a ponadto transformacją pseudowielomianową, czyli w konsekwencji również pseudowielomianową transformacją Turinga.

Ponieważ PROBLEM TRÓJPODZIAŁU jest silnie NP-zupełny, a więc DPCC dla analizowanej DSP jest silnie NP-trudny. ∎

Twierdzenie słabsze, zgodnie z którym DPCC dla DSP takiej, że ZSP jest bezpieczną siecią swobodnego wyboru, jest NP-trudny udowodniono w pracy [75].

Obecnie przebadamy złożoność obliczeniową PCC dla bezkonfliktowych i persystentnych ZSP*.

Twierdzenie 2.13

Jeśli dla silnie spójnej bezkonfliktowej SP $\mathcal{N}$ bez pętli istnieje takie znakowanie M_0, że ZSP $\mathcal{M} = <\mathcal{N},M_0>$ jest żywa i ograniczona,

to SP $\mathcal{N}$ jest grafem synchronizacji ($\mathcal{M}$ jest grafem znakowanym).

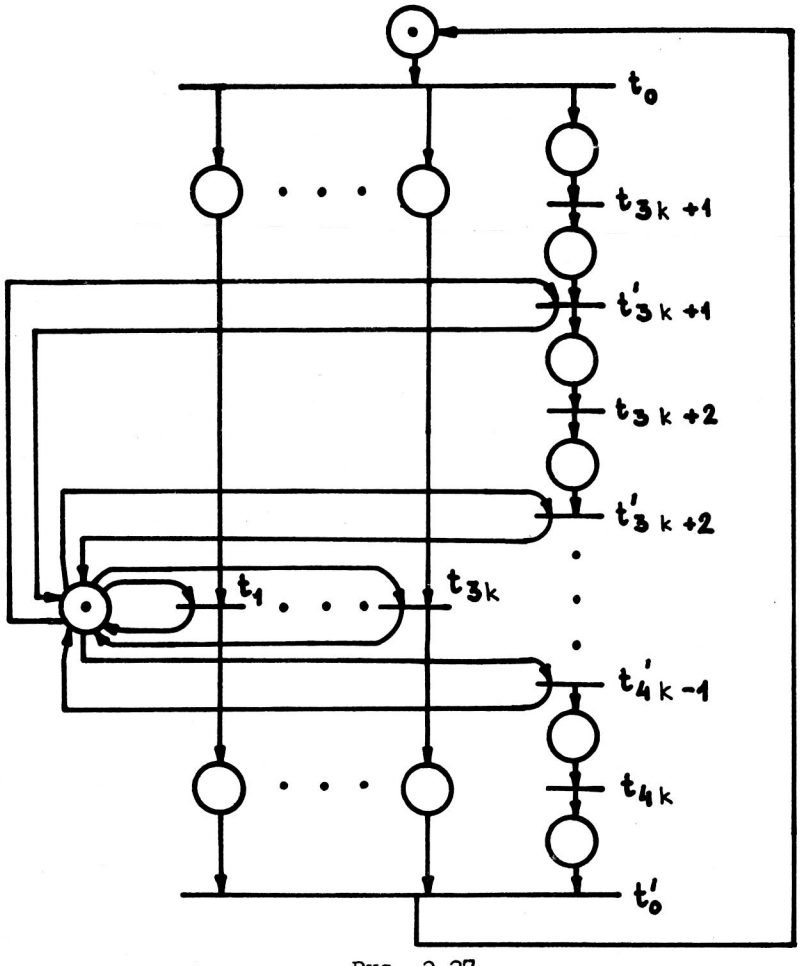
Dowód jest analogiczny do dowodu tw. 2.4. ■

Na podstawie powyższego tw. i tw. 2.10 formułujemy wniosek.

Wniosek 2.2

OPCC dla DSP, której SP jest bezkonfliktowa i bez pętli może być rozwiązany za pomocą algorytmu o złożoności $O(n^{3,5}L)$, gdzie $n = |T|$, L jest liczbą zależną od długości danych wejściowych.

Udowodnimy, że dla DSP*, których ZSP jest bezkonfliktowa i bezpieczna - DPCC jest silnie NP-trudny. W tym celu dokonamy pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla DSP, o bezkonfliktowej i bezpiecznej ZSP z rys. 2.27.



Rys. 2.27

Zakładamy, że czasy palenia spełniają zależności: $\mathcal{T}(t_i) = x_i$ dla $i \in \{1,\ldots,3k\}$, $\mathcal{T}(t_{3k+j}) = B$ dla $j \in \{1,\ldots,k\}$, $\mathcal{T}(t) = 0$ dla pozostałych przejść. Rozważamy czas cyklu dla **T-niezmiennika o wszystkich składowych równych jedności. Wartość funkcji kryterialnej: $\gamma = k\,B$. Dowód jest podobny do dowodu tw. 2.12.**

Twierdzenie 2.14 [78]

DPCC dla DSP, której ZSP jest bezkonfliktowa i bezpieczna, jest silnie NP-trudny. ∎

Ponieważ bezkonfliktowe ZSP^* są zawarte w klasie trwałych ZSP^*, a zatem otrzymujemy wniosek.

Wniosek 2.3

DPCC dla DSP, której ZSP jest trwała i bezpieczna jest silnie NP--trudny.

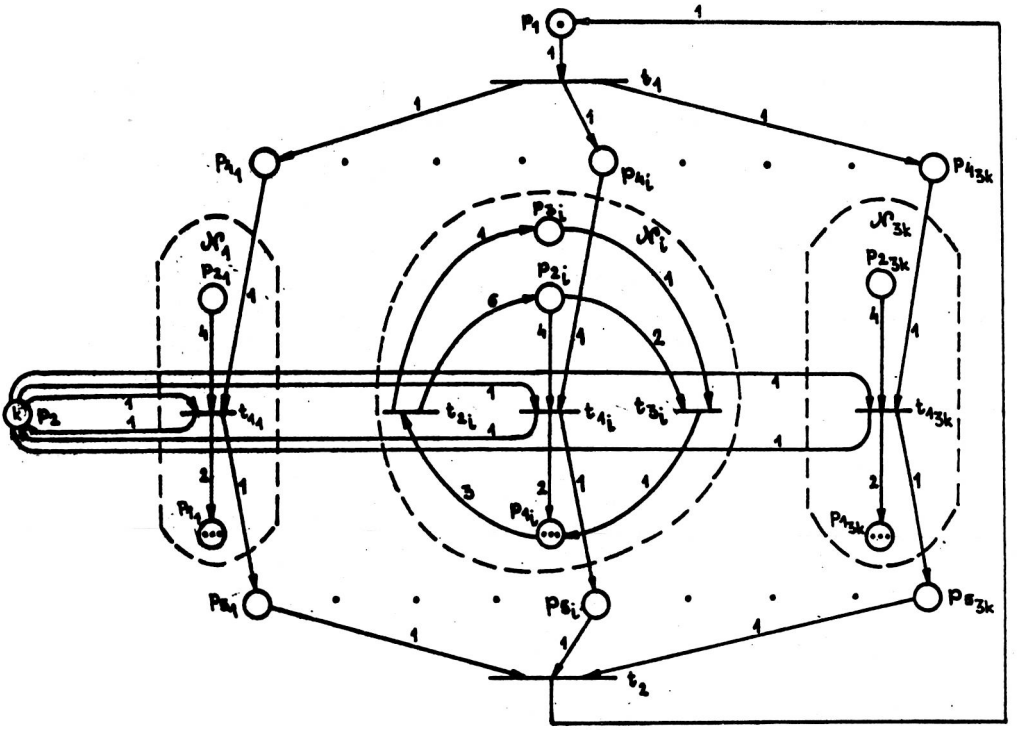
2.4.2. Sieci nie P-niezmiennicze

Dla pełni rozważań przeanalizujemy obecnie nie P-niezmiennicze SP^*.

Silne założenia o ograniczoności i żywotności ZSP $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$ nie gwarantują, że SP $\mathcal{N}$ jest P-niezmiennicza. W celu udowodnienia silnej NP-trudności DPCC dla DSP, której SP jest nie P-niezmiennicza - w pierwszej kolejności zbudujemy taką żywą i ograniczoną ZSP $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$, że SP $\mathcal{N}$ jest nie P-niezmiennicza.

Dla ZSP $\mathcal{M} = \langle \mathcal{N}, M_0 \rangle$ z rys. 2.28, ZSP^* $\mathcal{M}_j = \langle \mathcal{N}_j, M_{0_j} \rangle$ dla $j \in \{1,\ldots,i-1,i+1,\ldots,3k\}$ są izomorficzne z ZSP $\mathcal{M}_i = \langle \mathcal{N}_i, M_{0_i} \rangle$. Znakowanie początkowe zadane jest równościami: $M_{0_i}(p_{1_i}) = 3$, $M_{0_i}(p_{2_i}) = M_{0_i}(p_{3_i}) = 0$ dla $i \in \{1,\ldots,3k\}$, $M_0(p_1) = 1$, $M_0(p_2) = k$. Dla podsieci $\mathcal{M}_i$ wyodrębnionej z sieci $\mathcal{M}$, najkrótszymi sekwencjami palenia odtwarzającymi znakowanie M_{0_i} są $\sigma_{1_i} = t_{2_i} t_{1_i} t_{3_i}$, $\sigma_{2_i} = t_{2_i} t_{3_i} t_{1_i}$. Uwzględniając te sekwencje stwierdzamy, że ZSP $\mathcal{M}$ jest żywa i ograniczona. Stąd SP $\mathcal{N}$ jest strukturalnie żywa. Jednakże sieć ta nie jest strukturalnie ograniczona. Niech bowiem $M_{0_i}(p_{1_i}) = 6$, $M_{0_i}(p_{2_i}) = M_{0_i}(p_{3_i}) = 0$. W tym przypadku istnieje taka sekwencja palenia σ, której wektor palenia $f(\sigma)$ spełnia zależności: $f(t_{1_i}) = f(t_1) = f(t_2) = 3$, $f(t_{2_i}) = 2$, $f(t_{3_i}) = 0$. W wyniku tej sekwencji liczba znaczników w miejscu p_{3_i} jest zwiększana, przy zachowaniu liczby znaczników w pozostałych miejscach.

Obecnie udowodnimy, że SP $\mathcal{N}$ jest nie P-niezmiennicza. Dla każdego przejścia t_j P-niezmienniczej SP i dla pewnego wektora J o $|P|$ do-



Rys. 2.28

datnich składowych całkowitych spełniona musi być równość

$$\sum_{p_i \in \cdot t_j} J(p_i)\, W(<p_i, t_j>) = \sum_{p_i \in t_j \cdot} J(p_i)\, W(<t_j, p_i>).$$

Zatem aby SP $\mathcal{N}$ była P-niezmiennicza - prawdziwe muszą być równości:

dla przejścia t_1
$$J(p_1) = \sum_{i=1}^{3k} J(p_{4_i}), \tag{1}$$

dla przejścia t_2
$$\sum_{i=1}^{3k} J(p_{5_i}) = J(p_1), \tag{2}$$

dla przejścia t_{1_i}
$$J(p_2) + 4\, J(p_{2_i}) + J(p_{4_i}) = J(p_2) + 2\, J(p_{1_i}) + J(p_{5_i}), \tag{3}$$

dla przejścia t_{3_i}
$$2\, J(p_{2_i}) + J(p_{3_i}) = J(p_{1_i}). \tag{4}$$

W celu uproszczenia formuł, symbol $\sum_{i=1}^{3k}$ zastępujemy symbolem $\sum$. Z równości (1), (2) otrzymujemy $\sum J(p_{4_i}) = \sum J(p_{5_i})$ (5). Na podstawie wyrażenia (3) stwierdzamy, że $\sum 4\, J(p_{2_i}) + \sum J(p_{4_i}) = \sum 2\, J(p_{1_i}) + \sum J(p_{5_i})$ (6). Stąd uwzględniając formuły (5), (6) wnioskujemy, że $\sum 2\, J(p_{2_i}) = \sum J(p_{1_i})$ (7). Z równości (4) otrzymujemy $\sum 2\, J(p_{2_i}) + \sum J(p_{3_i}) = \sum J(p_{1_i})$ (8). Uwzględniając równości (7), (8) stwierdzamy, że $\sum J(p_{3_i}) = 0$. Zatem nie wszystkie składowe wektora J są dodatnie, czyli SP $\mathcal{N}$ jest nie P-niezmiennicza.

Pseudowielomianową transformację Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla rozpatrywanej sieci określają zależności:

1. $\mathcal{T}(t_{1_i}) = x_i$ dla $i \in \{1,\ldots,3k\}$, $\mathcal{T}(t_j) = 0$ dla pozostałych przejść,
2. składowe T-niezmiennika I równe są $I(t_i) = 1$ dla każdego przejścia,
3. wartość funkcji kryterialnej: $\gamma = k$ B.

Dowód, oparty na pomyśle: każdy z k znaczników z miejsca p_2 wykorzystywany jest na rzecz palenia trzech przejść ze zbioru $T' = \{t_{1_i} : i \in \{1,\ldots,3k\}\}$, jest podobny do dowodu tw. 2.12.

Twierdzenie 2.15

DPCC dla DSP, których SP jest siecią nie P-niezmienniczą jest silnie NP-trudny. ∎

Twierdzenie słabsze - zgodnie z którym DPCC dla powyższej klasy sieci jest NP-trudny zawarte jest w pracy [75].

2.4.3. NP-zupełność problemu czasu cyklu

Obecnie sformułujemy założenia, które wystarczają, aby DPCC dla DSP*, których SP jest silnie spójna, a ZSP - żywa i ograniczona, był silnie NP-zupełny.

Wymagamy, żeby ZSP powracała do znakowania początkowego po każdym cyklu pracy. Wymóg ten spełniają sieci z rys. 2.26, 2.27, 2.28 wykorzystywane w dowodach silnej NP-trudności DPCC. Założenie to jest często spełnione w praktyce, ponieważ wiele systemów cyklicznych powraca do stanu początkowego po cyklu pracy.

Ponadto zakładamy, że $\sum_{t_i \in T} I(t_i) \leq O(p(N(I_1)))$, czyli sumaryczna liczba paleń wszystkich przejść w czasie cyklu zadana T-niezmiennikiem I jest ograniczona od góry przez wielomian p od rozmiaru $N(I_1)$ instancji I_1 DPCC. Jeśli ograniczenie to nie jest spełnione, to istnieje takie przejście t_j, dla którego $I(t_j)$ jest wykładniczą funkcją rozmiaru $N(I_1)$. Przypadek taki nie ma istotnego znaczenia praktycznego.

Przy powyższych założeniach NDMT generuje wszystkie takie sekwencje składające się z elementów zbioru przejść, w których liczby wystąpień poszczególnych przejść są równe składowym T-niezmiennika I (sekwencje o długości ograniczonej od góry przez wielomian od rozmiaru $N(I_1)$). Dla wygenerowanych sekwencji - NDMT sprawdza czy są to sekwencje palenia, wyznacza czas wykonania czasowych sekwencji palenia przyporządkowanych tym sekwencjom palenia i porównuje ten czas z wartością funkcji kryterialnej. Liczba kroków etapu generacji i etapu sprawdzania zakończonego odpowiedzią "tak" jest $O(p_1(N(I_1)))$ dla pewnego wielomianu p_1. Zatem DPCC spełniający powyższe wymagania należy do klasy NP.

Podobnie jak w dowodzie tw. 2.12 możemy dokonać pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla DSP badanej w tym twierdzeniu.

Na podstawie powyższej analizy formułujemy następujące twierdzenie.

Twierdzenie 2.16

DPCC dla DSP $\mathcal{M}^t$, przy założeniach:

1. DSP $\mathcal{M}^t$ powraca do znakowania początkowego po czasie cyklu,
2. $\sum_{t_i \in T} I(t_i) \leqslant O(p(N(I_1)))$ gdzie p jest pewnym wielomianem od rozmiaru $N(I_1)$ instancji I_1 DPCC, jest silnie NP-zupełny. ∎

2.5. Oszacowanie minimalnego czasu cyklu

Ze względu na znaczną złożoność obliczeniową PCC uzasadnione jest szukanie oszacowań minimalnego czasu cyklu, które z jednej strony są możliwie dobrymi oszacowaniami, z drugiej zaś nie wymagają dużych nakładów obliczeniowych.

Ponieważ wykorzystywać będziemy macierz incydencji, a więc rozważać będziemy czyste SP^*. Uogólnienie uzyskanych rezultatów na SP^* z pętlami jest bardzo proste i nie będzie rozpatrywane.

W pierwszej kolejności rozważać będziemy P-niezmiennicze SP^*, a następnie sieci nie P-niezmiennicze.

2.5.1. Sieci P-niezmiennicze

Oszacowania minimalnego czasu cyklu dla P-niezmienniczych SP^* przedstawione są w pracy [73]. Najpierw zaprezentujemy dolne oszacowanie czasu cyklu zawarte w pracy [101], potem omówimy dolne oszacowanie z pracy [73], a następnie oba oszacowania porównamy.

Dolne oszacowanie minimalnego czasu cyklu dla P-niezmienniczych SP^{*} z krotnością łuku równą jedności - opisane w pracy [101] możemy uogólnić na sieci z łukami o większej krotności - otrzymując zależność

$$\gamma(I,x^{*}) \geq \frac{\sum_{t_j \in T} \left(\sum_{p_i \in {}^{\cdot}t_j} J_i \, \bar{c}_{ij} \right) I_j \, \mathcal{T}(t_j)}{\sum_{t_j \in T} \left(\sum_{p_i \in {}^{\cdot}t_j} J_i \right) \left(\sum_{p_i \in {}^{\cdot}t_j} M_0(p_i) \right)}, \qquad \text{(LB 1)}$$

gdzie $J_i = J(p_i)$, $I_j = I(t_j)$, J - P-niezmiennik, I - T-niezmiennik.

W celu wyznaczenia dolnego oszacowania czasu cyklu możemy wykorzystać rezultaty pracy [111]. W pracy tej zawarte jest oszacowanie dla sieci z czasem przypisanym miejscom. Ze względu na możliwość transformacji sieci z czasem przypisanym przejściom w sieć z czasem przypisanym miejscom (zgodnie z pracą [113]), możemy wykorzystać wyniki pracy [111]. Jednakże wymieniona transformacja wymaga dodania, do oryginalnej SP $\mathcal{N} = \langle P,T,F,W \rangle$, $|T|$ miejsc i $|T|$ przejść (czego ilustracją są rys. 2.22, 2.23).

Z kolei omówimy dolne oszacowanie z pracy [73]. W tym punkcie przyjmiemy inną interpretację palenia przejścia. Poprzednio zakładaliśmy, że w chwili rozpoczęcia palenia przejścia t_j - z miejsc $p_i \in {}^{\cdot}t_j$ usuwane są znaczniki w liczbie $\bar{c}_{ij}$. Obecnie znaczniki te traktujemy, od chwili rozpoczęcia palenia przejścia t_j do chwili zakończenia palenia tego przejścia, jako znaczniki znajdujące się w tym miejscu, ale zarezerwowane. Przez co znaczniki te nie mogą być wykorzystane do przygotowania innych przejść do palenia.

Dla P-niezmienniczej SP prawdziwa jest zależność:

$J^T M = J^T M_0$ dla każdego znakowania M osiągalnego z M_0.

Niech $M_{i_1}, M_{i_2}, \ldots, M_{i_p}$ będą znakowaniami kolejno osiągalnymi przez P-niezmienniczą DSP w czasie cyklu oraz niech $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_p}$ będą czasami trwania tych znakowań. Wówczas średnie znakowanie M^{*} i P-niezmiennik J czynią zadość zależnościom

$$J^T M^{*} = J^T \sum_{k=1}^{p} M_{i_k} \frac{\pi_{i_k}}{\sum_{k=1}^{p} \pi_{i_k}} = \sum_{k=1}^{p} J^T M_{i_k} \frac{\pi_{i_k}}{\sum_{k=1}^{p} \pi_{i_k}} =$$

$$= \sum_{k=1}^{p} J^T M_0 \frac{\pi_{i_k}}{\sum_{k=1}^{p} \pi_{i_k}} = J^T M_0 .$$

Iloczyn $c^-_{ij}\ I_j$ reprezentuje liczbę znaczników usuwanych z miejsca p_i w rezultacie I_j paleń przejścia t_j w czasie cyklu. Suma czasów pobytu $c^-_{ij}\ I_j$ znaczników w miejscu p_i jest co najmniej równa $c^-_{ij}\ I_j\ \mathcal{T}(t_j)$. Suma czasów pobytu wszystkich znaczników w miejscu p_i w czasie cyklu jest równa co najmniej

$$\sum_{t_j \in p_i^\bullet} c^-_{ij}\ I_j\ \mathcal{T}(t_j).$$

Wartość $\gamma\ M^{\#}(p_i)$ (γ - czas cyklu) wyraża czas pobytu znaczników w miejscu p_i w czasie cyklu. Zatem czas cyklu γ musi spełniać relację

$$\gamma\ M^{\#}(p_i) \geq \sum_{t_j \in p_i^\bullet} c^-_{ij}\ I_j\ \mathcal{T}(t_j).$$

Wykorzystując P-niezmiennik J, otrzymujemy

$$\gamma \geq \frac{\sum_{p_i \in P} J_i \left(\sum_{t_j \in p_i^\bullet} c^-_{ij}\ I_j\ \mathcal{T}(t_j) \right)}{\sum_{p_i \in P} J_i\ M_0(p_i)}. \qquad \text{(LB 2)}$$

Porównajmy dolne oszacowania LB1 i LB2. Liczniki obu oszacowań są równe. Natomiast dla mianowników prawdziwe są relacje

$$D_1 = \sum_{t_j \in T} \left(\sum_{p_i \in {}^\bullet t_j} J_i \right) \left(\sum_{p_i \in {}^\bullet t_j} M_0(p_i) \right) =$$

$$= \sum_{p_i \in P} J_i\ M_0(p_i) + \sum_{t_j \in T} \left(\sum_{p_i \in {}^\bullet t_j} J_i \sum_{\substack{p_k \in {}^\bullet t_j \\ p_k \neq p_i}} M_0(p_k) \right),$$

gdzie $\sum_{p_i \in P} J_i\ M_0(p_i) = D_2$.

Twierdzenie 2.17 [73]

Dolne oszacowanie LB2 minimalnego czasu cyklu jest lepsze niż oszacowanie LB1.

Złożoność obliczeniowa procesu wyznaczania oszacowania LB2 równa jest $O(m\ n)$, gdzie $m = |P|$, $n = |T|$.

W celu uzyskania lepszego dolnego oszacowania, możemy zdekomponować P-niezmienniczą SP na podsieci P-niezmiennicze. Metody dekompozycji przedstawione są w pracach [111], [112]. Dla każdej otrzymanej podsieci P-niezmienniczej możemy wyznaczyć dolne oszacowanie LB2. Ze zbioru uzyskanych oszacowań należy wybrać największe.

W celu uzyskania górnego oszacowania minimalnego czasu cyklu, wybieramy pewną czasową sekwencję palenia o wektorze palenia równym T-niezmiennikowi I. Możemy tu wykorzystać rezultaty pracy [127], posługując się diagramem palenia.

Pierwsze dolne oszacowanie minimalnego czasu cyklu dla sieci nie P-niezmienniczych przedstawione jest w pracy [73]. Obecnie przedstawimy to oszacowanie.

2.5.2. Sieci nie P-niezmiennicze

Dla nie P-niezmienniczej DSP $\mathcal{N}^t$ możemy zbudować P-niezmienniczą DSP $\mathcal{N}^{t\bullet}$ o identycznym minimalnym czasie cyklu. W tym celu dodajemy do sieci $\mathcal{N}^t$ miejsce p_{m+1}. Składowe macierzy incydencji $C^\bullet$ sieci $\mathcal{N}^{t\bullet}$ dla miejsca p_{m+1} są następujące

$$c^\bullet_{m+1,j} = -\sum_{i=1}^{m} c_{ij}, \qquad j = 1,\ldots,n,$$

gdzie c_{ij} są składowymi macierzy incydencji C sieci $\mathcal{N}^t$. Dzięki powyższemu układowi równości - dla sieci wynikowej $\mathcal{N}^{t\bullet}$ istnieje P-niezmiennik $J^{\bullet T} = \langle 1,\ldots,1\rangle$ o m+1 składowych.

Znakowanie początkowe $M^\bullet_0$ sieci $\mathcal{N}^{t\bullet}$ zdefiniowane jest równościami

$$M^\bullet_0(p_i) = M_0(p_i) \quad \text{dla każdego } p_i \in P,$$

$$M^\bullet_0(p_{m+1}) = \sum_{t_j \in T} c^{\bullet -}_{m+1,j}\, I_j,$$

gdzie $c^{\bullet -}_{m+1,j} = \begin{cases} c^\bullet_{m+1,j}, & \text{jeśli } c^\bullet_{m+1,j} < 0, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$

Podsieć $\mathcal{N}^t$ sieci $\mathcal{N}^{t\bullet}$ jest T-niezmiennicza. Zatem po pewnej sekwencji palenia σ, której wektor palenia równy jest T-niezmiennikowi I, znakowanie miejsc podsieci $\mathcal{N}^t$ jest odtwarzane. Z tej racji oraz ponieważ sieć $\mathcal{N}^{t\bullet}$ jest P-niezmiennicza, znakowanie miejsca p_{m+1} jest również odtwarzane. Zatem wektor I jest T-niezmiennikiem dla sieci $\mathcal{N}^{t\bullet}$.

Znakowanie początkowe $M^\bullet_0(p_{m+1})$ gwarantuje, że sekwencja palenia σ, której wektor palenia równy jest T-niezmiennikowi I, jest sekwencją palenia DSP $\mathcal{N}^{t\bullet}$ wtedy i tylko wtedy, gdy σ jest sekwencją palenia sieci $\mathcal{N}^t$. Zatem minimalne czasy cyklu sieci $\mathcal{N}^t$, $\mathcal{N}^{t\bullet}$ dla T-niezmiennika I są równe.

Twierdzenie 2.18 [73]

Dla nie P-niezmienniczej DSP i zadanego T-niezmiennika I można zbudować P-niezmienniczą DSP o identycznym minimalnym czasie cyklu dla tego T-niezmiennika. ∎

Stąd dla otrzymanej P-niezmienniczej DSP możemy wykorzystać oszacowanie LB2 - otrzymując w ten sposób dolne oszacowanie czasu cyklu dla nie P-niezmienniczej DSP.

2.6. Podsumowanie

Udowodniliśmy twierdzenia z zakresu zależności między SP^* bez konfliktów w przód (w tył) i bezkonfliktowymi SP^* bez pętli, a grafami synchronizacji oraz między SP^* bez współbieżności w przód (w tył) a automatami.

Wykazaliśmy twierdzenie o rozstrzygalności OPCC i DPCC. Twierdzenie słabsze, zgodnie z którym $OPCC_1$ i $DPCC_1$ są rozstrzygalne, zawarte jest w pracy [101].

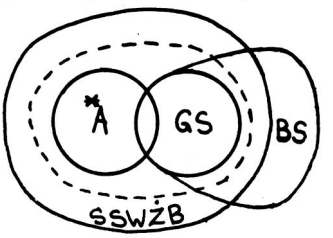
Zaprezentowaliśmy pierwszy algorytm wielomianowy dla OPCC dla grafu synchronizacji.

W pracy [101] udowodniono, że DPCC dla sieci P-niezmienniczych jest NP-trudny. W rozdziale niniejszym przedstawiliśmy twierdzenia znacznie mocniejsze, a mianowicie dowiedliśmy silnej NP-trudności dla podklas klasy sieci P-niezmienniczych.

Udowodniliśmy silną NP-trudność DPCC dla klasy sieci nie P-niezmienniczych będącej dopełnieniem klasy sieci P-niezmienniczych.

Podsumujmy wyniki dotyczące granicy między klasami sieci, dla których dla OPCC istnieje algorytm wielomianowy, a klasami z NP-trudnym DPCC. Z jednej strony, granica ta znajduje się między klasą SP^* zawierającą grafy synchronizacji i automaty, a klasą SP^* swobodnego wyboru posiadających żywe i bezpieczne znakowanie. Z drugiej strony, granica ta usytuowana jest między grafami synchronizacji a bezkonfliktowymi SP^*. Przy czym nawet dla bezpiecznych sieci bezkonfliktowych DPCC jest silnie NP-trudny.

Na rys. 2.29 przedstawiamy graficzną interpretację wyników dotyczących granicy złożoności obliczeniowej między łatwymi a trudnymi PCC^*. Na rysunku tym linia przerywana oznacza wymienioną granicę. Wynik opisany "*" jest zawarty w pracy [102], natomiast pozostałe rezultaty zostały uzyskane przez autora. Znaczenie skrótów podano w punkcie 2.2.



Rys. 2.29

Dla klas sieci P-niezmienniczych i nie P-niezmienniczych przedstawiliśmy wielomianowe dolne oszacowania czasu cyklu.

Podaliśmy warunki, przy których DPCC jest silnie NP-zupełny. Warunki te często są zgodne z cechami sytemów rzeczywistych.

3. OCENA WYDAJNOŚCI SYSTEMÓW PROCESÓW SEKWENCYJNYCH Z WZAJEMNYM WYKLUCZANIEM

Procesy w środowisku zwartym ubiegają się o zasoby - konkurując w dostępie do nich. Wiele z zasobów fizycznych (np. procesor, dysk, taśma, czytnik, drukarka, jednostka funkcjonalna architektury komputera), czy programowych (np. procedura o jednej kopii w pamięci operacyjnej, struktura danych, do której w danej chwili może mieć dostęp tylko jeden proces) wymaga wzajemnego wykluczania w dostępie do zasobów.

Komunikacja między procesami w środowisku zwartym wymaga dostępu procesów do wspólnej pamięci.

Rozważmy system czasu rzeczywistego opisany w punkcie 1.1.

Korzystają ze wspólnej pamięci: proces wejściowy zapisujący bloki danych pomiarowych do wydzielonego obszaru pamięci oraz proces przetwarzający bloki pobierane z tego obszaru. Ponadto inny wydzielony obszar użytkują: proces przetwarzania wysyłający bloki danych sterujących do tego obszaru oraz proces wyjściowy - pobierający te bloki.

W przypadku systemu wsadowego również istnieją pary procesów komunikujących się poprzez wspólną pamięć. Korzystają ze wspólnej pamięci: proces wejściowy zapisujący karty do wydzielonego obszaru pamięci i proces kompilacji programu tworzonego z zapisywanych kart. Innym przykładem jest para procesów: proces przetwarzania wprowadzający wyniki do wydzielonego obszaru i proces wyjściowy wyprowadzający wyniki z tego obszaru na drukarkę.

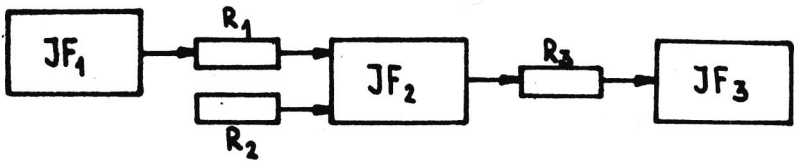
Dostęp do wspólnej pamięci realizowany jest zwykle w trybie wzajemnego wykluczania między procedurami operującymi na tej pamięci.

Do mechanizmów programowych gwarantujących wzajemne wykluczanie należą semafory Dijkstry [34] oraz monitory Hoare'a [48] rozwinięte przez Brinch Hansena w języku Concurrent Pascal [20].

Rozważamy cykliczne procesy sekwencyjne m.in. ze względu na ich znaczenie w systemach operacyjnych.

Przykładem systemu cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem jest system operacyjny systemu czasu rzeczywistego przedstawiony w punkcie 1.1. W tym przypadku - zasobami, do których dostęp realizowany jest w trybie wzajemnego wykluczania są: dwa dyski wirtualne oraz dysk realny.

Przykłady cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem można również wskazać w funkcjonowaniu komputerów równoległych. Rozważmy przykładowy fragment architektury takiego komputera ilustrowany rys. 3.1.



Rys. 3.1

Niech proces jednostki funkcjonalnej JF_2 będzie cyklicznym procesem sekwencyjnym:

CYKL
  POBRANIE ARGUMENTU Z R_1
  POBRANIE ARGUMENTU Z R_2
  DODANIE DWU ARGUMENTÓW
  WYSŁANIE WYNIKU DO R_3.

Niech procesy jednostek JF_1, JF_3 będą również cyklicznymi procesami sekwencyjnymi.

W tym przykładzie komunikacja między cyklicznymi procesami sekwencyjnymi odbywa się w trybie wzajemnego wykluczania w dostępie do rejestrów R_1, R_3.

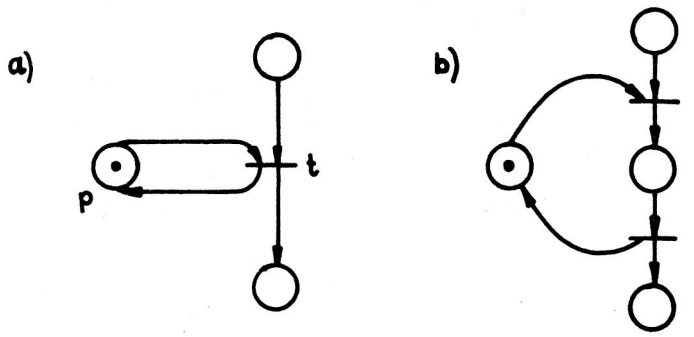
W punkcie pierwszym rozdziału scharakteryzowana jest badana klasa systemów cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem. Punkty drugi i trzeci zawierają analizę przypadku deterministycznego i stochastycznego. Rozdział kończy podsumowanie.

3.1. Charakterystyka badanej klasy systemów cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem

W celu zamodelowania wzajemnego wykluczania, wprowadzamy dodatkowe miejsce dla każdej jednostki zasobu lub dla każdego typu zasobów. Jeśli w miejscu tym znajduje się kropka, to znaczy, że dostępna jest dana jednostka zasobu lub jedna jednostka danego typu zasobów. Niech dla realizacji operacji wyrażonej przejściem t niezbędny jest zasób, z którym skojarzone jest miejsce p. Zatem na rzecz palenia przejścia t - z miejsca p pobierana jest kropka.

Zakładamy, że zasoby są przydzielane tylko na czas jednej operacji, a następnie zwracane. Wykorzystanie zasobu na rzecz następnej operacji wymaga kolejnego jego przydziału. Zatem jeśli z miejsca p przypisanego zasobowi skierowany jest łuk do przejścia t, to musi być skierowany łuk z tego przejścia do miejsca p. Stąd w sieci modelującej dostęp do zasobów istnieje fragment opisany rys. 3.2a. Natomiast

w sieci tej nie może wystąpić fragment z rys. 3.2b modelującej przydział zasobu dla dwu kolejnych operacji bez jego zwracania między tymi operacjami.



Rys. 3.2

Przyjmujemy, że w czasie przydzielania zasobów jednej operacji - nie są jednocześnie przydzielane zasoby dla innej operacji.

Zakładamy, że cykliczny proces sekwencyjny, przy pominięciu dostępu do zasobów, jest żywą ZSP.

Przyjęte założenia gwarantują, że ZSP, wyrażająca system cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem, jest żywa.

3.2. Przypadek deterministyczny

Szukamy granicy między takimi systemami cyklicznych procesów sekwencyjnych, dla których istnieje algorytm wielomianowy dla OPCC a systemami, dla których DPCC jest NP-trudny.

Uwaga 3.1

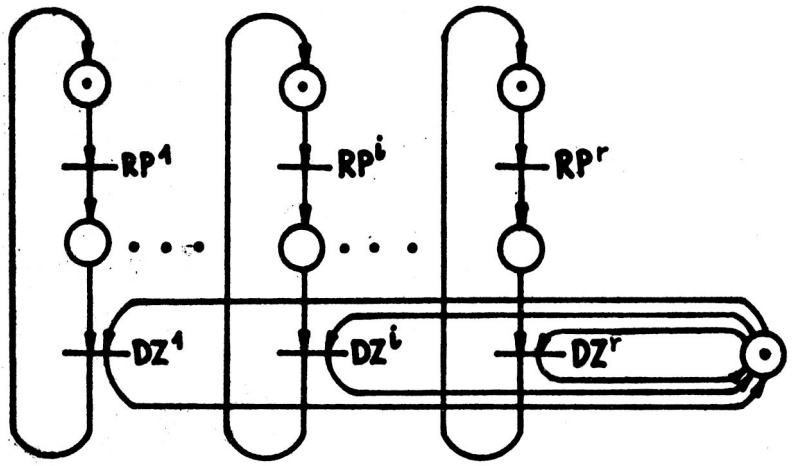
Rozważać będziemy tylko systemy procesów z jednym zasobem, w pewnych przypadkach bowiem jeden zasób wystarcza do NP-trudności DPCC.

W pierwszej kolejności przeanalizujemy systemy cyklicznych procesów sekwencyjnych opisanych obwodami z jednym znacznikiem.

Na początku przebadamy procesy z jednym żądaniem zasobu w czasie cyklu systemu. System cyklicznych procesów sekwencyjnych opisanych obwodami z jednym na cykl pracy systemu dostępem do zasobu może być wyrażony ZSP z rys. 3.3.

Na rysunku tym DZ^i oznacza dostęp do zasobu ze strony i-tego procesu, natomiast RP^i - pozostałą część tego procesu. Dolnym ograniczeniem LB czasu cyklu powyższej sieci jest

$$LB = \max \{ \mathcal{T}(P^1), \ldots, \mathcal{T}(P^r), \mathcal{T}(DZ) \} ,$$



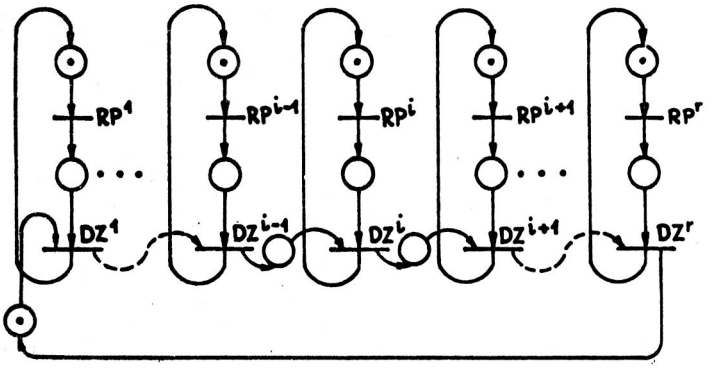
Rys. 3.3

gdzie

$$\mathcal{T}(P^i) = \mathcal{T}(RP^i) + \mathcal{T}(DZ^i), \qquad i \in \{1,\ldots,r\}$$

$$\mathcal{T}(DZ) = \sum_{i=1}^{r} \mathcal{T}(DZ^i).$$

Udowodnimy, że wartość LB jest osiągalna. Załóżmy, że procesy korzystają z zasobu w kolejności $P^1,P^2,\ldots,P^r$. Dla takiej kolejności możemy sieć z rys. 3.3 przetransformować w graf znakowany przedstawiony na rys. 3.4.



Rys. 3.4

Dla grafu znakowanego minimalny czas cyklu równy jest [102]

$$\gamma(<1,\ldots,1>^T, x^M) = \max_{C_i \in C^M} \frac{\mathcal{T}(C_i)}{M(C_i)},$$

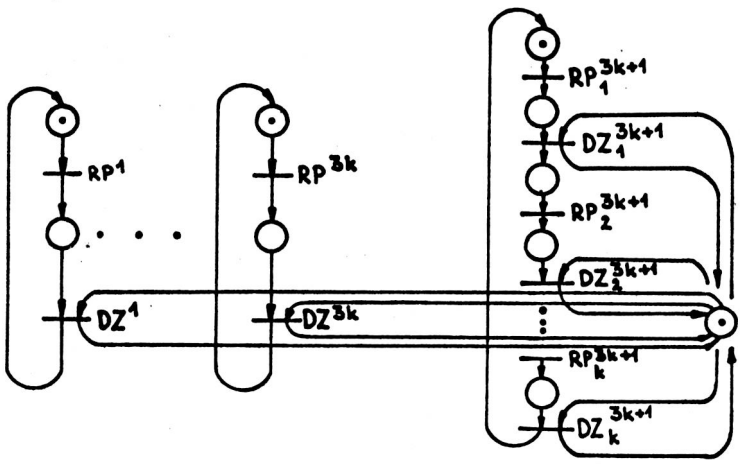
gdzie $\mathcal{T}(C_i)$ jest sumą czasów palenia przejść w obwodzie C_i, $M(C_i)$ - liczbą znaczników w miejscach obwodu C_i, C^M - zbiorem wszystkich obwodów w grafie znakowanym. Graf z rys. 3.4 zawiera dokładnie r+1 obwodów z $\mathcal{T}(C_1) = \mathcal{T}(P^1), \ldots, \mathcal{T}(C_r) = \mathcal{T}(P^r)$, $\mathcal{T}(C_{r+1}) = \mathcal{T}(DZ)$, przy czym w każdym z nich jest dokładnie jeden znacznik. A zatem minimalny czas cyklu dla sieci z rys. 3.3 wynosi $\gamma(<\underbrace{1,1,\ldots,1}_{2r \text{ jedynek}}>^T, x^M) = LB$.

Algorytm 3.1

Krok 1. Obliczyć wartości $\mathcal{T}(P^1), \ldots, \mathcal{T}(P^r)$, $\mathcal{T}(DZ)$.

Krok 2. Wyznaczyć maksymalną wartość spośród uzyskanych w kroku 1.

Złożoność obliczeniowa kroku 1 wynosi O(r). Taka sama jest złożoność kroku 2.



Rys. 3.5

Twierdzenie 3.1 [74]

Dla systemu cyklicznych procesów sekwencyjnych, opisanych obwodami z jednym w czasie cyklu systemu dostępem do zasobu, istnieje algorytm wielomianowy dla OPCC o złożoności obliczeniowej równej O(r), gdzie r jest liczbą procesów. ∎

Dla systemu cyklicznych procesów sekwencyjnych, opisanych obwodami, w których dla wszystkich procesów z wyłączeniem jednego - liczba dostępów do zasobu w czasie cyklu systemu równa jest jedności, udowodnimy silną NP-trudność DPCC. W tym celu przeprowadzimy pseudowielomianową transformację Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla sieci z rys. 3.5 - opisującej system 3k+1 procesów.

Zakładamy, że:

1. $\mathcal{T}(DZ^i) = x_i$ dla $i \in \{1,\ldots,3k\}$,
   $\mathcal{T}(DZ_i^{3k+1}) = 0$ dla $i \in \{1,\ldots,k\}$,
   $\mathcal{T}(RP_i^{3k+1}) = B$ dla $i \in \{1,\ldots,k\}$,
   $k\,B \geq \mathcal{T}(RP^i) + \mathcal{T}(DZ^i)$ dla $i \in \{1,\ldots,3k\}$,
2. T-niezmiennik I zadany jest równością $I(t) = 1$ dla każdego przejścia,
3. Wartość funkcji kryterialnej: $\gamma = k\,B$.

Dowód silnej NP-trudności DPCC dla badanego systemu procesów jest podobny do dowodu tw. 2.12.

Twierdzenie 3.2

DPCC dla systemu cyklicznych procesów sekwencyjnych, opisanych obwodami, w których dla wszystkich procesów z wyłączeniem jednego - liczba dostępów do zasobu w czasie cyklu systemu równa się jedności, jest silnie NP-trudny. ∎

Twierdzenie słabsze - zgodnie z którym DPCC dla rozpatrywanego systemu jest NP-trudny, podaliśmy w pracy [74].

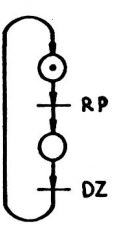
Obecnie przebadamy systemy cyklicznych procesów sekwencyjnych opisanych za pomocą automatów z jednym znacznikiem.

W pierwszej kolejności rozważamy procesy z jednym dostępem do zasobu w czasie cyklu systemu, a następnie procesy bez ograniczenia na liczbę dostępów do zasobu w czasie cyklu.

Cykliczny proces sekwencyjny reprezentowany automatem z jednym dostępem do zasobu w czasie cyklu pracy systemu może być zinterpretowany jako obwód z rys. 3.6.

Przejście DZ oznacza dostęp do zasobu, natomiast RP - pozostałą część procesu, na którą składają się wszystkie przejścia, z wyjątkiem przejścia DZ.

W tym przypadku możemy na podstawie tw. 3.1 sformułować następujące twierdzenie.



Rys. 3.6

Twierdzenie 3.3 [77]

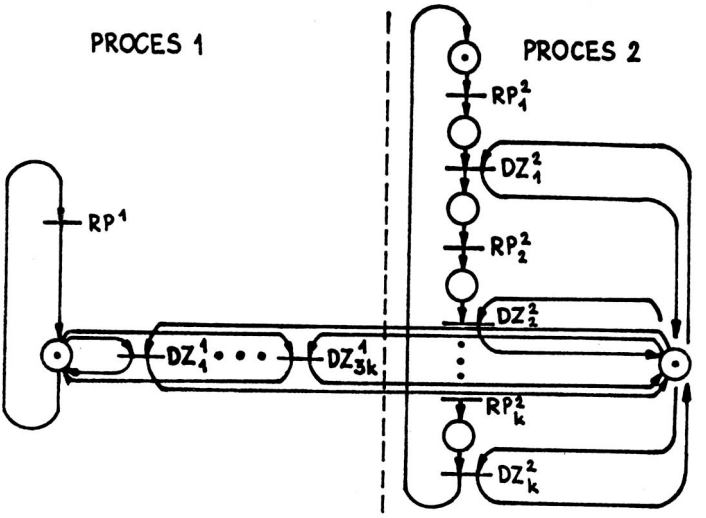
Dla systemu cyklicznych procesów sekwencyjnych, opisanych automatami z jednym w czasie cyklu dostępem do zasobu, istnieje algorytm wielomianowy dla OPCC o złożoności obliczeniowej równej O(r), gdzie r jest liczbą procesów. ■

Z kolei przeanalizujemy procesy opisane automatami bez ograniczenia na liczbę dostępów do zasobu w czasie cyklu systemu. W tym przypadku udowodnimy, że DPCC jest silnie NP-trudny dla dwu procesów, czyli jest silnie NP-trudny również dla większej liczby procesów.

W celu udowodnienia silnej NP-trudności DPCC dla dwu procesów posłużymy się siecią z rys. 3.7.

W definicji pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla sieci z rys. 3.7 zakładamy, że

1. $\mathcal{T}(DZ_i^1) = x_i$ dla $i \in \{1,\ldots,3k\}$,
   $\mathcal{T}(RP^1) = 0$,
   $\mathcal{T}(DZ_i^2) = 0$ dla $i \in \{1,\ldots,k\}$,
   $\mathcal{T}(RP_i^2) = B$ dla $i \in \{1,\ldots,k\}$,



Rys. 3.7

2. T-niezmiennik I równy jest I(t) = 1 dla wszystkich przejść,
3. Wartość funkcji kryterialnej: $\gamma = k\,B$.

Dowód silnej NP-trudności rozważanego systemu procesów jest podobny do dowodu tw. 2.12.

Twierdzenie 3.4

DPCC dla systemu cyklicznych procesów sekwencyjnych wyrażonych automatami jest silnie NP-trudny dla dwu i więcej procesów. ∎

Twierdzenie słabsze, wg którego DPCC dla powyższego systemu jest NP-trudny, przedstawiliśmy w pracy [77].

Ułamek ρ_i wykorzystania zasobu Z_i dla T-niezmiennika I oraz dla czasu cyklu γ zadany jest równością

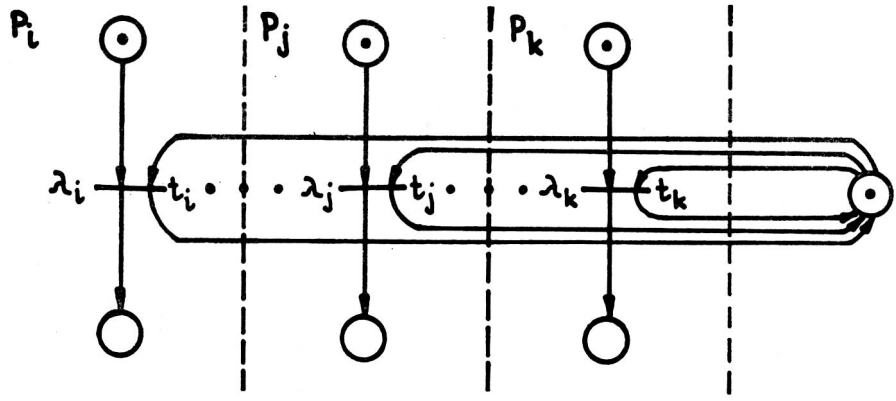
$$\rho_i = \frac{\sum_{t_j \in T} r_{ij}\, I(t_j)\, \tau(t_j)}{\gamma},$$

gdzie T jest zbiorem przejść, r_{ij} jest zmienną równą jedności, jeśli zasób Z_i jest wymagany dla palenia przejścia t_j oraz zeru w przeciwnym przypadku.

3.3. Przypadek stochastyczny

Czas palenia przejścia t_i procesu sekwencyjnego jest zadany zmienną losową o rozkładzie wykładniczym. Rozważania oparte są na USSP*.

Przeanalizujmy wzajemne wykluczanie procesów sekwencyjnych w dostępie do zasobów, z których każdej jednostce przypisane jest jedno miejsce (rys. 3.8).



Rys. 3.8

Zakładamy, że jeśli instrukcja (operacja) do jej realizacji wymaga zasobu, to niezbędna jest tylko jedna jednostka tylko jednego typu zasobu.

Uogólnienie rozważań na przypadek przypisania wszystkim jednostkom zasobu danego typu - jednego miejsca oraz dopuszczenie możliwości

użytkowania jednostek więcej niż jednego typu zasobu przez pewne instrukcje (operacje) nie jest trudne. W celu skrócenia sformułowań, zamiast pojęciem jednostka zasobu danego typu będziemy posługiwali się określeniem zasób.

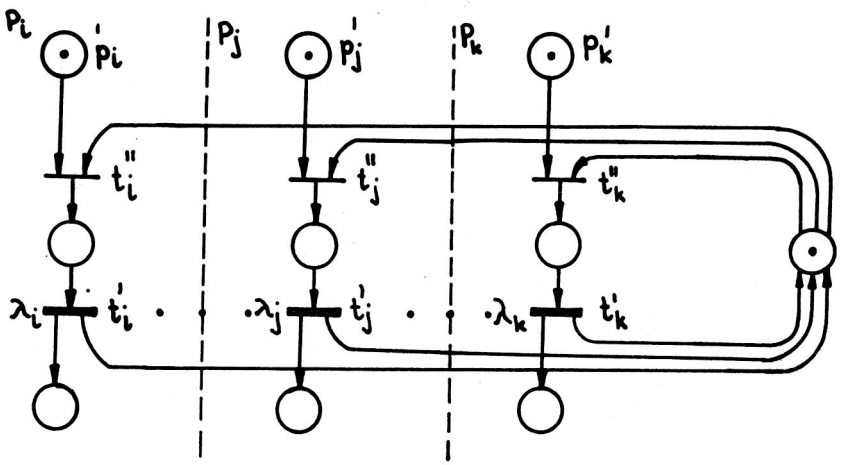
Ponadto przyjmujemy, że wybór kierunku przepływu sterowania nie następuje bezpośrednio przed użytkowaniem zasobu.

Należy przeanalizować skutki zakończenia palenia przejść czasowych dwu typów:

1) przejść wyrażających operacje, do których realizacji niezbędny jest pewien zasób,

2) przejść, do których palenia nie jest wymagana kropka z miejsca przypisanego zasobowi.

W czasie palenia przejścia przyporządkowanego użytkowaniu zasobu można osiągnąć znakowanie, dla którego wymagany jest dostęp do zasobu przez więcej niż jeden proces. Niech $P_i,\ldots,P_j,\ldots,P_k$ będą procesami, które osiągnęły znakowanie poprzedzające użytkowanie zasobu. Zasada wyboru do palenia jednego z przejść $t_i,\ldots,t_j,\ldots,t_k$ (rys. 3.8) określa prawdopodobieństwa palenia tych przejść. ZSP z rys. 3.8 może być zinterpretowana jako USSP z rys. 3.9.



Rys. 3.9

Prawdopodobieństwa palenia przejść $t''_i,\ldots,t''_j,\ldots,t''_k$ są funkcją znakowania sieci. W szczególnym przypadku, gdy jedno z przejść t_j ma priorytet nad pozostałymi, wówczas dla rozważanego znakowania M prawdopodobieństwo palenia przejścia t''_j równe jest $p_M(t''_j) = 1$. Prawdopodobieństwo palenia przejścia t''_j musi być funkcją znakowania, np. bowiem dla znakowania M', dla którego nie ma kropki w miejscu p'_j - przejście to nie może być palone, czyli $p_{M'}(t''_j) = 0$.

Dla procesu interpretowanego za pomocą USSP, stan zanikający jest osiągany, jeśli znacznik znajduje się w miejscu, w którym następuje wybór kierunku przepływu sterowania (miejsce p'_i na rys. 1.14), lub w miejscu poprzedzającym użytkowanie zasobu.

Niech r będzie liczbą procesów sekwencyjnych.

Z własności procesu Markowa wynika, że prawdopodobieństwo jednoczesnego zakończenia palenia przez co najmniej dwa przejścia czasowe jest równe zeru. Zatem, dla co najwyżej jednego z procesów sekwencyjnych może być osiągnięty punkt wyboru kierunku przepływu sterowania. Osiągnięcie tego punktu przez jeden z procesów powoduje osiągnięcie stanu zanikającego przez USSP, wyrażającą system procesów z wzajemnym wykluczaniem. Po wyborze kierunku przepływu sterowania sieć ta osiąga stan uchwytny.

Niech s będzie liczbą miejsc przyporządkowanych zasobom. Podsieć zarządzania zasobem (rys. 3.9) jest w stanie zanikającym, gdy dla co najmniej jednego procesu sekwencyjnego P_j jest znacznik w miejscu wejściowym p'_j przejścia t''_j oraz w miejscu przypisanym zasobowi również jest znacznik.

Gdy proces P_j osiąga znakowanie poprzedzające korzystanie z zasobu, a z zasobu nie korzysta żaden inny proces, wówczas sieć zarządzania zasobem jest w stanie zanikającym. Po przydzieleniu zasobu procesowi P_j (palone jest przejście t''_j) sieć ta osiąga stan uchwytny.

Z własności procesu Markowa wynika, że w danej chwili może nastąpić ukończenie korzystania z zasobu dla co najwyżej jednego zasobu. Jeśli w tej chwili na zwalniany zasób czeka co najmniej jeden proces, to sieć zarządzania tym zasobem osiąga stan zanikający. W skrajnym przypadku mogą żądać dostępu do zasobu wszystkie procesy w liczbie r. Po paleniu jednego z przygotowanych przejść t''_j sieć zarządzania zasobem osiąga stan uchwytny.

Na podstawie danej analizy stwierdzamy, że USSP, modelująca system procesów sekwencyjnych z wzajemnym wykluczaniem, po stanie zanikającym osiąga stan uchwytny w wyniku palenia tylko jednego przejścia bezzwłocznego (tzn. USSP nie przechodzi przez pośrednie stany zanikające).

Rozważmy złożoność obliczeniową wyrażenia (1.5) służącego wyznaczeniu prawdopodobieństw przejść między stanami uchwytnymi USSP modelu-

jącej system procesów z wzajemnym wykluczaniem. Prawdopodobieństwa przejść ze stanów zanikających do uchwytnych są określone wyrażeniem $\sum_{h=0}^{k_0} C^h D$. Ponieważ po stanie zanikającym USSP opisującej ten system nie może nastąpić stan zanikający, a więc $k_0 = 0$. Stąd $\sum_{h=0}^{k_0} C^h D = I D = D$. Zatem macierz W (wyrażenie 1.5) możemy wyznaczyć z zależności

$$W = F + E D.$$

Wyznaczmy złożoność obliczeniową powyższego wyrażenia. Złożoność wyrażenia $E' = E D$ jest $O(K_t K_v K_t) = O(K_t^2 K_v)$. Złożoność sumy $F + E'$ jest $O(K_t^2)$. Zatem złożoność wyrażenia W wynosi $O(K_t^2 K_v)$.

Z kolei przedstawimy podstawy, efektywniejszej niż powyższa, metody wyznaczania prawdopodobieństw przejść między stanami uchwytnymi.

W pierwszej kolejności rozważymy wyznaczanie prawdopodobieństw przejść ze stanów uchwytnych do stanów uchwytnych i zanikających. W stanie uchwytnym i nie jest przygotowane do palenia żadne z przejść bezzwłocznych. W stanie tym, dla każdego z procesów jest przygotowane do palenia dokładnie jedno przejście czasowe. Zatem w stanie uchwytnym przygotowanych jest do palenia r przejść czasowych (r jest liczbą procesów). Prawdopodobieństwa palenia poszczególnych przejść czasowych są zadane zależnością (1.1). Złożoność obliczeniowa wyznaczania prawdopodobieństw przejść ze stanu uchwytnego i do r stanów równa jest $O(r)$. W praktyce prawdziwa jest relacja $r \ll K_t, K_v$. Dla każdego z K_t stanów uchwytnych przeprowadzamy analogiczne rozważania. Zatem złożoność obliczeniowa wyznaczania prawdopodobieństw przejść ze stanów uchwytnych równa jest $O(K_t r)$.

Jeśli stanem osiąganym ze stanu uchwytnego i jest stan uchwytny j, to prawdopodobieństwa otrzymane na podstawie zależności (1.1) są prawdopodobieństwami w_{ij} w wyrażeniu (1.3).

Jeśli stanem osiąganym ze stanu uchwytnego i jest stan zanikający v, to prawdopodobieństwa przejść ze stanu v do stanów uchwytnych wyznaczamy w sposób następujący.

Jeśli USSP, opisująca system procesów sekwencyjnych z wzajemnym wykluczaniem, przechodzi ze stanu zanikającego v do uchwytnego w wyniku palenia przejścia bezzwłocznego t_j, to prawdopodobieństwo takiego zdarzenia jest równe $p_M(t_j)$, gdzie M jest znakowaniem skojarzonym ze stanem v. Liczba l_v stanów uchwytnych osiąganych ze stanu v jest równa:

1. Dla punktu rozpływu sterowania - liczbie przejść bezzwłocznych, do których są skierowane łuki z miejsca odpowiadającego rozgałęzieniu przepływu sterowania, a zawierającego znacznik dla stanu v,

2. W razie żądania dostępu do zasobu - liczbie tych procesów, które osiągnęły znakowanie poprzedzające korzystanie z zasobu, a które ponadto dla znakowania M (skojarzonego ze stanem v) charakteryzują się relacją $p_M(t_j'') > 0$. Złożoność obliczeniowa wyznaczania prawdopodobieństw przejść ze stanu zanikającego v do l_v stanów uchwytnych równa jest $O(l_v)$. W praktyce prawdziwa jest relacja $l_v \ll K_t, K_v$.

Dla prezentacji algorytmu opartego na powyższych stwierdzeniach wprowadzamy oznaczenia:

T_p - zbiór przejść przygotowanych do palenia dla stanu p,

$s \xrightarrow{t_r} u$ - w wyniku palenia przejścia t_r dla stanu s osiągany jest stan u,

U - zbiór stanów uchwytnych,

V - zbiór stanów zanikających.

Algorytm 3.1

Dla każdego ze stanów $i \in U$ przeprowadzić następujące obliczenia:

1. Dla każdego przejścia czasowego $t_p \in T_i$ wyznaczyć prawdopodobieństwa osiągania stanów z takich, że $i \xrightarrow{t_p} z$ na podstawie zależności

$$p_{iz} = \frac{\lambda_p}{\sum_{t_r \in T_i} \lambda_r}.$$

2. Jeśli $z \in U$, to $w_{iz} = p_{iz}$.

3. Jeśli $z \in V$, to wykonać obliczenia:

A. Dla każdego przejścia bezzwłocznego $t_s \in T_z$ wyznaczyć prawdopodobieństwa osiągania stanów $j \in U$ takich, że $z \xrightarrow{t_s} j$ na podstawie równości

$$p_{zj} = p_M(t_s).$$

B. Dla każdego stanu $j \in U$ takiego, że $z \xrightarrow{t_s} j$ wyznaczyć prawdopodobieństwa w_{ij} na podstawie wyrażenia $w_{ij} = p_{iz}\, p_{zj}$.

Złożoność obliczeniowa algorytmu 3.1 jest równa $O(K_t \; r \; \bar{l})$, gdzie $\bar{l}$ jest średnią liczbą stanów uchwytnych osiąganych ze stanu zanikającego liczoną dla wszystkich stanów zanikających.

Charakteryzując wyznaczanie prawdopodobieństw przejść między stanami uchwytnymi USSP, opisującej system procesów z wzajemnym wykluczaniem, na podstawie rozważań poprzedzających algorytm 3.1, formułujemy wniosek.

Wniosek 3.1

Złożoność obliczeniowa algorytmu 3.1 jest, dla przypadków prak-

tycznych, znacznie mniejsza niż złożoność obliczeniowa wyrażenia (1.5).

Korzystając z wartości w_{ij} (wyznaczonych na podstawie algorytmu 3.1) możemy dla zachowania cyklicznego obliczyć, zgodnie z punktem 1.2.2.2, rozkład prawdopodobieństw stanów uchwytnych w stanie stacjonarnym, ułamki czasów pobytu w stanach uchwytnych, średni czas cyklu systemu procesów.

Obecnie wyznaczymy ułamek ρ_i wykorzystania zasobu Z_i. Niech wielkość r_{ij} przyjmuje wartość jeden, jeśli zasób Z_i jest wykorzystywany w stanie uchwytnym j oraz zero w przeciwnym razie. Ułamek ρ_i jest zadany zależnością

$$\rho_i = \sum_{j \in U} r_{ij} v_j,$$

gdzie: U - zbiór stanów uchwytnych,
v_j - ułamek czasu pobytu w stanie j.

3.4. Podsumowanie

Modele z czasem wykonania instrukcji reprezentowanym nieujemną liczbą rzeczywistą są przydatne do opisu programów napisanych w językach programowania niskiego poziomu oraz dla niższych poziomów abstrakcji systemu. Jednakże dla języków wysokiego poziomu, w szczególności dla języków modularnych, wymagany jest stochastyczny model czasu wykonania instrukcji (modułu).

Dla przypadku deterministycznego podsumujmy wyniki dotyczące granicy między systemami cyklicznymi procesów sekwencyjnych, dla których istnieje algorytm wielomianowy dla OPCC a systemami, dla których DPCC jest NP-trudny. Własnością krytyczną dla złożoności obliczeniowej jest liczba dostępów do zasobów w czasie cyklu systemu. Rozpatrywaliśmy systemy z jednym zasobem, w pewnych przypadkach bowiem jeden zasób wystarcza, by DPCC był NP-trudny.

Dla procesów opisanych obwodami interesująca nas granica złożoności obliczeniowej znajduje się między systemami procesów z jednym dostępem do zasobu w czasie cyklu systemu a systemami, w których dla wszystkich procesów, z wyłączeniem jednego, liczba dostępów do zasobu w czasie cyklu równa jest jedności.

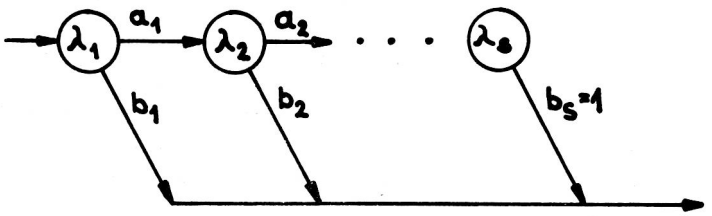
W przypadku procesów wyrażonych automatami granica ta usytuowana jest między systemami procesów z jednym dostępem do zasobu w czasie cyklu dla dowolnej liczby procesów a systemami dwu procesów w razie braku ograniczeń na liczbę dostępów do zasobu.

Rozwiązanie OPCC dla przypadku deterministycznego może być wykorzystane jako dolne oszacowanie wartości średniej czasu cyklu dla przy-

padku mieszanego. W tym celu, należy prawdopodobieństwa wyboru kierunku przepływu sterowania oraz prawdopodobieństwa przydziału zasobu w warunkach konfliktów zasobowych dla przypadku mieszanego przetransformować w składowe T-niezmiennika i rozwiązać OPCC dla przypadku deterministycznego.

Dla przypadku stochastycznego sformułowaliśmy specyficzną dla systemów procesów z wzajemnym wykluczaniem metodę wyznaczania prawdopodobieństw przejść między stanami uchwytnymi, która jest znacznie efektywniejsza obliczeniowo od ogólnej metody macierzowej opisanej w pracy [4].

Czasy palenia przejść czasowych USSP są opisane zmienną losową o rozkładzie wykładniczym. Zatem znaczne uogólnienie stanowi próba wprowadzenia reprezentacji czasu za pomocą zmiennej o rozkładzie Cox'a. Rozkład ten jest rozkładem wielostopniowym (rys. 3.10), przy czym każdy stopień jest opisany rozkładem wykładniczym.



Rys. 3.10

Liczba λ_i jest parametrem rozkładu wykładniczego i-tego stopnia, b_i jest prawdopodobieństwem zakończenia procesu po i-tym stopniu, a_i jest prawdopodobieństwem realizacji stopnia (i+1)-szego po stopniu i-tym.

Ranga rozkładów Cox'a wynika po pierwsze - z faktu przynależenia do tej klasy m.in. takich rozkładów, jak rozkłady wykładniczy i Erlanga. Z drugiej zaś strony klasa rozkładów Cox'a charakteryzuje się dużym zakresem współczynnika zmienności zmiennej losowej określonym następująco:

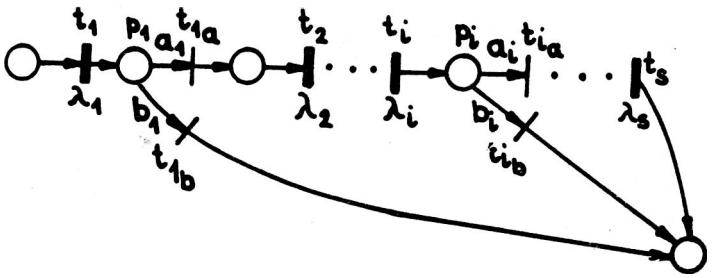
$$C(X) = \frac{\sigma(X)}{E(X)},$$

gdzie

$\sigma(X)$ - odchylenie standardowe zmiennej losowej X,

$E(X)$ - wartość oczekiwana zmiennej losowej X. Stąd klasa ta jest przydatna w aproksymacji wielu innych rozkładów.

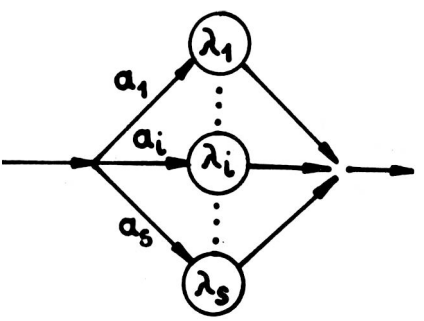
Rozkład Cox'a możemy wyrazić za pomocą USSP z rys. 3.11 [76].



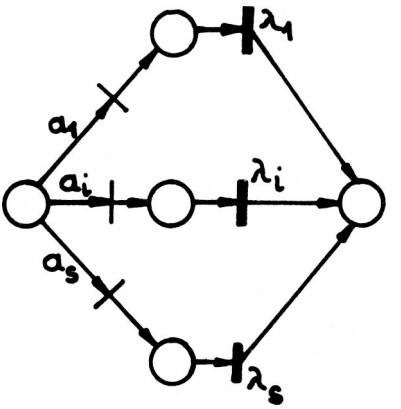
Rys. 3.11

Czas palenia przejścia t_i jest wyrażony zmienną losową o rozkładzie wykładniczym z parametrem λ_i. Przejścia t_{ia}, t_{ib} są przejściami bezzwłocznymi. Dla znakowania zawierającego kropkę w miejscu p_i - prawdopodobieństwa palenia przejść t_{ia}, t_{ib} są równe odpowiednio a_i, b_i. Zatem USSP* umożliwiają wyrażanie czasów wykonywania instrukcji (operacji) opisanych za pomocą rozkładów Coxa.

Rozkład hiperwykładniczy może być zinterpretowany za pomocą struktury przedstawionej na rys. 3.12.



Rys. 3.12



Rys. 3.13

Liczba a_i jest prawdopodobieństwem wyboru zmiennej losowej o rozkładzie wykładniczym z parametrem λ_i, przy czym $\sum_{i=1}^{s} a_i = 1$.

Rozkład hiperwykładniczy możemy wyrazić za pomocą USSP z rys.3.13.

Dotychczas rozważaliśmy przydział zasobów na czas jednej operacji - po czym zasoby były zwalniane. Ponadto zakładaliśmy, iż w czasie

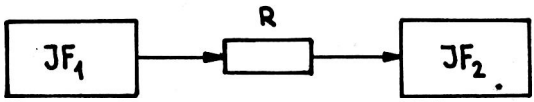
przydzielania zasobów jednej operacji, nie są jednocześnie przydzielane zasoby dla innej operacji. Dzięki powyższym założeniom, rozpatrywane dotychczas systemy procesów są systemami żywymi. Ograniczenia te często nie są spełnione. Przy rezygnacji z nich należy sięgnąć po wyniki z zakresu algorytmów unikania martwego punktu lub algorytmów wyprowadzania systemu z martwego punktu, jeśli zaistnieje [17], [41], [68] - w zależności od wybranej strategii rozwiązywania zagadnienia. Dla tego ogólniejszego problemu, sieci muszą zawierać struktury modelujące wymienione algorytmy lub przynajmniej efekty działania tych algorytmów (co może zmniejszyć złożoność obliczeniową w porównaniu z przypadkiem, gdy algorytmy te są modelowane).

Dla każdego z rozważanych trzech przypadków systemów: deterministycznego, stochastycznego i mieszanego - metodyka oceny wydajności dla problemu ogólniejszego (przy rezygnacji z dwu powyższych założeń dotyczących przydziału zasobów) jest podobna do rozważanej w tym rozdziale. W przypadku deterministycznym złożoność obliczeniowa PCC dla problemu ogólniejszego jest nie mniejsza niż dla problemu rozpatrywanego w pracy. Dlatego analizowana granica złożoności obliczeniowej znajduje się między systemami procesów nie bardziej złożonymi niż rozpatrywane w pracy.

4. OCENA WYDAJNOŚCI KOMUNIKUJĄCYCH SIĘ PROCESÓW SEKWENCYJNYCH (CSP)

Paradygmatyczne języki programowania CSP, CCS zawierają podstawowy mechanizm wymiany komunikatów w środowisku rozproszonym, a mianowicie synchronizację dwu procesów sekwencyjnych. Podobieństwo obu tych języków doprowadziło do stworzenia języka będącego ich syntezą - zwanego CCSP [95]. Ze względu na podobieństwo - podobną analizę do analizy dla języka CSP można przeprowadzić dla języka CCS. Język CSP został wykorzystany w budowie języka Ada, który może odegrać tak dużą rolę w rozwoju programowania współbieżnego, jak język Pascal w programowaniu sekwencyjnym. Ponadto na podstawie języka CSP skonstruowano język Occam dla Transputerów.

Przykłady procesów komunikujących się można wskazać również w zakresie funkcjonowania komputerów równoległych. Przeanalizujmy fragment takiego komputera ilustrowany rys. 4.1.



Rys. 4.1

Niech proces jednostki funkcjonalnej JF_1 przesyła, obliczoną w tym procesie, wartość wyrażenia e pod zmienną x procesu jednostki funkcjonalnej JF_2 poprzez zmienną r procesu rejestru R. Niech działanie każdej z jednostek JF_1, JF_2 oraz rejestru R opisuje cykliczny proces sekwencyjny:

1. PROCES JF_1
   CYKL
   ⋮
   WYSŁANIE WART. WYR. e DO PR. R
   ⋮
2. PROCES R
   CYKL
   PODSTAWIENIE POD ZM. r WART. WYR. Z PR. JF_1
   WYSŁANIE WART. WYR. r DO PR. JF_2
3. PROCES JF_2
   CYKL
   ⋮
   PODSTAWIENIE POD ZM. x WART. WYR. Z PR. R
   ⋮

W pierwszym punkcie rozdziału przedstawiamy model sieciowy dla podzbioru CSP na poziomie syntaktycznym. W kolejnych dwu punktach omawiamy przypadek deterministyczny i stochastyczny. Dla obu tych przypadków badamy zachowanie przejściowe i cykliczne. Rozdział kończy podsumowanie.

4.1. Reprezentacja sieciowa komunikujących się procesów sekwencyjnych

Rozważać będziemy ocenę wydajności CSP na poziomie syntaktycznym. Przedmiotem analizy będzie opisany niżej podzbiór CSP.

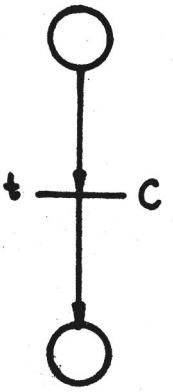
Instrukcję przypisania o postaci

C ::= x := e,

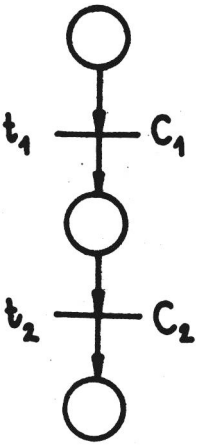
gdzie x jest zmienną, e - wyrażeniem, można przedstawić SP z rys. 4.2.

Złożenie sekwencyjne dwu instrukcji C_1, C_2 oznaczane symbolem C_1; C_2 opisujemy SP z rys. 4.3.

Instrukcję pustą skip ilustruje pojedyncze miejsce.



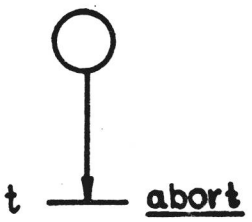
Rys. 4.2



Rys. 4.3

Instrukcja abort, powodująca zerwanie obliczeń, jest reprezentowana za pomocą SP z rys. 4.4.

Komunikacja między dwoma współbieżnie przebiegającymi procesami sekwencyjnymi jest przeprowadzana za pomocą instrukcji wejścia P?x oraz instrukcji wyjścia Q!e, gdzie P,Q są nazwami procesów. Niech instrukcja wejścia P_2?x występuje w procesie P_1 oraz niech instrukcja wyjścia P_1!e występuje w procesie P_2. Instrukcje te wykonywane są współbieżnie, a ich rezultatem jest przypisanie zmiennej x procesu P_1 - wartości wyrażenia e obliczanej w procesie P_2 (x := e). Jeśli jedna z powyższych instrukcji jest gotowa do wykonania wcześniej niż druga, to pierwsza instrukcja jest opóźniana. Model sieciowy komunikacji przedstawia rys. 4.5.



Rys. 4.4

Instrukcja dozorowana ma postać

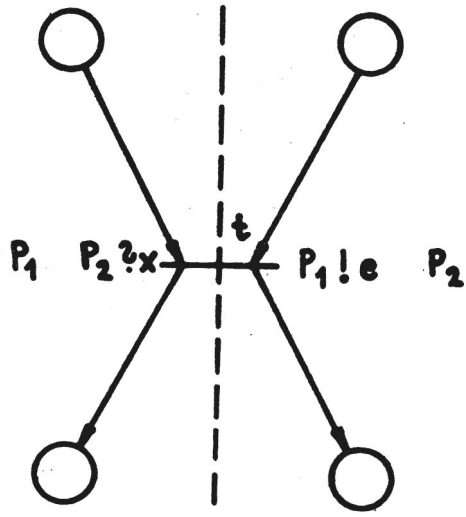
G ⟶ C,

gdzie G - dozór,
C - instrukcja.

Warunkiem koniecznym wykonania instrukcji C jest spełnienie dozoru G. Rozważamy uproszczoną wersję dozorów. Zakładamy, że dozór jest wyrażeniem boolowskim. Nie analizujemy dozorów zawierających instrukcje komunikacji.

Instrukcja alternatywy jest zapisywana w postaci:

$\underline{\text{if}}\ G_1 \longrightarrow C_1 \,[]\, \ldots \,[]\, G_n \longrightarrow C_n\ \underline{\text{fi}}$.



Rys. 4.5

Instrukcja ta określa wykonanie co najwyżej jednej z instrukcji składowych. Jeśli żaden z dozorów nie jest spełniony, to instrukcja alternatywy powoduje błąd. W przeciwnym razie wykonywana jest jedna z tych instrukcji, których dozór jest spełniony. Analizę dozorów i wybór co najwyżej jednej instrukcji do wykonania traktujemy jako pojedynczą akcję wyrażoną przejściem t_0. Rys. 4.6 zawiera SP opisującą instrukcję alternatywy.

Jeśli żaden z dozorów nie jest spełniony, to palone jest przejście t_a.

Instrukcja iteracji zapisywana jest w postaci:

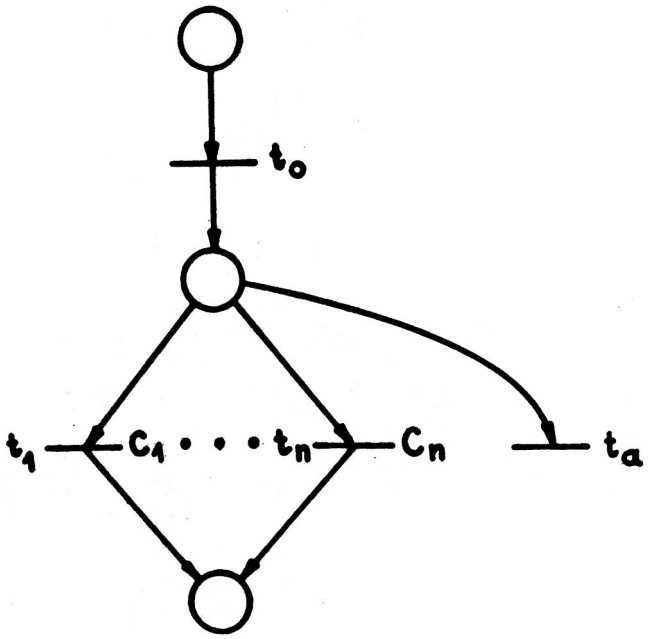
$\underline{\text{do}}\ G_1 \longrightarrow C_1 \,[]\, \ldots \,[]\, G_n \longrightarrow C_n\ \underline{\text{od}}$

Instrukcja ta wyznacza tyle powtórzeń, ile jest możliwe. Jeśli na początku wykonywania instrukcji iteracji żaden z dozorów nie jest spełniony, to instrukcja ta jest kończona bez efektów. W przeciwnym razie do wykonania wybierana jest jedna z tych instrukcji, których dozory są spełnione. Następnie cała instrukcja iteracji jest wykonywana powtórnie. Rys. 4.7 ilustruje model sieciowy instrukcji iteracji.

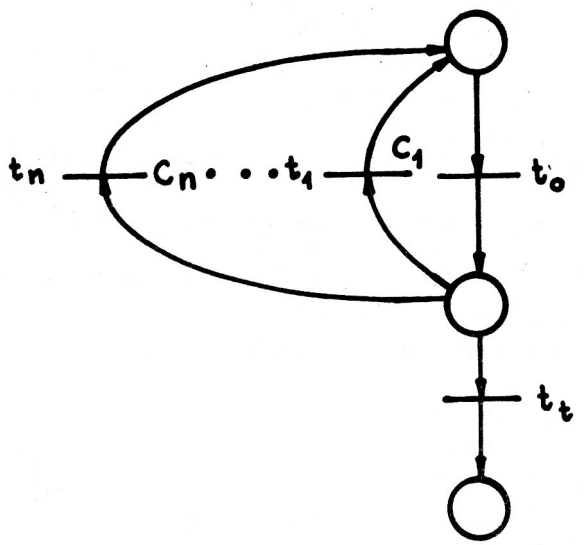
Przejście t_t jest palone, gdy kończone jest wykonanie instrukcji.

Ogólnie, instrukcja C jest zdefiniowana jako

$C ::= x := e \mid C_1; C_2 \mid \underline{\text{skip}} \mid \underline{\text{abort}} \mid P?x \mid P!e \mid$
$\mid \underline{\text{if}}\ G_1 \longrightarrow C_1 \,[]\, \ldots \,[]\, G_n \longrightarrow C_n\ \underline{\text{fi}} \mid$
$\mid \underline{\text{do}}\ G_1 \longrightarrow C_1 \,[]\, \ldots \,[]\, G_n \longrightarrow C_n\ \underline{\text{od}} \mid .$



Rys. 4.6



Rys. 4.7

Program Pr jest złożeniem równoległym procesów sekwencyjnych

$$Pr ::= P_1 :: C_1 \| \ldots \| P_n :: C_n,$$

gdzie P_i - nazwa procesu,
C_i - instrukcja.

Procesy sekwencyjne $P_1,\ldots,P_n$ są wykonywane współbieżnie.

Rozważamy przejściowe i cykliczne zachowanie programu Pr. Zachowanie przejściowe dotyczy programów ze stopem. W przypadku zachowania cyklicznego każdy proces sekwencyjny P_i zdefiniowany jest instrukcją iteracji C_i; ponadto wymagamy, aby ZSP reprezentująca program Pr była żywa.

4.2. Przypadek deterministyczny

Pragniemy znaleźć granicę między przypadkami CSP^*, dla których istnieje algorytm wielomianowy dla OPCW (OPCC), a przypadkami CSP^*, dla których DPCW (DPCC) jest NP-trudny.

4.2.1. Zachowanie przejściowe

W pierwszej kolejności analizować będziemy procesy sekwencyjne opisane za pomocą dróg prostych, a następnie procesy sekwencyjne wyrażone automatami.

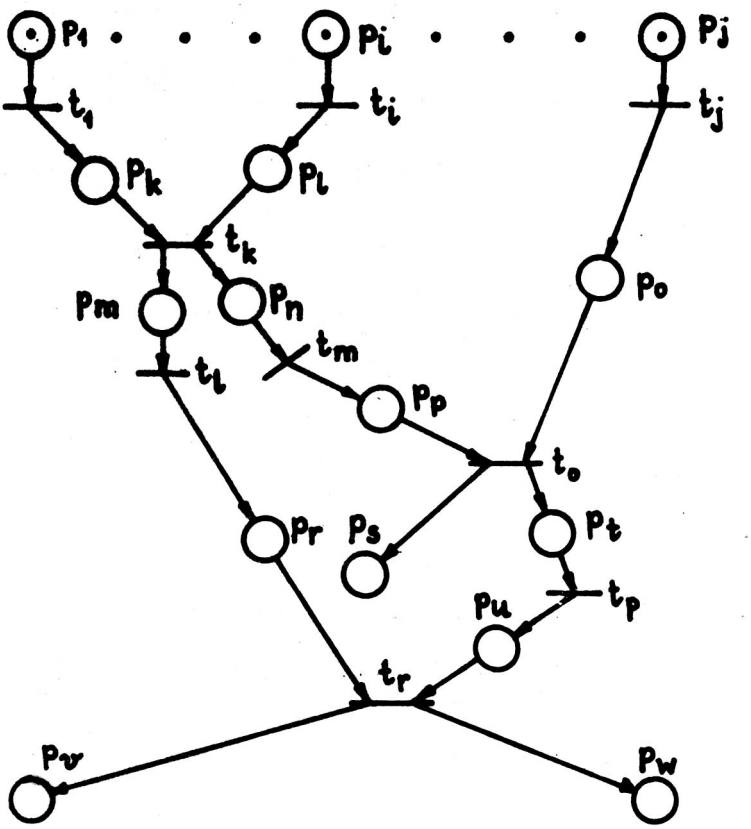
Przykład komunikujących się procesów sekwencyjnych opisanych drogami prostymi przedstawia ZSP z rys. 4.8.

Miejsca $p_1,\ldots,p_i,\ldots,p_j$ są miejscami początkowymi procesów sekwencyjnych $P_1,\ldots,P_i,\ldots,P_j$, podczas gdy miejsca $p_v,\ldots,p_s,\ldots,p_w$ - miejscami końcowymi. Przejście t_k reprezentuje komunikację między procesami P_1,P_i, przejście t_0 - między procesami P_i,P_j, natomiast przejście t_r - między procesami P_1,P_j.

Ponieważ procesy sekwencyjne są opisane drogami prostymi, a więc wektor H, wyrażający zachowanie przejściowe, ma wszystkie składowe równe jedności. Minimalny czas wykonania dla tego wektora jest równy maksymalnemu sumarycznemu czasowi palenia spośród wszystkich dróg skierowanych z miejsc początkowych do miejsc końcowych, biorąc pod uwagę drogi skierowane z miejsca początkowego jednego procesu do miejsca końcowego innego procesu.

Pragniemy znaleźć jak najefektywniejszy algorytm dla OPCW. W tym celu transformujemy sieć z rys. 4.8 w graf skierowany. Przejściu t_i przypisujemy wierzchołek v_i, natomiast miejscu p_j z incydentnymi z nim łukami - przypisujemy łuk a_j. Istnieją trzy sytuacje:

1. Miejsce jest incydentne z jednym łukiem skierowanym do tego miejsca i jednym łukiem skierowanym z miejsca (np. miejsce p_k),



Rys. 4.8

2. Miejsce jest incydentne z jednym łukiem skierowanym z tego miejsca (np. miejsce p_1),

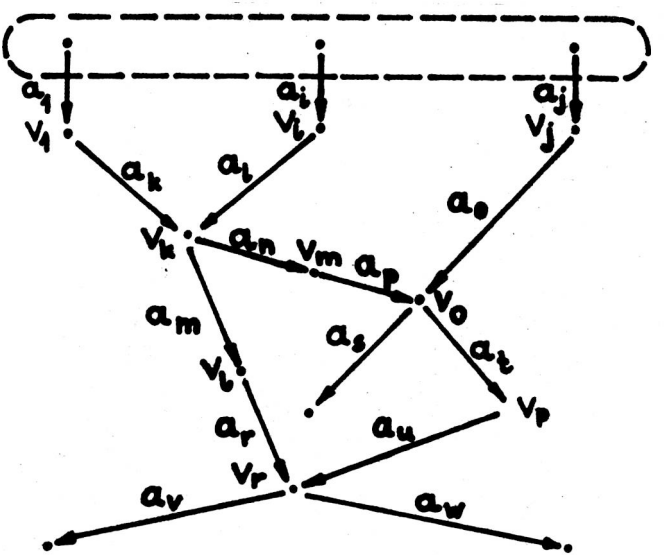
3. Miejsce jest incydentne z jednym łukiem skierowanym do tego miejsca (np. miejsce p_v).

Dla sieci z rys. 4.8 otrzymujemy graf skierowany z rys. 4.9.

Miejscom początkowym (końcowym) sieci są przyporządkowane wierzchołki początkowe (końcowe) grafu.

Czas palenia przejścia t_i równy $\tau(t_i)$ przypisujemy wierzchołkowi v_i, natomiast liczbę zero - wierzchołkom początkowym i końcowym. Wierzchołki początkowe traktujemy jako jeden wierzchołek v_0. Otrzymany graf jest acykliczny.

W celu rozwiązania OPCW, znajdujemy najdłuższą drogę z wierzchołka v_0 do dowolnego z wierzchołków końcowych. Dla problemu najdłuższej drogi w grafie acyklicznym istnieje algorytm [70] o złożoności O(m), gdzie m jest liczbą łuków grafu równą liczbie miejsc w ZSP opisującej CSP.



Rys. 4.9

Twierdzenie 4.1

Dla komunikujących się procesów sekwencyjnych wyrażonych drogami prostymi, OPCW może być rozwiązany za pomocą algorytmu o złożoności obliczeniowej O(m), gdzie m jest liczbą miejsc ZSP reprezentującej CSP. ∎

W przypadku ogólniejszym, gdy proces sekwencyjny wyrażony jest automatem, udowodnimy, że DPCW jest silnie NP-trudny dla dwu procesów. Stąd DPCW jest silnie NP-trudny również dla większej liczby procesów.

Rozważmy ZSP modelującą CSP dla dwu procesów ilustrowaną rys. 4.10.

Obecnie przedstawimy pseudowielomianową transformację Turinga PROBLEMU TRÓJPODZIAŁU do DPCW dla ZSP z rys. 4.10.

Zakładamy, że:

1. Czasy palenia przejść spełniają równości:

$\tau(t_i) = x_i$ dla $i \in \{1,\ldots,3k\}$,

$\tau(t_{3k+j}) = B + 4\tau(t_0)$ dla $j \in \{1,\ldots,k\}$.

2. Wektor H zadany jest równościami:

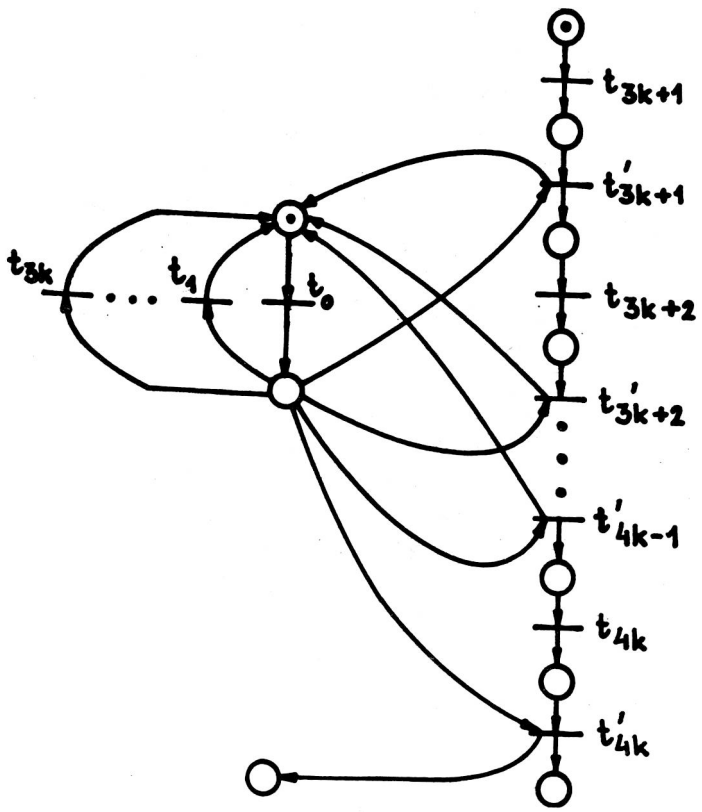
$H(t_0) = 4k$,

$H(t) = 1$ dla pozostałych przejść.

3. Wartość funkcji kryterialnej:

$$\gamma = k\,B + 4k\,\tau(t_0) + \sum_{j=1}^{k} \tau(t'_{3k+j}).$$

Dowód jest podobny do dowodu tw. 2.12.



Rys. 4.10

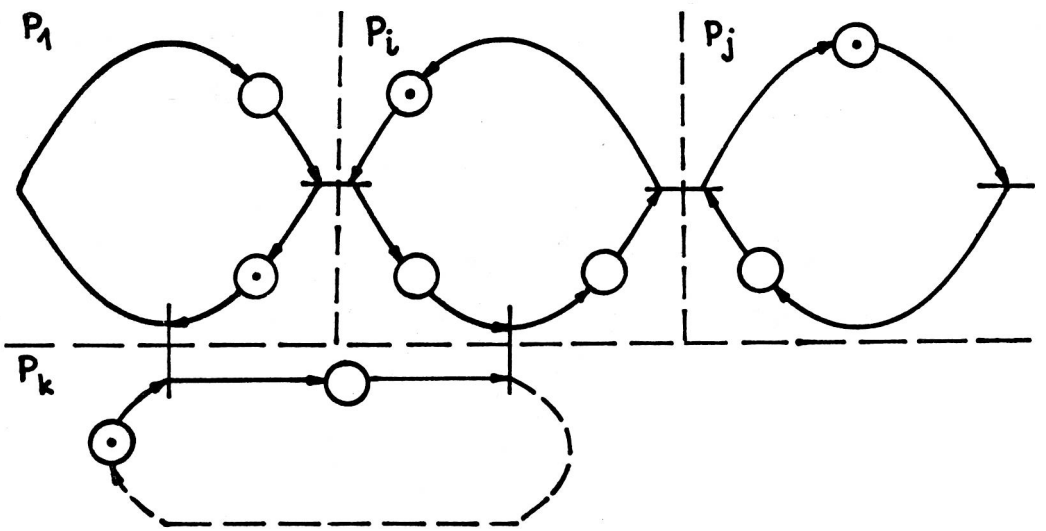
Twierdzenie 4.2

Dla komunikujących się procesów sekwencyjnych reprezentowanych automatami, DPCW jest silnie NP-trudny dla dwu i większej liczby procesów. ∎

Z powyższych rozważań wynika, że granica pomiędzy tymi CSP^{*}, dla których istnieje algorytm wielomianowy dla OPCW, a CSP^{*}, dla których DPCW jest NP-trudny, znajduje się między CSP^{*}, opisanymi drogami prostymi dla dowolnej liczby procesów, a CSP^{*} wyrażonymi automatami dla dwu procesów.

4.2.2. Zachowanie cykliczne

W pierwszej kolejności rozważymy cykliczne procesy sekwencyjne reprezentowane za pomocą obwodów. Przykład ZSP dla tego rodzaju komunikujących się procesów sekwencyjnych ilustruje rys. 4.11.



Rys. 4.11

ZSP* dla komunikujących się procesów sekwencyjnych wyrażonych obwodami są żywymi i bezpiecznymi grafami znakowanymi (SP* są grafami synchronizacji). Żywotność jest konsekwencją założenia przyjętego w punkcie 4.1. Natomiast bezpieczeństwo sieci wynika z obecności tylko jednej kropki w obwodzie opisującym proces.

Dla żywych i ograniczonych grafów znakowanych istnieje algorytm wielomianowy dla OPCC (punkt 2.4.1).

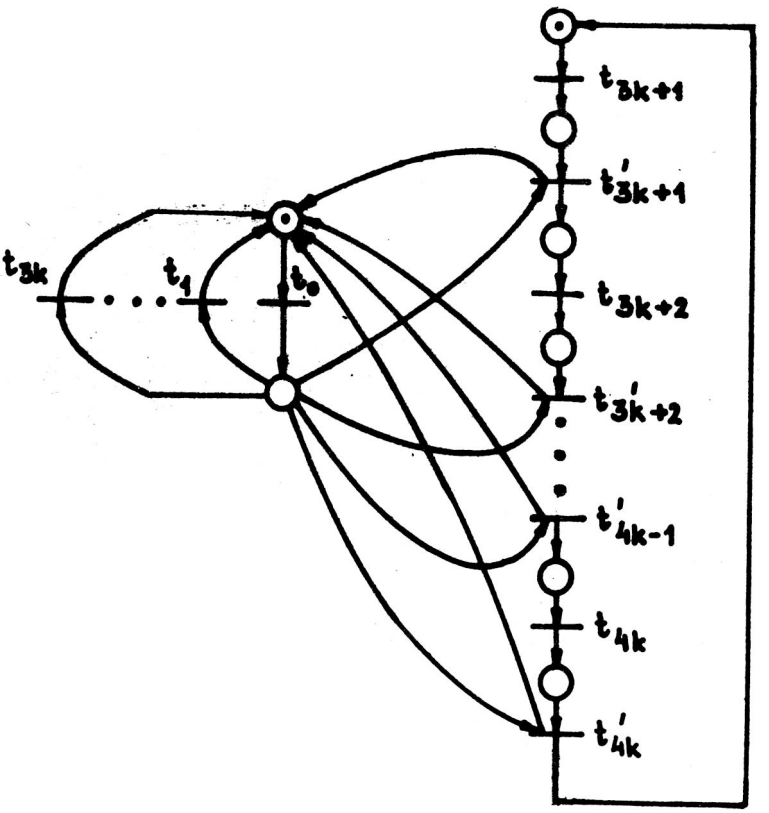
W pracy [36] jest przedstawiony algorytm dla OPCC dla żywych i bezpiecznych grafów znakowanych. Złożoność obliczeniowa tego algorytmu równa jest $O(mnK)$, gdzie $m = |P|$, $n = |T|$, K jest sumaryczna liczbą kropek we wszystkich miejscach $p_i \in P$. Dla CSP*, których procesy są opisane obwodami, liczby m, n spełniają warunek $m \leqslant 2n$, ponieważ każde z przejść ma co najwyżej dwa miejsca wejściowe. Liczba K jest równa liczbie r procesów sekwencyjnych. Zatem otrzymujemy złożoność obliczeniową $O(n^2 r)$, czyli metoda ta jest znacznie efektywniejsza niż algorytm oparty na programowaniu liniowym.

Twierdzenie 4.3

Dla komunikujących się procesów sekwencyjnych reprezentowanych

obwodami, OPCC może być rozwiązany za pomocą algorytmu o złożoności $O(n^2 r)$, gdzie n jest liczbą przejść sieciowego modelu CSP, natomiast r - liczbą procesów. ∎

W celu udowodnienia, że DPCC dla CSP^* o dwu procesach opisanych automatami jest silnie NP-trudny wykorzystamy ZSP z rys. 4.12.



Rys. 4.12

Dowód jest podobny do dowodu tw. 4.2.

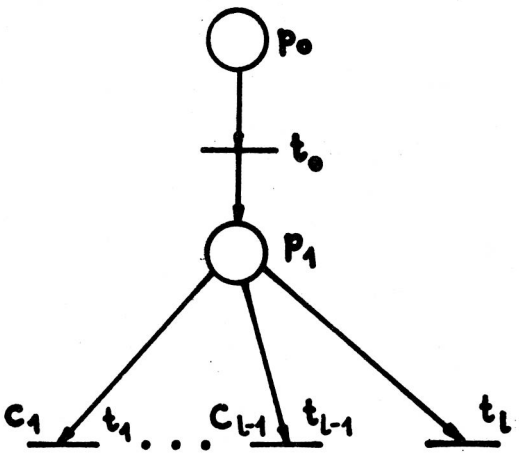
Twierdzenie 4.4

Dla komunikujących się procesów sekwencyjnych wyrażonych automatami, DPCC jest silnie NP-trudny dla dwu i więcej procesów. ∎

Z powyższej analizy wnioskujemy, że interesująca nas granica znajduje się między CSP^*, opisanymi obwodami dla dowolnej liczby procesów, a CSP^* reprezentowanymi automatami dla dwu procesów.

4.3. Przypadek stochastyczny

Rozważania dla tego przypadku są oparte na USSP*. W sieciowej reprezentacji instrukcji alternatywy i iteracji występuje fragment przedstawiony na rys. 4.13.



Rys. 4.13

Przejście t_1 odwzorowuje przejście t_a w przypadku instrukcji alternatywy lub przejście t_t dla instrukcji iteracji. USSP przypisana powyższej sieci jest ilustrowana rys. 4.14. Powyższe dwa rodzaje instrukcji są jedynymi, dla których następuje rozgałęzienie przepływu sterowania.

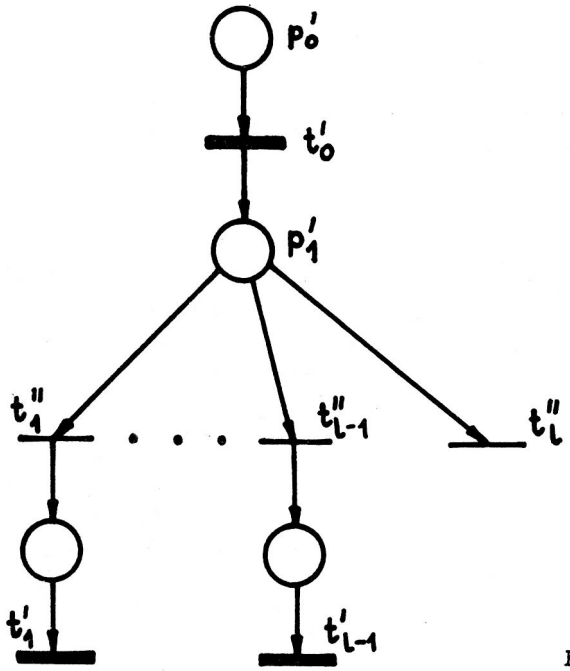
Niech r będzie liczbą procesów sekwencyjnych. Dla pojedynczego procesu - stan zanikający jest osiągany, jeśli znacznik znajduje się w miejscu, w którym następuje rozgałęzienie przepływu sterowania. Na rys. 4.14 miejscem takim jest miejsce p'_1. Znacznik w tym miejscu pojawia się po zakończeniu palenia przejścia czasowego t'_0. Sposób wyznaczania prawdopodobieństw $p(t''_i)$ palenia przejść bezzwłocznych dla stanu, w którym znacznik jest w miejscu p'_1 przedstawiony jest w punkcie 1.2.2.

W zachowaniu USSP, opisującej CSP można wyeksponować proces Markowa. Z własności tego procesu wynika, że prawdopodobieństwo jednoczesnego zakończenia palenia przez co najmniej dwa przejścia czasowe jest równe zeru. Zatem w danej chwili stan zanikający może być osiągnięty dla co najwyżej jednego procesu. Ponadto po paleniu przejścia bezzwłocznego danego procesu - proces ten osiąga stan uchwytny. Z podanych rozważań wnioskujemy, że po stanie zanikającym USSP wyrażającej CSP - osiągany jest stan uchwytny (nie może być osiągnięty kolejny stan zanikający).

Dalsza część rozważań, którą można oprzeć na analizie systemów procesów sekwencyjnych z wzajemnym wykluczaniem, zostanie pominięta.

4.4. Podsumowanie

Przedstawimy tylko te wnioski, które nie są analogiczne do wniosków dla systemów procesów z wzajemnym wykluczaniem.



Rys. 4.14

Podsumujmy wyniki uzyskane dla przypadku deterministycznego.

Dla zachowania przejściowego, interesująca nas granica złożoności obliczeniowej znajduje się między CSP^*, opisanymi drogami prostymi dla dowolnej liczby procesów, a CSP^* wyrażonymi automatami dla dwu procesów.

W przypadku zachowania cyklicznego badana granica złożoności obliczeniowej jest usytuowana między CSP^*, wyrażonymi obwodami dla dowolnej liczby procesów, a CSP^* reprezentowanymi automatami dla dwu procesów.

Przez iterację wewnętrzną rozumiemy każdą iterację z wyłączeniem iteracji obejmującej cały proces cykliczny.

Własnością krytyczną dla złożoności obliczeniowej jest istnienie w procesie sekwencyjnym iteracji wewnętrznej.

Wartości minimalnego czasu wykonania i minimalnego czasu cyklu mogą być wykorzystane jako dolne oszacowania średniego czasu wykonania i średniego czasu cyklu dla przypadku mieszanego.

Dla procesów bez rozgałęzień przepływu sterowania (procesy opisane drogami prostymi lub obwodami) przypadki deterministyczny i mieszany są identyczne. Dla łańcucha Markowa z K stanami, złożoność obliczeniowa wyznaczania macierzy odwrotnej I-W o wymiarach K×K dla zachowa-

nia przejściowego jest równa $O(K^3)$. Złożoność rozwiązania układu M równań liniowych dla zachowania cyklicznego jest taka sama. Dla przypadku deterministycznego bez rozgałęzień, złożoność obliczeniowa OPCW (OPCC) jest równa O(m), gdzie $m = |P|$ ($O(n^2 r)$, przy czym $n = |T|$, natomiast r jest liczbą procesów). Ponieważ zwykle prawdziwa jest relacja $M \gg n,m,r$, a więc dla procesów bez rozgałęzień, metody wyznaczania rozwiązań przypadku deterministycznego są znacznie efektywniejsze obliczeniowo niż metody przypadku mieszanego.

Dla takich stochastycznych modeli CSP^*, do których nie można zastosować teorii łańcuchów Markowa - problemy analizy są bardziej złożone. W takiej sytuacji należy rozważać relacje, opisujące sumę zmiennych losowych ze względu na sekwencyjne wykonywanie instrukcji pojedynczego procesu oraz relacje wyrażające maksimum zmiennych losowych z racji komunikacji między procesami. Problemy sumy i maksimum zmiennych losowych badane są w analizie stochastycznych sieci PERT. Zagadnienia maksimum są znacznie trudniejsze niż zagadnienia sumy. W celu otrzymania oszacowań wartości średniej i wariancji czasu ukończenia dla stochastycznej sieci PERT, można wykorzystać oszacowania sumy i maksimum zmiennych losowych z prac [10], [33]. Sieci PERT są sieciami acyklicznymi. Zatem rezultaty uzyskane dla tych sieci mogą być użyte w analizie CSP^* bez instrukcji iteracji. Jednakże dla większych CSP^* oszacowania czasu wykonania mogą być zbyt słabe ze względu na znaczną liczbę oszacowań pośrednich.

5. OCENA WYDAJNOŚCI SYSTEMÓW PROCESÓW SEKWENCYJNYCH KOMUNIKUJĄCYCH SIĘ POPRZEZ BUFORY

Komunikacja synchroniczna stosowana m.in. w CCS i CSP nie wyczerpuje trybów wymiany komunikatów w środowisku rozproszonym. Dużą rangę ma komunikacja asynchroniczna (poprzez bufory o nieograniczonej pojemności) oraz częściowo synchronizowana (poprzez bufory o ograniczonej pojemności). Proces nadawczy wysyła komunikat do bufora, natomiast proces odbiorczy pobiera komunikat.

Proponowane podejścia w zakresie komunikacji poprzez bufory są oparte m.in. na koncepcji procesów rozproszonych Brinch Hansena [21] i sieci procesów Kahna [56]. Rezultaty pracy Hansena wykorzystano w projekcie języka Ada. Prace [60], [116] zawierają zastosowania komunikacji poprzez bufory.

Zakładamy, że przesyłany komunikat nie jest tracony ani deformowany. Zatem ze względu na komunikację bez błędów nie jest wymagany mechanizm time-out'u. Ponadto komunikat może być pobrany z bufora natychmiast po przesłaniu go do bufora.

Komunikacja poprzez bufory jest stosowana również w środowisku zwartym. W systemie operacyjnym systemu czasu rzeczywistego, omawianym w punkcie 1.1, bufory w postaci dysków wirtualnych są wykorzystywane do komunikacji między następującymi cyklicznymi procesami sekwencyjnymi:

- procesem wejściowym a procesem przetwarzania,
- procesem przetwarzania a procesem wyjściowym.

W rozdziale tym badamy cykliczne procesy sekwencyjne. Punkt pierwszy zawiera definicje podstawowych pojęć z zakresu komunikacji poprzez bufory między procesami sekwencyjnymi - wyrażone w języku sieci Petriego. W punkcie drugim jest opisany przypadek deterministyczny. Ze względu na podobieństwo przypadku stochastycznego do przypadku stochastycznego dla CSP, przypadek ten nie będzie analizowany. Rozdział zamknięty jest podsumowaniem.

5.1. Reprezentacja sieciowa systemów procesów komunikujących się poprzez bufory

Przedstawione niżej definicje są oparte na pracy [106].

Proces sekwencyjny komunikujący się poprzez bufory jest piątką $A = < S,B,T,F,M_0 >$, przy czym:

- S jest zbiorem stanów,
- B jest zbiorem buforów,
- $\mathcal{N}(A) = <<S \cup B,T,F>,M_0>$ jest ZSP procesu A, przy czym $S \cup B$ - zbiór miejsc, T - zbiór przejść, F - zbiór łuków o krotności równej jedności, M_0 - znakowanie początkowe,
- Obcięcie ZSP $\mathcal{N}(A)$ do zbioru stanów S określone jako ZSP $S(A) = <<S,T,F \cap (S \cup T)^2>,M_0|S>$ jest procesem sekwencyjnym procesu A, którego SP $\mathcal{S} = < S,T,F \cap (S \cup T)^2 >$ jest silnie spójna.

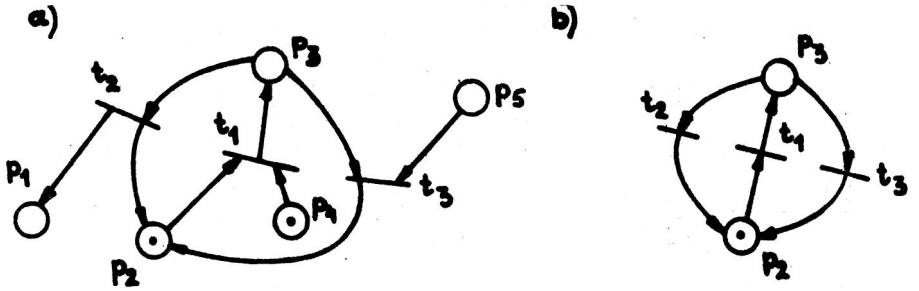
SP $\mathcal{S}$ jest silnie spójna, ponieważ rozpatrujemy cykliczne procesy sekwencyjne.

Przykład procesu sekwencyjnego A komunikującego się poprzez bufory ilustruje rys. 5.1a.

Miejsca p_1,p_4,p_5 są buforami. Łuk skierowany z przejścia t_i do bufora p_k oznacza, że akcja wyrażona przejściem t_i wysyła komunikat do bufora p_k. Łuk skierowany z bufora p_i do przejścia t_j wskazuje, że dla wykonania akcji opisanej przejściem t_j niezbędny jest komunikat pobierany z bufora p_i. Rys. 5.1b zawiera obcięcie ZSP $\mathcal{N}(A)$ procesu A z rys. 5.1a do zbioru stanów.

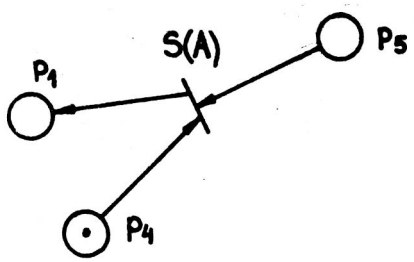
Dla procesu sekwencyjnego komunikującego się poprzez bufory $A = < S,B,T,F,M_0 >$ definiujemy strukturę zgrubną procesu A jako ZSP $C(A) = <<B,\{S(A)\},F'>,M_0'>$ (tzn. jedynym elementem zbioru przejść jest ZSP S(A)) spełniającą warunek

$\forall b \in B((bF'S(A) \iff \exists t \in T \; bFt) \wedge (S(A)F'b \iff \exists t \in T \; tFb) \wedge M_0'(b) = M_0(b))$.

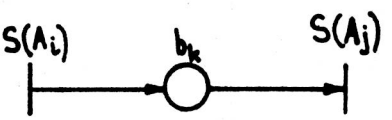


Rys. 5.1

Strukturę zgrubną procesu A z rys. 5.1a przedstawia rys. 5.2.



Rys. 5.2



Rys. 5.3

Niech $\mathcal{N} = \ll P,T,F>,M_0>$, $\mathcal{N}' = \ll P',T',F'>,M_0'>$ będą dwoma ZSP$^{\#}$. Niech $P \cap T' = T \cap P' = \emptyset$ oraz niech $\forall p \in P \cap P' \; M_0(p) = M_0'(p)$. Sumą ZSP$^{\#}$ $\mathcal{N}, \mathcal{N}'$ jest sieć

$$\mathcal{N} \cup \mathcal{N}' = \ll P \cup P', T \cup T', F \cup F' >, M_0 \cup M_0' >.$$

System r procesów sekwencyjnych komunikujących się poprzez bufory $A_i = <S_i,B_i,T_i,F_i,M_{0i}>$ zdefiniowany jest jako ZSP $\mathcal{N} = \bigcup_{i=1}^{r} \mathcal{N}(A_i)$ pod warunkiem, że

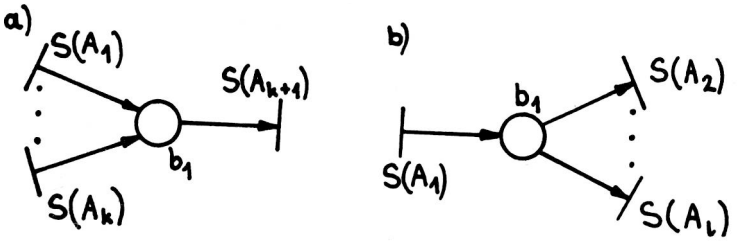
$$\forall i,j \in \{1,\ldots,r\} \; (i \neq j \implies S_i \cap S_j = T_i \cap T_j = \emptyset).$$

System procesów komunikujących się poprzez bufory oznaczać będziemy skrótem SPKB.

Struktura zgrubna SPKB $\mathcal{N}$ oznaczona symbolem $C(\mathcal{N})$ określona jest równością $C(\mathcal{N}) = \bigcup_{i=1}^{r} C(A_i)$.

Jeśli w strukturze zgrubnej SPKB istnieje fragment reprezentowany rys. 5.3, to znaczy, że proces $S(A_1)$ nadaje komunikaty do bufora b_k, natomiast proces $S(A_j)$ z tego bufora komunikaty pobiera.

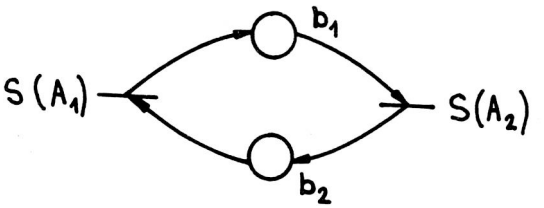
Obecnie podamy przykłady struktur zgrubnych.



Rys. 5.4

Struktura zgrubna z rys. 5.4a obrazuje współpracę poprzez bufor - wielu procesów nadawczych i jednego odbiorczego, natomiast struktura z rys. 5.4b - współpracę jednego procesu nadawczego z wieloma procesami odbiorczymi.

Przykład, bardziej złożonej niż powyższe, struktury zgrubnej ilustruje rys. 5.6. W strukturze tej nie ma obwodów; natomiast obwód zawiera struktura zgrubna z rys. 5.5.



Rys. 5.5

Rozważamy deterministyczną i niedeterministyczną komunikację między procesami. W przypadku komunikacji deterministycznej, wymagamy, aby wynik obliczeń był jednoznaczny i nie zależał od wzajemnej szybkości obliczeń współbieżnych. Stąd każdy bufor ma co najwyżej jeden proces źródłowy i co najwyżej jeden proces przeznaczenia (w przeciwnym razie względne szybkości dwu procesów mogłyby wpływać na przydział ostatniego komunikatu z dzielonego bufora wejściowego lub na przydział ostatniego wolnego miejsca w buforze wyjściowym). Dodatkowo wymagamy, aby dla stanów z rozgałęzieniem przepływu sterowania (z miejsca będącego

stanem skierowany jest więcej niż jeden łuk), wybór kierunku przepływu sterowania nie zależał od bieżącej liczby komunikatów w buforach, ponieważ liczba ta może być wynikiem wzajemnych szybkości przebiegu procesów źródłowych i przeznaczenia. Zatem wymagamy, aby ZSP $\mathcal{M}(A)$ procesu A była siecią swobodnego wyboru.

Zakładamy, że SPKB jest żywy tzn. ZSP $\mathcal{M} = \bigcup_{i=1}^{r} \mathcal{M}(A_i)$ jest żywa. Problem żywotności dla komunikacji deterministycznej badany jest w pracach [14], [106]. Dla komunikacji niedeterministycznej możemy wykorzystać rozwiązania problemu żywotności dla sieci Petriego.

5.2. Przypadek deterministyczny

W pierwszej kolejności rozważać będziemy komunikację poprzez bufory bez ograniczeń na pojemność buforów, a następnie komunikację poprzez bufory o ograniczonej pojemności.

5.2.1. Komunikacja poprzez bufory bez ograniczeń ich pojemności

Rozważamy SPKB*, których struktura zgrubna nie zawiera obwodów oraz SPKB* z obwodami w ich strukturze zgrubnej.

Najpierw przeanalizujemy OPCC dla SPKB* bez obwodów w ich strukturze zgrubnej.

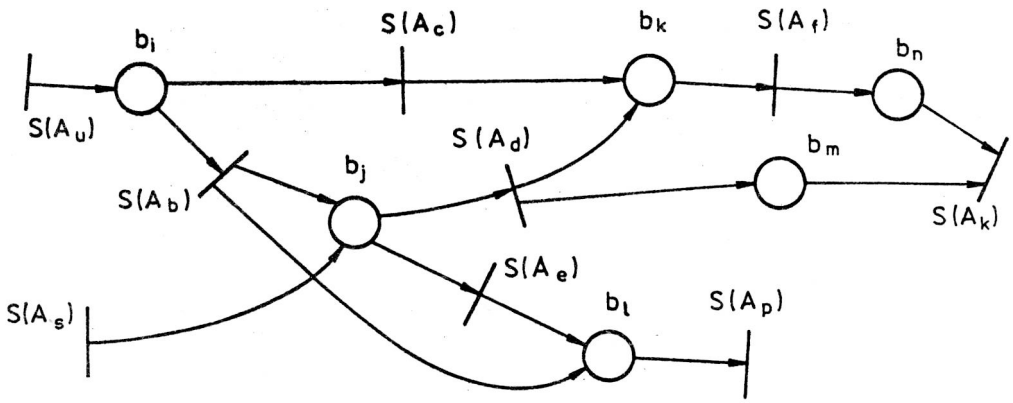
W pracy [104] zawarty jest rezultat dotyczący współpracy między wieloma procesami nadającymi a jednym odbierającym dla podklasy procesów sekwencyjnych komunikujących się poprzez bufory. Podklasa ta zawiera takie procesy A_i, których proces sekwencyjny S(A) (zdefiniowany w punkcie 5.1) jest reprezentowany obwodem z jedną kropką. Dla tego przypadku w pracy [104] zawarty jest algorytm wielomianowy.

Udowodnimy, że dla SPKB* bez obwodów w strukturze zgrubnej istnieje algorytm wielomianowy dla OPCC. Rozważamy OPCC dla SPKB i danego T-niezmiennika I. Z własności T-niezmiennika wynika, że po sekwencji palenia σ, której wektor palenia równy jest T-niezmiennikowi I, liczby znaczników w buforach są takie same jak przed sekwencją σ. Dla procesu sekwencyjnego komunikującego się poprzez bufory $A_i = \langle S_i, B_i, T_i, F_i, M_{0i} \rangle$ z T-niezmiennika wybieramy te składowe $I(t_j)$, które dotyczą przejść $t_j \in T_i$. Te składowe tworzą T-niezmiennik I_i dla procesu A_i. Niech $\gamma_{min}(A_i)$ oznacza minimalny czas cyklu procesu sekwencyjnego A_i komunikującego się poprzez bufory dla T-niezmiennika I_i. Wartość $\gamma_{min}(A_i)$ równa jest minimalnemu czasowi cyklu procesu sekwencyjnego $S(A_i)$ procesu A_i dla T-niezmiennika I_i (wyznaczonemu na podstawie wyników uzyskanych dla DSP*, których SP jest automatem). Minimalny czas cyklu $\gamma(I, x^{*})$ dla SPKB i T-niezmiennika I musi spełniać nierówność:

$$\gamma(I,x^*) \geq LB = \max\left\{\gamma_{min}(A_1),\ldots,\gamma_{min}(A_r)\right\}.$$

Dowiedziemy, że dolne ograniczenie LB może być osiągnięte dla odpowiedniego wyboru chwil, w których rozpoczynane jest palenie przejść procesów sekwencyjnych.

W strukturze zgrubnej SPKB - przejście $S(A_i)$ reprezentuje proces sekwencyjny procesu A_i komunikującego się poprzez bufory. W strukturze tej wybierzmy te przejścia $S(A_k)$, do których nie jest skierowany żaden łuk. Zbiór tych przejść oznaczmy symbolem Θ. Dla struktury zgrubnej z rys. 5.6 zbiór $\Theta = \left\{S(A_u), S(A_s)\right\}$.



Rys. 5.6

Dla każdego przejścia $S(A_j) \notin \Theta$ szukamy takiej drogi skierowanej z dowolnego $S(A_k) \in \Theta$ do przejścia $S(A_j)$, która zawiera największą liczbę przejść $S(A_p)$. Dla przejścia $S(A_j)$ wymieniona liczba przejść oznaczona jest symbolem l_j. Przejściom $S(A_k) \in \Theta$ przypisujemy liczbę $l_k = 0$. Jako przykład rozpatrzmy strukturę zgrubną z rys. 5.6. Wartości liczb l_j są następujące: $l_u = l_s = 0$, $l_c = l_b = 1$, $l_d = l_e = 2$, $l_f = l_p = 3$, $l_k = 4$.

Symbolem $I(b_j)$ oznaczmy zbiór procesów sekwencyjnych A_i przesyłających komunikaty do bufora b_j, natomiast symbolem $O(b_j)$ - zbiór procesów sekwencyjnych pobierających komunikaty z bufora b_j. Dla procesów $A_d \in I(b_j)$ oraz $A_f \in O(b_j)$ prawdziwa jest nierówność $l_d < l_f$. Wymagamy, aby palenie przejść procesu $A_f \in O(b_j)$ w k-tym czasie cyklu procesu A_f rozpoczynane było po zakończeniu palenia przejść procesu $A_d \in I(b_j)$ w k-tym czasie cyklu procesu A_d.

Dla procesu sekwencyjnego A_j komunikującego się poprzez bufory, niech przejście $t_p \in T_j$ będzie przejściem, którego palenie rozpoczynane jest na początku czasu cyklu procesu A_j. Niech dla procesu A_j -

chwila $\tau_j(k)$ ($k \in \{1,2,\ldots\}$) będzie chwilą, w której rozpoczynane jest palenie przejścia t_p po raz $I(t_p)(k-1) + 1$. Chwila $\tau_j(k)$ zadana jest równością

$$\tau_j(k) = l_j \text{ LB} + (k-1) \text{ LB} = (l_j + k-1) \text{ LB},$$

gdzie l_j LB opisuje chwilę, w której rozpoczynane jest palenie przejścia t_p po raz pierwszy. Palenie wszystkich przejść procesu A_j następuje w przedziale $[\tau_j(k), \tau_j(k+1)[$. Z powyższego rozumowania wnioskujemy, że $\gamma(I,x^*) = \text{LB}$.

Wyznaczeniu wartości $\gamma(I,x^*)$ służy algorytm 5.1.

Algorytm 5.1

1. Dla każdego procesu sekwencyjnego A_j komunikującego się poprzez bufory ($j \in \{1,\ldots,r\}$) obliczyć

$$\gamma_{min}(A_j) = \sum_{t_k \in T_j} I(t_k)\,\mathcal{T}(t_k),$$

gdzie T_j - zbiór przejść procesu A_j.

2. Wyznaczyć

$$\gamma(I,x^*) = \max\left\{\gamma_{min}(A_1),\ldots,\gamma_{min}(A_r)\right\}.$$

Złożoność obliczeniowa kroku 1 równa jest $O(n)$, gdzie n jest sumaryczną liczbą przejść w procesach $A_1,\ldots,A_r$. Liczba n spełnia równanie $n = \sum_{i=1}^{r} |T_i|$, gdzie $|T_i|$ jest liczbą przejść procesu A_i. Złożoność obliczeniowa kroku 2 wynosi $O(r)$. Ponieważ $r < n$, a więc złożoność algorytmu równa jest $O(n)$.

Twierdzenie 5.1 [77]

Dla SPKB bez obwodów w strukturze zgrubnej, OPCC może być rozwiązany za pomocą algorytmu wielomianowego o złożoności obliczeniowej $O(n)$, gdzie n jest liczbą przejść SPKB. ∎

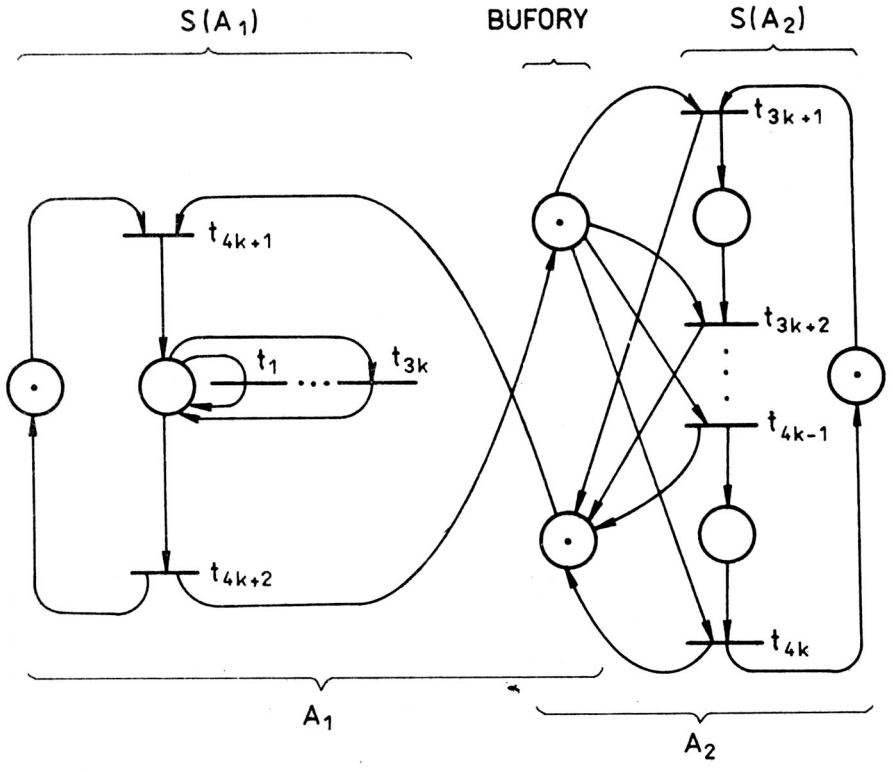
W przypadku SPKB z obwodami w strukturze zgrubnej, DPCC dla dwu procesów jest silnie NP-trudny - jak zostanie to dalej udowodnione. Zatem DPCC jest silnie NP-trudny również dla większej liczby procesów.

W celu udowodnienia, że DPCC jest silnie NP-trudny dla dwu procesów wykorzystamy ZSP z rys. 5.7.

W definicji pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU do DPCC dla powyższej sieci zakładamy:

1. Czasy palenia przejść spełniają równości:

$$\mathcal{T}(t_i) = x_i \text{ dla } i \in \{1,\ldots,3k\}, \quad \mathcal{T}(t_{3k+j}) = B \text{ dla } j \in \{1,\ldots,k\},$$
$$\mathcal{T}(t_{4k+1}) = \mathcal{T}(t_{4k+2}) = 0,$$



Rys. 5.7, A_i - proces sekwencyjny komunikujący się poprzez bufory, $S(A_i)$ - proces sekwencyjny
Fig. 5.7, A_i - sequential process communicating by buffers, $S(A_i)$ - sequential process

2. T-niezmiennik I spełnia wymagania:

$I(t_{4k+1}) = I(t_{4k+2}) = k$ oraz $I(t) = 1$

dla pozostałych przejść,

3. Wartość funkcji kryterialnej: $\gamma = k$ B.

Dowód podobny do dowodu tw. 2.12 pomijamy.

Twierdzenie 5.2

Dla SPKB z obwodami w strukturze zgrubnej, DPCC jest silnie NP--trudny dla dwu i większej liczby procesów. ∎

Twierdzenie słabsze, wg którego DPCC dla powyższego przypadku jest NP-trudny, zawarto w pracy [77].

5.2.2. Komunikacja poprzez bufory o ograniczonej pojemności

Rozważamy bufory o pojemności równej jedności. Zakładamy, że komunikat w buforze wejściowym akcji, której przyporządkowane jest przejście t, znajduje się przez czas trwania tej akcji $\mathcal{T}(t)$. Powodowani tym założeniem, przyjmujemy inną interpretację procesu palenia przejść niż w punkcie 1.2.1.

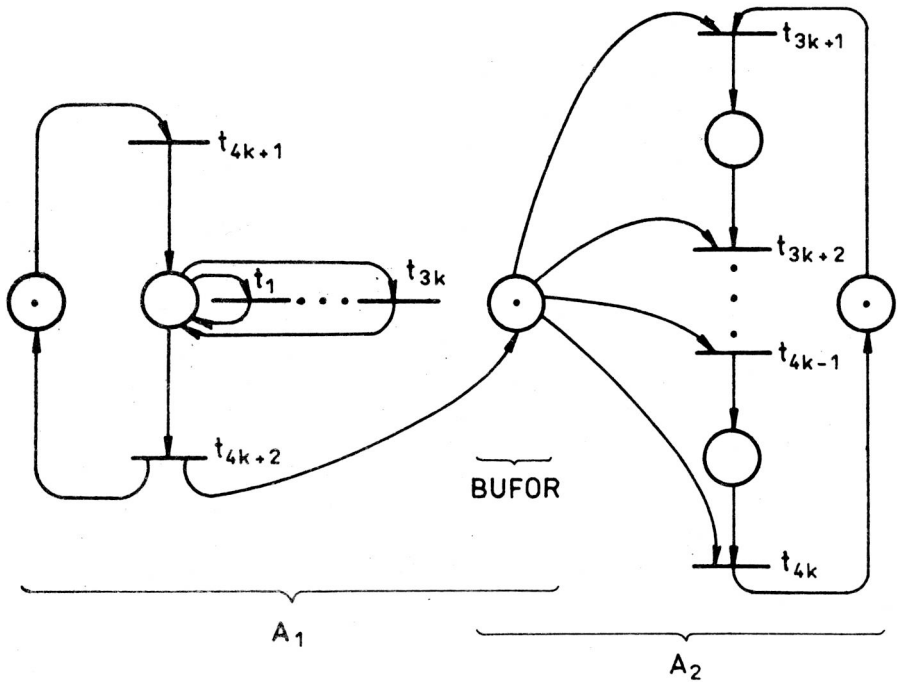
Niech palenie przejścia t_j rozpoczyna się w chwili τ'. Znaczniki w miejscach wejściowych p_i przejścia t_j - niezbędne do palenia tego przejścia są rezerwowane przez czas palenia $\mathcal{T}(t_j)$ - pozostając w miejscach $p_i \in {}^\bullet t_j$. Jeśli w chwili $\tau'' = \tau' + \mathcal{T}(t_j)$ nie ma znaczników w buforach $b_t \in t_j^\bullet$, to zarezerwowany znacznik jest usuwany z każdego miejsca $p_i \in {}^\bullet t_j$ oraz jeden znacznik jest dodawany do każdego miejsca $p_k \in t_j^\bullet$. Jednak jeśli w buforze $b_u \in t_j^\bullet$ znajduje się znacznik, to rezerwowany znacznik jest usuwany z każdego miejsca $p_i \in {}^\bullet t_j$, natomiast znakowanie miejsc $p_k \in t_j^\bullet$ nie jest zmieniane. Niech chwila $\tau''' > \tau''$ będzie najwcześniejszą chwilą, w której nie ma znaczników w buforach $b_t \in t_j^\bullet$. W tym przypadku w chwili τ''', jeden znacznik jest dodawany do każdego miejsca $p_k \in t_j^\bullet$.

W pierwszej kolejności przeanalizujemy DPCC dla SPKB bez obwodów w strukturze zgrubnej. W tym przypadku, w celu udowodnienia silnej NP--trudności DPCC dla dwu procesów wykorzystamy ZSP z rys. 5.8.

Dowód silnej NP-trudności DPCC dla powyższej sieci, oparty na pseudowielomianowej transformacji Turinga PROBLEMU TRÓJPODZIAŁU do DPCC, jest podobny do dowodu tw. 5.2.

Twierdzenie 5.3

DPCC dla SPKB o pojemności buforów równej jedności oraz bez obwodów w strukturze zgrubnej jest silnie NP-trudny dla dwu i więcej procesów. ∎



Rys. 5.8

ZSP z rys. 5.7 może być wykorzystana w celu udowodnienia silnej NP-trudności DPCC dla SPKB z pojemnością buforów równą jedności. Otrzymujemy zatem następujące twierdzenie.

Twierdzenie 5.4

DPCC dla SPKB o pojemności buforów równej jedności oraz z obwodami w strukturze zgrubnej jest silnie NP-trudny dla dwu i więcej procesów. ∎

5.3. Podsumowanie

Przedstawimy tylko te wnioski, które nie są analogiczne do wniosków dla systemów procesów z wzajemnym wykluczaniem.

Podsumujmy rezultaty uzyskane dla przypadku deterministycznego.

Dla komunikacji poprzez bufory bez ograniczeń na ich pojemność - własnością krytyczną dla złożoności obliczeniowej PCC jest istnienie obwodów w strukturze zgrubnej SPKB. Interesująca nas granica złożoności obliczeniowej znajduje się między $SPKB^{*}$ bez obwodów w strukturze zgrubnej dla dowolnej liczby procesów, a $SPKB^{*}$ z obwodami w strukturze zgrubnej dla zaledwie dwu procesów.

W sytuacji komunikacji z ograniczeniem pojemności buforów - PCC jest bardziej złożony z racji tego ograniczenia. DPCC jest silnie NP--trudny dla SPKB* bez obwodów w strukturze zgrubnej dla zaledwie dwu procesów.

Efektywny algorytm 5.1 może być wykorzystany do komunikacji deterministycznej i niedeterministycznej (określenia obu typów komunikacji znajdują się w punkcie 5.1). Sieci z rys. 5.7, 5.8 opisują komunikację deterministyczną. Stąd dla przypadków SPKB* opisanych twierdzeniami 5.2, 5.3, 5.4, DPCC jest silnie NP-trudny dla komunikacji deterministycznej. W podobny sposób możemy dowieść silnej NP-trudności DPCC dla komunikacji niedeterministycznej w przypadkach opisanych powyższymi trzema twierdzeniami.

W przypadku komunikacji poprzez kanały FIFO (ten rodzaj komunikacji istnieje w takich językach, jak CHILL, SDL), sieci FIFO [84] są adekwatniejszym formalizmem niż sieci rozważane w tym rozdziale. W sieciach FIFO istnieje szczególny typ buforów, a mianowicie kolejki FIFO. Dla sieci tych można przeprowadzić podobną analizę - opierając się na metodyce z tego rozdziału.

6. ZAKOŃCZENIE

Zestawienie wyników autora w zakresie złożoności obliczeniowej problemu czasu cyklu dla ważniejszych klas deterministycznych sieci Petriego zawiera tabela 1.

Praca zawiera wyniki autora rozwijające metody oceny wydajności wszystkich zasadniczych mechanizmów stosowanych do współpracy między procesami sekwencyjnymi (tabele 2,3,4).

Dla sieci z czasem przypisanym miejscom (omówionych w punkcie 2.4.1) możemy przeprowadzić rozważania analogiczne do zawartych w pracy.

Wyniki pracy wskazują na dużą złożoność obliczeniową zagadnień oceny wydajności systemów współbieżnych z wykorzystaniem sieci Petriego, która jest rezultatem dużej złożoności tych systemów. W celu pokonania tego ograniczenia, niezbędna jest dekompozycja sieci lub przynajmniej zbioru znakowań osiągalnych ze znakowania początkowego. Szczególnie ważna jest dekompozycja hierarchiczna z racji m.in. wielopoziomowej budowy strukturalizowanych systemów operacyjnych, warstwowej struktury sieci komputerowych i hierarchizacji architektury komputerów. Zatem nabierają znaczenia opisy typu we-wy dla sieci Petriego z czasem. Opisy takie powinny charakteryzować czasy, po jakich występują poszczególne zdarzenia wyjściowe podsieci (pojawianie się kropek w miejscach wyjś-

Tabela 1

Złożoność obliczeniowa problemu czasu cyklu dla wybranych klas deterministycznych sieci Petriego

Własności wybranych klas sieci Petriego		Wynik
Charakterystyka silnie spójnych SP^*		Twierdzenie 2.3
Zależności między SP^* bez konfliktów w przód (w tył) a grafami synchronizacji		Twierdzenie 2.4 (2.5)
Zależności między SP^* bez współbieżności w przód (w tył) a automatami		Twierdzenie 2.6 (2.7)
Zależności między bezkonfliktowymi SP^* bez pętli a grafami synchronizacji		Twierdzenie 2.13
Rozstrzygalność DPCC i OPCC dla DSP^*		Twierdzenie 2.8
Złożoność obliczeniowa problemu czasu cyklu		
Klasa sieci	Złożoność	Wynik
Grafy synchronizacji	$O(n^{3,5}L)$	Twierdzenie 2.10
Sieci swobodnego wyboru	Silnie NP-trudny	Twierdzenie 2.12
Sieci bezkonfliktowe bez pętli	$O(n^{3,5}L)$	Wniosek 2.2
Sieci bezkonfliktowe	Silnie NP-trudny	Twierdzenie 2.14
Sieci trwałe	Silnie NP-trudny	Wniosek 2.3
Sieci nie P-niezmiennicze	Silnie NP-trudny	Twierdzenie 2.15
NP-zupełność DPCC		Twierdzenie 2.16
Oszacowania minimalnego czasu cyklu		
Sieci P-niezmiennicze		Twierdzenie 2.17
Sieci nie P-niezmiennicze		Twierdzenie 2.18

m - liczba miejsc sieci, n - liczba przejść sieci,
L - liczba zależna od długości danych wejściowych

ciowych, palenie przejść wyjściowych) w następstwie zdarzeń wejściowych (pojawianie się kropek w miejscach wejściowych, palenie przejść wejściowych) pod warunkiem określonego znakowania miejsc wewnętrznych podsieci. Modele typu we-wy dla sieci otrzymujemy w rezultacie abstrahowania od szczegółów działania sieci nie objawiających się bezpośrednio na zewnątrz. Przykład opisu typu we-wy dla deterministycznych roz-

Tabela 2

Ocena wydajności systemów procesów sekwencyjnych z wzajemnym wykluczaniem

Złożoność obliczeniowa problemu czasu cyklu (przypadek deterministyczny)				
Problem			Złożoność	Wynik
Model procesu	Liczba dostępów procesu do zasobu w czasie cyklu systemu	Liczba procesów		
Obwód	1	r	O(r)	Twierdzenie 3.1
Obwód	1 dla r-1 procesów	r	Silnie NP-trudny	Twierdzenie 3.2
Automat	1	r	O(r)	Twierdzenie 3.3
Automat	Bez ograniczeń	2	Silnie NP-trudny	Twierdzenie 3.4
Wyznaczanie macierzy W prawdopodobieństw przejść (przypadek stochastyczny)				
Porównanie złożoności obliczeniowej algorytmu 3.1 z algorytmem macierzowym z pracy [4]				Wniosek 3.1

szerzonych grafów znakowanych zawiera praca [71]. Podobne wyniki można uzyskać dla deterministycznych sieci Petriego - korzystając z rezultatów z zakresu abstrahowania zawartych w pracach [8],[72],[119].

Proces abstrahowania dla sieci bez ograniczeń nałożonych na ich postać jest złożony obliczeniowo. Uproszczenie oceny wydajności można uzyskać poprzez założenia strukturalizacji sieci. Ponieważ strukturalizacja oprogramowania jest naturalną metodą upraszczania procesu jego budowy, a więc warto skorzystać z programowania strukturalnego [67] w formułowaniu metodyki oceny wydajności oprogramowania z zastosowaniem sieci Petriego. Przykład zastosowania sieci Petriego w hierarchicznej ocenie wydajności oprogramowania współbieżnego opisuje praca [53].

Złożoność obliczeniową zmniejszają, ale zwykle nieznacznie, transformacje sieci zachowujące charakterystyki czasowe [128].

Obecnie przedstawimy, specyficzne dla przypadku deterministycznego kierunki dalszych badań.

Badania przedstawione w pracy dotyczą czasowej złożoności obliczeniowej problemu czasu cyklu i czasu wykonania.

Tabela 3

Ocena wydajności komunikujących się procesów sekwencyjnych

Złożoność obliczeniowa problemu czasu wykonania (przypadek deterministyczny)			
Problem		Złożoność	Wynik
Model procesu	Liczba procesów		
Droga prosta	r	O(m)	Twierdzenie 4.1
Automat	2	Silnie NP-trudny	Twierdzenie 4.2
Złożoność obliczeniowa problemu czasu cyklu (przypadek deterministyczny)			
Problem		Złożoność	Wynik
Model procesu	Liczba procesów		
Obwód	r	$O(n^2 r)$	Twierdzenie 4.3
Automat	2	Silnie NP-trudny	Twierdzenie 4.4

m - liczba miejsc ZSP reprezentującej CSP,
n - liczba przejść ZSP reprezentującej CSP

Tabela 4

Ocena wydajności systemów procesów sekwencyjnych komunikujących się poprzez bufory

Złożoność obliczeniowa problemu czasu cyklu (przypadek deterministyczny)					
Problem				Złożoność	Wynik
Model procesu	Pojemność buforów	Struktura zgrubna	Liczba procesów		
Automat	Bez ograniczeń	Bez obwodów	r	O(n)	Twierdzenie 5.1
Automat	Bez ograniczeń	Z obwodami	2	Silnie NP-trudny	Twierdzenie 5.2
Automat	Równa jedności	Bez obwodów	2	Silnie NP-trudny	Twierdzenie 5.3
Automat	Równa jedności	Z obwodami	2	Silnie NP-trudny	Twierdzenie 5.4

n - liczba przejść SPKB

Ponieważ wiele decyzyjnych problemów czasu cyklu jest NP-trudnych, a więc dalszą charakteryzację tych problemów możemy uzyskać poprzez badanie ich pamięciowej złożoności obliczeniowej [39].

Dla NP-trudnych problemów czasu cyklu lub czasu wykonania możemy budować algorytmy oparte na metodzie podziałów i ograniczeń, a w szczególności możemy formułować takie algorytmy pod kątem hierarchicznej oceny wydajności.

W konstruowaniu algorytmów opartych na metodzie podziałów i ograniczeń możemy skorzystać z rezultatów otrzymanych w pracy.

Metodę przeglądu przestrzeni rozwiązań możemy oprzeć na rozumowaniu zawartym w dowodzie tw. 2.8 o rozstrzygalności problemu czasu cyklu.

W wyznaczaniu dolnego oszacowania minimalnego czasu cyklu możemy posłużyć się oszacowaniami o wielomianowej czasowej złożoności obliczeniowej - przedstawionymi w punkcie 2.5.

W przeglądzie przestrzeni rozwiązań, opartym na dowodzie tw. 2.8, dla poszczególnych rozwiązań są budowane bezpieczne grafy znakowane. Algorytm o najmniejszej złożoności obliczeniowej dla optymalizacyjnego problemu czasu cyklu dla bezpiecznych grafów znakowanych jest zawarty w pracy [36]. Algorytm ten można zastosować do określania górnych oszacowań.

Jeśli algorytm, oparty na metodzie podziałów i ograniczeń, jest zbyt złożony obliczeniowo dla danego zastosowania, to przybliżone rozwiązanie możemy otrzymać za pomocą oszacowań minimalnego czasu cyklu (punkt 2.5) lub ich odpowiedników dla minimalnego czasu wykonania.

Badań wymaga ocena jakości tych oszacowań i ich poprawa.

Obecnie omówimy, wynikające z uzyskanych w pracy rezultatów sugestie konstrukcji algorytmów heurystycznych do oceny wydajności systemów procesów sekwencyjnych.

W punkcie 2.1 pracy przedstawiliśmy przykłady sieci P-niezmienniczych. W zakresie systemów procesów sekwencyjnych sieciami P-niezmienniczymi są podsieci opisujące poszczególne procesy. Ponadto mechanizmy współpracy procesów sekwencyjnych są często wyrażane podsieciami P-niezmienniczymi. Przykładami takich podsieci są: podsieć opisująca zarządzanie zasobem (rys. 2.1), podsieć ilustrująca komunikację między procesami z potwierdzaniem odbioru komunikatu (rys. 2.2), podsieć reprezentująca komunikację między procesami poprzez bufory (rys. 5.7).

W celu otrzymania dolnego oszacowania czasu cyklu, dla każdej z wyodrębnionych w sieci wyrażającej system procesów podsieci P-niezmienniczej możemy wyznaczyć dolne oszacowanie czasu cyklu i spośród uzyskanych wybrać największe. Dolne oszacowanie możemy również określić w sposób opisany poniżej. W sieci modelującej system procesów wyróżniamy tę jej podsieć (zawierającą np. wybrane procesy i wybrane mechanizmy

współpracy), dla której istnieje algorytm wielomianowy dla optymalizacyjnego problemu czasu cyklu.

Aby uzyskać górne oszacowanie minimalnego czasu cyklu, możemy wybrać dowolną sekwencję palenia o wektorze palenia równym T-niezmiennikowi określającemu tryb pracy systemu i dla niej obliczyć minimalny czas cyklu.

W przypadku systemów wyrażonych uogólnionymi stochastycznymi sieciami Petriego, zasadnicza trudność tkwi w braku satysfakcjonujących metod analizy hierarchicznej łańcuchów Markowa. Ogólne metody dekompozycji dokładnej łańcuchów Markowa [62] nie mają większego znaczenia praktycznego z racji wymaganych przez te metody założeń. Rangi nabierają zatem metody przybliżone. Dużym wkładem w zakresie dekompozycji przybliżonej łańcuchów Markowa są wyniki dla systemów prawie całkowicie dekomponowalnych [29], [30]. Dla tych systemów uzyskano satysfakcjonujące rezultaty przy bardzo małych oddziaływaniach między stanami z różnych grup stanów zagregowanych względem oddziaływań między stanami tej samej grupy. Niestety w większości systemów współbieżnych informatyki względne oddziaływania między stanami z różnych grup nie są tak małe.

Wydaje się, że trudno będzie uzyskać ogólne metody dekompozycji łańcuchów Markowa przydatne w praktyce. Sensownym kierunkiem badawczym jest szukanie rozwiązań dokładnych i przybliżonych dla wybranych klas systemów rzeczywistych. Na przykład istotnym osiągnięciem jest dokładne rozwiązanie równań stanu stacjonarnego dla paru wieloprocesorowych systemów magistralowych [3].

Obiecującym kierunkiem poszukiwań rozwiązań przybliżonych jest łączenie uogólnionych stochastycznych sieci Petriego (USSP*) z sieciami kolejkowymi i sieciami PERT oraz GERT [35]. Złożoność obliczeniowa metod rozwiązywania sieci kolejkowych i sieci PERT jest znacznie mniejsza niż metod dla USSP*. Z drugiej strony, USSP* umożliwiają modelowanie tych cech systemów komputerowych, dla których nie można uzyskać rozwiązania w postaci iloczynowej (poprzez dekompozycję) problemów analizy sieci kolejkowych.

Innym rozwiązaniem problemu oceny wydajności z zastosowaniem stochastycznych sieci Petriego, możliwym do przyjęcia również dla sieci z ogólniejszymi rozkładami czasów palenia przejść niż rozważane w pracy, jest symulacja oparta na tych sieciach. W szczególności można wykorzystać wyniki dla symulacji regeneracyjnej (z osiąganiem stanów o identycznych charakterystykach probabilistycznych) dla sieci stochastycznych [43].

Wśród mechanizmów stosowanych przy współpracy procesów sekwencyjnych, znaczenie komunikujących się procesów sekwencyjnych (CSP) i rachunku systemów komunikujących się (CCS) jest szczególnie duże. Jest

to spowodowane m.in. następującymi faktami:

1. Język LOTOS (język formalnych specyfikacji ułsug i protokołów w sieciach komputerowych budowanych zgodnie z wymaganiami Międzynarodowej Organizacji Standaryzacji - ISO) oparty jest na języku CCS,

2. Język Occam (dla Transputerów) skonstruowano na podstawie języka CSP.

Dlatego bardzo ważne wydaje się szukanie efektywnych metod oceny wydajności dla CSP i CCS z uwzględnieniem strukturalizacji i abstrahowania.

LITERATURA

[1] AJMONE MARSAN M., BALBO G., BOBBIO A., CHIOLA G., CONTE G., CUMANI A., On Petri nets with stochastic timing, [w:] [99] 80-87.

[2] AJMONE MARSAN M., BALBO G., CHIOLA G., CONTE G., Modeling the software architecture of a prototype parallel machine, [w:] Proc. SIGMETRICS Conf., ACM, Banf., Alberta, Canada, May 1987.

[3] AJMONE MARSAN M., BALBO G., CHIOLA G., DONATELLI S., On the product-form solution of a class of multiple-bus multiprocessor system models, J. Systems and Software, Vol. 1, No. 2, 1986, 117--124.

[4] AJMONE MARSAN M., BALBO G., CONTE G., A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems, ACM Trans. Comput. Systems, Vol. 2, May 1984, 93-122.

[5] AJMONE MARSAN M., BOBBIO A., CONTE G., CUMANI A., Performance analysis of degradable multiprocessor systems using generalized stochastic Petri nets, IEEE Computer Society Distributed Processing Technical Committee Newsletter, Vol. 6, No. SI-1, January 1984, 47-54.

[6] Analytical Queueing Models,IEEE Computer, Special Number, Vol.13, April 1980.

[7] ANDRE C., BOERI F., MARIN J., Synthèse et réalisation des systemes logiques à évolutions simultanées, RAIRO, Vol. 10, No 4, 1976, 67-86.

[8] ANDRE C., The behaviour of a Petri net on a subset of transitions, RAIRO, Vol. 17, No. 1, 1983, 5-21.

[9] AUGUSTYNEK Z., Natura czasu, Warszawa, PWN, 1975.

[10] AVEN T., Upper (lower) bounds on the mean of the maximum (minimum) of a number of random variables, J. Appl. Prob., Vol. 22, 1985, 723-728.

[11] BAER J.L., ELLIS C.S., Model, design and evaluation of a compiler for a parallel processing environment, IEEE Trans. Software Engrg., SE-3, No. 6, 1977, 394-405.

[12] BALBO G., BRUELL S.C., GHANTA S., Combining queueing network and generalized stochastic Petri net model for the analysis of some software blocking phenomena, IEEE Trans. Software Engrg., SE-12, No. 4, 1986, 561-576.

[13] BASKETT F., CHANDY K.M., MUNTZ R., PALACIOS J., Open, closed, and mixed networks of queues with different classes of customers, J. ACM, Vol. 22, No. 2, 1975, 248-260.

[14] BERTHELOT G., MEMMI G., REISIG W., A control structure for sequential processes synchronized by buffers,[w:]Proc. Conf. Information Sciences and Systems, The John Hopkins University, Baltimore, USA, March 1983.

[15] BERTHELOT G., TERRAT R., Petri nets theory for correctness of protocols, IEEE Trans. Comm., COM-30, No. 12, 1982, 2497-2505.

[16] BŁAŻEWICZ J., Problemy optymalizacji kombinatorycznej - złożoność obliczeniowa, algorytmy aproksymacyjne, Warszawa-Łódź, PWN, 1986.

[17] BŁAŻEWICZ J., CELLARY W., SŁOWIŃSKI R., WĘGLARZ J., Badania operacyjne dla informatyków, Warszawa, WNT, 1983.

[18] BŁAŻEWICZ J., DRABOWSKI M., WĘGLARZ J., Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans. Comput., C-35, No. 5, 1986, 389-393.

[19] BRINCH HANSEN P., The programming language Concurrent Pascal, IEEE Trans. Software Engrg., SE-1, No. 3, 1975, 199-207.

[20] BRINCH HANSEN P., The Architecture of Concurrent Programs, Prentice-Hall, 1977.

[21] BRINCH HANSEN P., Distributed processes: A concurrent programming concept, Comm. ACM, Vol. 21, No. 11, 1978, 934-941.

[22] BRUNO G., MARCHETTO G., Process-translatable Petri nets for the rapid prototyping of process control systems, IEEE Trans. Software Engrg., SE-12, No. 2, 1986, 346-357.

[23] CARLIER J., CHRETIENNE P., GIRAULT C., Modelling scheduling problems with timed Petri nets,[w:] G. Rozenberg (red.), Advances in Petri Nets 1984, Lecture Notes in Computer Science, Vol. 188, Springer, Berlin 1985, 62-82.

[24] CHRETIENNE P., Le réseaux de Petri temporisés, Thése d'Etat, Universite Paris VI, 1983.

[25] CHRETIENNE P., Analyse des régimes transitoire et asymptotique d'un graphe d'evénements temporisé, TSI, Vol. 4, No. 1, 1985, 127-143.

[26] COFFMAN E.G., Jr., (red.), Computer and Job-Shop Scheduling Theory, John Wiley & Sons, New York 1976.

[27] COFFMAN E.G., Jr., DENNING P.J., Operating Systems Theory, Englewood Cliffs, Prentice Hall 1973.

[28] COOLAHAN J.E., Jr., ROUSSOPOULOS N., Timing requirements for time-driven systems using augmented Petri nets, IEEE Trans. Software Engrg., SE-9, No. 5, 1983, 603-616.

[29] COURTOUIS P.J., Error analysis in nearly-completely decomposable stochastic systems, Econometrica, Vol. 43, No. 4, 1975, 691-709.

[30] COURTOUIS P.J., SEMAL P., Bounds for the positive eigenvectors of nonnegative matrices and their approximations by decomposition, J. ACM, Vol. 31, No. 4, 1984, 804-825.

[31] CRESPI-REGHIZZI S., MANDRIOLI D., A decidability theorem for a class of vector-addition systems, Inform. Processing Lett., Vol. 3, No. 3, Jan. 1975, 78-80.

[32] DEMPSTER M.A.H., LENSTRA J.K., RINNOY KAN A.H.G., Deterministic and Stochastic Scheduling, D. Reidel Publishing Company, Dordrecht, Boston, London, 1982.

[33] DEVROYE L.P., Inequalities for the completion times of stochastic PERT networks, Mathematics of Operations Research, Vol. 4, No. 4, Nov. 1979, 441-447.

[34] DIJKSTRA E.W., The structure of the THE multiprogramming system, Comm. ACM, Vol. 11, No. 5, 1968, 341-346.

[35] ELMAGHRABY S.E., Activity Networks: Project Planning and Control by Network Models, John Wiley & Sons, New York, 1977.

[36] FIERLA A., Performance evaluation using timed marked graphs, praca przedłożona do recenzji.

[37] FLORIN G., NATKIN S., On open synchronized queueing networks [w:] [99], 226-233.

[38] FORTIER P.J., Design and Analysis of Distributed Real-Time Systems, Mc Graw-Hill, Inc. New York, 1985.

[39] GAREY M.R., JOHNSON D.S., Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[40] GENRICH H.J., LAUTENBACH K., System modelling with high-level Petri nets, Theoret. Comput. Sci., Vol. 13, 1981, 109-136.

[41] GOLD M.E., Deadlock prediction: easy and difficult cases, SIAM J. on Computing, Vol. 7, No. 3, 1978, 320-336.

[42] GOPALAKRISHNAN G.C., Synthesizing synchronous digital VLSI Controllers using Petri nets, [w:] [100], 94-99.

[43] HAAS P.J., SHEDLER G.S., Stochastic Petri nets with simultaneous transition firings, [w:] [100], 24-32.

[44] HACK M., Analysis of production schemata by Petri nets, MIT, Cambridge, Project MAC, Rep. TR-94, 1972.

[45] HANEN C., Optimizing microprograms for recurrent loops on pipelined architectures using timed Petri nets, Univ. P. et M. Curie, Lab. MASI, Rep. RR 221, 1988.

[46] HANEN C., CHRETIENNE P., CARLIER J., Modelling and optimizing pipelines with timed Petri nets, Proc. 7th European Workshop Applications and Theory of Petri nets, Oxford 1986, 432-447.

[47] HEIDELBERGER P., LAVENBERG S., Computer performance evaluation methodology, IEEE Trans. Comput., Vol. C-33, No. 12, 1984, 1195--1219.

[48] HOARE C.A.R., Monitors: An operating systems structuring concept, Comm. ACM, Vol. 17, No. 10, 1974, 549-557.

[49] HOARE C.A.R., Communicating sequential processes, Comm. ACM, Vol. 21, No. 8, 1978, 666-677.

[50] HOLLYDAY M.A., VERNON M.K., A generalized timed Petri net model for performance analysis,[w:] [99], 181-190.

[51] HOWELL R.R., ROSIER L.E., Recent results on the complexity of problems related to Petri nets,[w:] [109], 45-72.

[52] JANTZEN M., VALK R., Formal properties of place/transition nets, [w:] Brauer W. (red.), Net Theory and Applications, Lecture Notes in Computer Science, Vol. 84, Springer, Berlin 1980, 165-211.

[53] JARNICKI J., MAGOTT J., Hierarchic performance evaluation of concurrent programming using Petri nets,[w:] Proc. Int. Conf. RELCOMEX'87, Książ, Poland, October 1987, 99-106.

[54] JENSEN K., Coloured Petri nets and the invariant-method, Theoret. Comput. Sci., Vol. 14, No. 3, 1981, 317-336.

[55] JONES N.D., LANDWEBER L.H., LIEN Y.D., Complexity of some problems in Petri nets, Theoret. Comput. Sci., Vol. 4, 1977, 277-299.

[56] KAHN G., Semantics of simple language for parallel programming, Information Processing 74, IFIP, North-Holland, 1974, 471-475.

[57] KARMARKAR N., A new polynomial-time algorithm for linear programming, Combinatorica, Vol. 4, No. 4, 1984, 373-395.

[58] KAVI K.A., BUCKLES B.P., NARAYAN BHAT U., Isomorphisms between Petri nets and data flow graphs, IEEE Trans. Software Engrg., SE-13, No. 10, 1987, 1127-1134.

[59] KELLER R.M., Formal verification of parallel programs, Comm. ACM, Vol. 19, No. 7, 1976, 371-384.

[60] KESSELS J.L.W., The Soma: A programming construct for distributed computations, IEEE Trans. Software Engrg., SE-7, No. 5, 1981.

[61] KLUGE W.E., SCHLÜTTER H., Petri net models for the evaluation of applicative programs based on λ-expressions, IEEE Trans. Software Engrg., SE-9, No. 4, 1983, 411-427.

[62] KOROLJUK W.S., TURBIN A.F., Połumarkowskije prociessy i ich priłożenija, Naukowa Dumka, Kijów 1978.

[63] KOSARAJU S.R., Decidability of reachability in vector addition systems, [w:] Proc. 14th Ann. Symp. on Theory of Computing, 1982, 267-281.

[64] KRÄMER B., Stepwise construction of non-sequential software systems using a net-based specification language, [w:] Advances in Petri Nets, 1984, G. Rozenberg (red.), Lecture Notes in Computer Science, Vol. 188, Springer, Berlin 1985, 307-330.

[65] KRÜCKBERG F., JAXY M., Mathematical methods for calculating invariants in Petri nets, [w:] [109], 104-131.

[66] LANDWEBER L.H., ROBERTSON E.L., Properties of conflict free and persistent Petri nets, J. ACM, Vol. 25, No. 3, 1978, 352-364.

[67] LEDGARD H.F., MARCOTTY M., A genealogy of control structures, Comm. ACM, Vol. 18, No. 11, 1975, 629-639.

[68] LEUNG J.Y.T., MONIEN B., On the complexity of deadlock recovery, Fundamenta Informaticae, Vol. IX, 1986, 323-342.

[69] LEWIS T.G., Software Engineering, Analysis and Verification, Prentice-Hall Company, Preston, Virginia, 1982.

[70] LIPSKI W., Kombinatoryka dla programistów, Warszawa, WNT, 1982.

[71] MAGOTT J., Abstrahowanie dla rozszerzonych grafów markowanych, Podstawy Sterowania, Tom 12, 1982, z. 3-4, 97-116.

[72] MAGOTT J., Abstrahowanie dla sieci Petriego, Podstawy Sterowania, Tom 13, 1983, z. 1, 75-98.

[73] MAGOTT J., Performance evaluation of concurrent systems using Petri nets, Inform. Processing. Lett., Vol. 18, Jan. 1984, 7-13.

[74] MAGOTT J., Performance evaluation of systems of cyclic sequential processes with mutual exclusion using Petri nets, Inform. Processing. Lett., Vol. 21, Nov. 1985, 229-232.

[75] MAGOTT J., New NP-complete problems in performance evaluation of concurrent systems using Petri nets, IEEE Trans. Software Engrg., SE-13, No. 5, 1987, 578-581.

[76] MAGOTT J., Czasowe sieci Petriego - możliwości, ograniczenia, rozszerzenia, Zeszyty Naukowe Politechniki Śląskiej Nr 894, Automatyka, Gliwice 1986, 141-152.

[77] MAGOTT J., Performance evaluation of systems of cyclic sequential processes with mutual exclusion and communication by buffers using Petri nets, [w:] [100], 146-153.

[78] MAGOTT J., Performance evaluation of concurrent systems using conflict-free and persistent Petri nets, Inform. Processing Lett., Vol. 26, Oct. 1987, 77-80.

[79] MAGOTT J., Złożoność obliczeniowa algorytmów i problemów minimalnego czasu cyklu dla wybranych klas sieci Petriego, artykuł przyjęty do druku w Archiwum Automatyki i Telemechaniki.

[80] MARTINEZ J., SILVA M., A simple and fast algorithm to obtain all invariants of a generalized Petri net, [w:] C. Girault, W. Reisig (red.), Applications and Theory of Petri Nets, Fachberichte Informatik, Vol. 52, Springer, Berlin 1982, 301-310.

[81] MAZURKIEWICZ A., Trace theory, [w:] Brauer W., Reisig W., Rozenberg G. (red.), Proc. Advanced Course Petri Nets, Bad Honnef, September 1986, Part II, Lecture Notes in Computer Science, Vol. 255, Springer, Berlin 1987, 279-324.

[82] MEIJER E., Petri net models for the λ-calculus, [w:] [109] 162-180.

[83] MEKLY L.J., YAU S.S., Software design representation using abstract process networks, IEEE Trans. Software Engrg., SE-6, No. 5, 1980, 420-435.

[84] MEMMI G., FINKEL A., An introduction to FIFO nets-Monogeneous nets: a subclass of FIFO nets, Theoret. Comput. Sci., Vol. 35, 1985, 191-214.

[85] MERLIN P., FARBER D.J., Recoverability of communication protocols - Implications of a theoretical study, IEEE Trans. Comm., COM-24, No. 9, 1976, 1036-1043.

[86] MILNE G.J., CIRCAL: A calculus for circuit description, INTEGRATION, the VLSI journal, Vol. 1, 1983, 121-160.

[87] MILNER R., A Calculus of Communicating Systems, Lecture Notes in Computer Science, Vol. 92, Springer, Berlin 1980.

[88] MISUNAS D., Petri nets and speed independent design, Comm. ACM, Vol. 16, No. 8, 1973, 474-481.

[89] MOLLOY M.K., Performance analysis using stochastic Petri nets, IEEE Trans. Comput., C-31, No. 9, 1982, 913-917.

[90] MOLLOY M.K., On the integration of delay and throughput measures in distributed processing models, Ph.D. dissertation, Univ. of California, Los Angeles, 1981.

[91] MORGAN E.T., RAZOUK R.R., Interactive state-space analysis of concurrent systems, IEEE Trans. Software Engrg., SE-13, No. 10, 1987, 1080-1091.

[92] NATKIN S., Reseaux de Petri stochastiques, These de Docteur Ingeneur CNAM - Paris, June 1980.

[93] NELSON R.A., HAIBT L.M., SHERIDIAN P.B., Casting Petri nets into programs, IEEE Trans. Software Engrg., SE-9, No. 5, 1983, 590-602.

[94] NIELSEN M., CCS and its relationship to net theory, [w:] BRAUER W., REISIG W., ROZENBERG G., (red.), Proc. Avanced Course Petri Nets, Bad Honnef, September 1986, Part II, Lecture Notes in Computer Science, Vol. 255, Springer, Berlin 1987, 393-415.

[95] OLDEROG E.R., Operational Petri net semantics for CCSP, [w:] [109], 196-223.

[96] OZSU M.T., Modeling and analysis of distributed database concurrency control algorithms using an extended Petri net formalism, IEEE Trans. Software Engrg., SE-11, No. 10, 1985, 1225-1239.

[97] PATIL S.S., DENNIS J.B., The description and realization of digital systems, RAIRO, Feb. 1973, 55-66.

[98] PETERSON J.L., Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, 1981.

[99] Proc. Int. Workshop Timed Petri Nets, Torino, Italy, July 1985, IEEE Computer Society Press, 1985.

[100] Proc. Int. Workshop Petri Nets and Performance Models, Madison, USA, August 1987, IEEE Computer Society Press, 1987.

[101] RAMAMOORTHY C.V., HO G.S., Performance evaluation of asynchronous concurrent systems by Petri nets, IEEE Trans. Software Engrg., SE-6, No. 5, 1980, 440-449.

[102] RAMCHANDANI C., Analysis of asynchronous concurrent systems by Petri nets, MIT, Cambridge, Project MAC, Rep. TR-120, 1974.

[103] RAZOUK R.R., PHELPS C.V., Performance analysis using timed Petri nets, Univ. California, Irvine, Depart. Inform. Comput. Sci., Rep. TR 206, 1983.

[104] REGHBATI H.K., Performance analysis of message-based systems, [w:] Sunshine C. (red.), Proc. 2nd Int. Workshop Protocol Specification, Testing, and Verification, Iddyllwild, California, USA, May 1982, North-Holland, 1982.

[105] REISER M., LAVENBERG S.S., Mean-value analysis of closed multichain queuing networks, J. ACM, Vol. 27, No. 2, 1980, 313-322.

[106] REISIG W., Deterministic buffer synchronization of sequential processes, Acta Informatica, Vol. 18, 1982, 117-134.

[107] REISIG W., Petri Nets: An Introduction, Springer, Berlin 1985.

[108] ROLIN P., Using Petri nets in measurement of a distributed data base system, [w:] C. Girault, W. Reisig (red.), Applications and Theory of Petri Nets, Fachberichte Informatik, Vol. 52, Springer, Berlin 1982.

[109] ROZENBERG G. (red.), Advances in Petri Nets 1987, Lecture Notes in Computer Science, Vol. 266, Springer, Berlin 1987.

[110] SHAPIRO S.D., A stochastic Petri net with applications to modelling occupancy times for concurrent task systems, Networks, Vol. 9, 1979, 375-379.

[111] SIFAKIS J., Use of Petri nets for performance evaluation, [w:] Beilner R., Gelenbe E. (red.), Measuring, Modelling and Evaluating Computer Systems, North-Holland, Amsterdam 1977, 75-93.

[112] SIFAKIS J., Structural properties of Petri nets,[w:]J. Winkowski (red.), Mathematical Foundations of Computer Science 1978, Lecture Notes in Computer Science, Vol. 64, Springer, Berlin 1978, 474-483.

[113] SIFAKIS J., Performance evaluation of systems using nets,[w:] Brauer W. (red.), Net Theory and Applications, Lecture Notes in Computer Science, Vol. 84, Springer, Berlin 1980, 307-319.

[114] SMIGIELSKI T., MURATA T., SOWA M., A timed Petri net and simulation of dataflow computer, [w:] [99], 56-63.

[115] STANKOVIC J.A., RAMAMRITHAM K., CHENG S., Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems, IEEE Trans. Comput., C-34, No. 12, 1985, 1130-1143.

[116] STAUNSTRUP J., Message passing communication versus procedure call communication, Software - Practice and Experience, Vol. 12, 1982, 223-234.

[117] STOTTS P.D. Jr., A hierarchical graph model of concurrent real--time software systems, Ph.D. dissertation, University of Virginia, Charlottesville, 1985.

[118] TRIVEDI K.S., Probability and Statistics with Reliability,Queueing and Computer Science Applications, Edgewood Cliffs, Prentice--Hall, 1982.

[119] VALETTE R., Analysis of Petri nets by stepwise refinement, J. Comput. System Sci., Vol. 18, No. 1, 1979, 35-46.

[120] VERNON M., ZAHORJAN J., ŁAZOWSKA D., A comparison of performance Petri nets and queueing network models, Univ. Washington, Seattle, Comput. Sci. Depart., Rep. TR 86-09-09, 1986.

[121] VIDONDO F., Galileo: Design language for real-time systems [w:] Proc. ITT Conf. Programming Productivity and Quality, New York, June 1983.

[122] VOSS K., Using predicate/transition nets to model and analyze distributed database systems, IEEE Trans. Software Engrg., SE-6, No. 6, 1980, 539-544.

[123] WINKOWSKI J., An algebraic description of system behaviours, Theoret. Comput. Sci., Vol. 21, 1982, 315-340.

[124] YAU S.S., CAGLAYAN M.U., Distributed software system design representation using modified Petri nets, IEEE Trans. Software Engrg., SE-9, No. 6, 1983, 733-745.

[125] ZAMOJSKI W., A reliability-functional computer system model based on Petri nets, [w:] Proc. IASTED Int. Symp. Reliability and Quality Control, Paris, France, June 1987, 89-91.

[126] ZHAO W.W., RAMAMRITHAM K., STANKOVIC J.A., Scheduling tasks with resource requirements in hard real-time systems, IEEE Trans. Software Engrg., SE-13, No. 5, 1987, 564-577.

[127] ZUBEREK W.M., Timed Petri nets and preliminary performance evaluation, [w:] Proc. 7th IEEE Ann. Symp. Computer Architecture, La Baule, France, May 1980, 88-96.

[128] ZUBEREK W.M., Application of timed Petri nets to analysis of multiprocessor realizations of digital filters, [w:] Proc. 25 Symp. Circuits and Systems, Houghton, Michigan, August 1982.

Praca wpłynęła do Redakcji 1988.05.10

PETRI NETS IN COMPUTER SYSTEMS PERFORMANCE EVALUATION

In the present paper, performance of computer systems is evaluated by means of the timed Petri nets. For this purpose, there are developed the foundations of deterministic Petri nets (a transition firing time is given by a real number, transitions to fire are chosen according to a vector of relative frequencies of transition firings). The main sequential processes cooperation facilities such as mutual exclusion, communication by synchronizing a sender and a receiver, communication by buffers are studied and evaluation of their performance is carried out using Petri nets (in particular, the deterministic ones). With regard to structural and functional complexity of computer systems, the main point is the computational complexity of performance evaluation problems. For systems of sequential processes with the above cooperation facilities, properties that are critical for computational complexity have been found.

СЕТИ ПЕТРИ В ОЦЕНКЕ ПРОИЗВОДИТЕЛЬНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Настоящая работа посвящена оценке производительности компьютерных систем с употреблением сетей Петри. В работе развиты основы детерминистических сетей Петри (время топки перехода задано действительным числом, зато выбор переходов для топки должен удовлетворять ограничениям, заданным вектором взаимных частот топки переходов) для потребностей оценки производительности компьютерных систем. Кроме этого, проведён анализ производительности таких основных механизмов содействия согласованно проходящих секвенционных процессов как: взаимное исключение, связь через синхронизацию отправителя и получателя, связь через буферы. Оценки производительности сделаны на базе сетей Петри (с особым учётом детерминистических сетей). Из-за структурной и функциональной сложностей компьютерных систем, основным вопросом в области проблем оценки производительности является расчётная сложность этих проблем. Для систем секвенционных процессов с вышеперечисленными механизмами содействия определены свойства, которые являются критическими для расчётной сложности.

SPIS TREŚCI

WSTĘP 4
0.1. Współbieżność w systemach komputerowych 4
0.2. Znaczenie sieci Petriego w ocenie wydajności systemów komputerowych 6
0.3. Cel pracy 9
0.4. Zakres pracy 11
1. DEFINICJE 15
1.1. Definicje podstawowe 15
1.2. Sieci Petriego z czynnikiem czasu 23
1.2.1. Deterministyczne sieci Petriego 25
1.2.2. Stochastyczne sieci Petriego 32
1.2.3. Mieszane sieci Petriego 36
1.3. Złożoność obliczeniowa problemów kombinatorycznych 36
2. ZŁOŻONOŚĆ OBLICZENIOWA PROBLEMU CZASU CYKLU DLA WYBRANYCH KLAS DETERMINISTYCZNYCH SIECI PETRIEGO 41
2.1. Wybrane klasy sieci 42
2.2. Zależności między klasami sieci 49
2.3. Rozstrzygalność problemu czasu cyklu 54
2.4. Złożoność obliczeniowa problemu czasu cyklu 55
2.4.1. Sieci P-niezmiennicze 55
2.4.2. Sieci nie P-niezmiennicze 65
2.4.3. NP-zupełność problemu czasu cyklu 67
2.5. Oszacowanie minimalnego czasu cyklu 68
2.5.1. Sieci P-niezmiennicze 68
2.5.2. Sieci nie P-niezmiennicze 71
2.6. Podsumowanie 72
3. OCENA WYDAJNOŚCI SYSTEMÓW PROCESÓW SEKWENCYJNYCH Z WZAJEMNYM WYKLUCZANIEM 73
3.1. Charakterystyka badanej klasy systemów cyklicznych procesów sekwencyjnych z wzajemnym wykluczaniem 74
3.2. Przypadek deterministyczny 75
3.3. Przypadek stochastyczny 80
3.4. Podsumowanie 85
4. OCENA WYDAJNOŚCI KOMUNIKUJĄCYCH SIĘ PROCESÓW SEKWENCYJNYCH (CSP) 88
4.1. Reprezentacja sieciowa komunikujących się procesów sekwencyjnych 89
4.2. Przypadek deterministyczny 93
4.2.1. Zachowanie przejściowe 93
4.2.2. Zachowanie cykliczne 97
4.3. Przypadek stochastyczny 99
4.4. Podsumowanie 99

5. OCENA WYDAJNOŚCI SYSTEMÓW PROCESÓW SEKWENCYJNYCH KOMUNIKUJĄCYCH SIĘ POPRZEZ BUFORY 101
5.1. Reprezentacja sieciowa systemów procesów komunikujących się poprzez bufory 102
5.2. Przypadek deterministyczny 105
5.3. Podsumowanie 110
6. ZAKOŃCZENIE 111
LITERATURA 117

CONTENTS

INTRODUCTION 4
0.1. Concurrency in computer systems 4
0.2. Importance of Petri nets in evaluation of a computer system performance 6
0.3. Investigation objective 9
0.4. Scope of investigation 11
1. DEFINITIONS 15
1.1. Basic definitions 15
1.2. Petri nets with time factor 23
1.2.1. Deterministic Petri nets 25
1.2.2. Stochastic Petri nets 32
1.2.3. Mixed Petri nets 36
1.3. Computational complexity of combinatorial problems 36
2. COMPUTATIONAL COMPLEXITY OF CYCLE TIME PROBLEM FOR CHOSEN CLASSES OF DETERMINISTIC PETRI NETS 41
2.1. Chosen classes of nets 42
2.2. Relations between classes of nets 49
2.3. Decidability of cycle time problem 54
2.4. Computational complexity of cycle time problem 55
2.4.1. P-invariant nets 55
2.4.2. Non-P-invariant nets 65
2.4.3. NP-completeness of cycle time problem 67
2.5. Estimates of minimal cycle time 68
2.5.1. P-invariant nets 68
2.5.2. Non-P-invariant nets 71
2.6. Summary 72
3. PERFORMANCE EVALUATION OF SYSTEMS OF SEQUENTIAL PROCESSES WITH MUTUAL EXCLUSION 73
3.1. Description of the investigated class of systems of cyclic sequential processes with mutual exclusion 74
3.2. Deterministic case 75
3.3. Stochastic case 80
3.4. Summary 85
4. PERFORMANCE EVALUATION OF COMMUNICATING SEQUENTIAL PROCESSES (CSP) 88
4.1. Net representation of communicating sequential processes. 89
4.2. Deterministic case 93
4.2.1. Transient behaviour 93
4.2.2. Cyclic behaviour 97
4.3. Stochastic case 99
4.4. Summary 99

5. PERFORMANCE EVALUATION OF SYSTEMS OF SEQUENTIAL PROCESSES COMMUNICATING BY BUFFERS 101
5.1. Net representation of systems of sequential processes communicating by buffers 102
5.2. Deterministic case 105
5.3. Summary 110
6. CONCLUDING REMARKS 111
REFERENCES 117

PRACE NAUKOWE INSTYTUTU CYBERNETYKI TECHNICZNEJ
(wydane w latach 1986—1988)

Nr 72, Monografie nr 13, E. Rafajłowicz, *Dobór sterowań optymalnych w identyfikacji systemów liniowych o parametrach rozłożonych*, Wrocław 1986 150,—

Nr 73, Konferencje nr 30, *Microcomputer'86*, Wrocław 1986 138,—

Nr 74, Konferencje nr 31, *II naukowo-techniczna konferencja radzieckich i polskich młodych uczonych* (w języku rosyjskim), Wrocław 1986 200,—

Nr 75, Konferencje nr 32, *II Krajowa konferencja robotyki, t. I*, Wrocław 1988 470,—

Nr 76, Konferencje nr 33, *II Krajowa konferencja robotyki, t. II*, Wrocław 1988 420,—

Nr 77, Konferencje nr 34, *II Krajowa konferencja robotyki, t. III*, Wrocław 1988 460,—

Nr 78, Konferencje nr 35, *II Krajowa konferencja robotyki, t. IV*, Wrocław 1988 340,—

Nr 79, Konferencje nr 36, *Design, practice, education — Microcomputer '88*, Wrocław 1988 270,—

Cena zł 250,—

Subscription should be sent (at any time of the year) to:

„Ars Polona"

Krakowskie Przedmieście 7, 00-068 Warszawa
or OR PAN, 00-901 Warszawa, PKiN, POLAND
Bank account number: PKO BP XV Oddz. W-wa 1658-201045-139-11

Wydawnictwa Politechniki Wrocławskiej
ma stale na składzie Księgarnia Wr 49
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
oraz Wojewódzka Księgarnia Techniczna
ul. Świdnicka 8, 50-067 Wrocław

ISSN 0324-9786