INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ

Raport serii: PREPRINTY nr 83/2000

Metody oceny wydajności systemów opisywanych Mapami Stanów

(rozprawa doktorska)

Tomasz BABCZYŃSKI

PROMOTOR:

prof. dr hab. inż. Jan Magott

Słowa kluczowe:

- **inżynieria wydajności oprogramowania**
- **sieci Petriego**
- **procesy stochastyczne**
- **łańcuchy Markowa**

WROCŁAW
Czerwiec 2000

SPIS TREŚCI

1. Wprowadzenie

1.1. Ocena wydajności, inżynieria wydajności

Istnieją dwa podejścia do kwestii wydajności obiektowo projektowanych systemów. Pierwsze z nich — tradycyjne odkłada problem badania aspektów wydajnościowych do czasu zakończenia projektu lub implementacji jak opisano to w pracach [11] i [58]. Podejście takie zwiększa ryzyko niespełnienia nałożonych wymagań wydajnościowych przez gotowy projekt. Drugie, nowe podejście nazywane jest *inżynierią wydajności oprogramowania* (ang. *software performance engineering — SPE* [62]). Stosując to podejście powinno się brać pod uwagę ograniczenia wydajnościowe w czasie całego cyklu życia oprogramowania począwszy od tak wczesnych stadiów jak analiza wymagań czy projekt wstępny. W Inżynierii Wydajności Oprogramowania wykorzystywane są istniejące modele służące do oceny wydajności. Jako najważniejsze z nich wymienić tu należy:

— modele wydajnościowe programów sekwencyjnych i równoległych [21],
— Sieci Petriego [1],
— łańcuchy Markowa [21],
— modele kolejkowe [15].

Stosowanie tych modeli pozwala ocenić czy projektowany system spełnia narzucone wymagania odnośnie wydajności i na tej podstawie podejmować decyzje co do dalszego kierunku projektu. Ponieważ modele te wywodzą się z okresu gdy nie stosowano metod inżynierii wydajności oprogramowania, więc nie są zintegrowane z metodami specyfikacji i projektowania. Dla różnych modeli są prowadzone prace w tym kierunku. Istnieją już narzędzia stosujące metodę SPE i ułatwiające analizę zagadnień wydajnościowych podczas obiektowo zorientowanego projektowania systemów . Jedno z nich opisane jest w pracy [63]. Stosowanie tego narzędzia wymaga aby zachowanie się systemu w czasie opisywane było tzw. *grafem wykonania* (ang.

execution graph). Graf ten różni się od modeli dynamicznych stosowanych w różnych metodologiach projektowania obiektowo zorientowanego.

W projektowaniu zorientowanym obiektowo używane są trzy modele: strukturalny, dynamiczny i funkcjonalny. Przy rozważaniu zagadnień związanych z wydajnością najbardziej przydatny jest model dynamiczny. Jako ten rodzaj modelu najczęściej stosowane są Mapy Stanów. Jako przykłady wymienić można takie metodologie jak: metoda Boocha [11], OMT [58], a także UML [12], [65].

Jak pokazano w podręczniku technologii obiektowej [69], Mapy Stanów odgrywają kluczową rolę w różnych narzędziach wspomagających projektowanie obiektowe. Mają one istotną przewagę nad innymi stosowanymi formalizmami np. algebraicznymi polegającą na dostarczeniu przyjaznego użytkownikom graficznego interfejsu, co czyni specyfikacje zapisywane przy pomocy Map Stanów czytelniejszymi dla inżynierów nie będących informatykami.

Języki formalne i metody formalne, do których należą też Mapy Stanów są szczególnie zalecane przez normę IEC 1509 do tworzenia systemów mających wpływ na bezpieczeństwo, [59]. Warto rozwijać te właśnie metody tak, aby można było je stosować także w przypadku gdy są nałożone wymagania na wydajność tworzonych systemów. Formalne języki specyfikacji są rozszerzane o elementy związane z czasem co umożliwia ich zastosowanie do oceny wydajności systemów. Wymienić tu można czasowe rozszerzenia języka SDL [9], LOTOS [40] czy Estelle [44].

W ciągu ostatnich kilku lat daje się zauważyć duże zainteresowanie stochastycznymi algebrami procesowymi. Świadczą o nim liczne publikacje jak choćby: [10], [13], [30], [45], [64]. Algebry opisywane tam nie są jednak oparte na powszechnie przyjętych metodologiach projektowania zorientowanego obiektowo takich jak metoda Boocha [11], OMT [58] czy UML [12]. Metody powyższe używają natomiast formalizmu Map Stanów do opisu zachowania się projektowanych systemów w czasie.

Jak więc widać, Mapy Stanów mogą w najbliższym czasie stać się jednym z popularniejszych i ważniejszych narzędzi projektowania dużych systemów. Systemy takie mają nałożone różnorodne wymagania, które mają spełnić. Wśród nich także wymagania wydajnościowe. Przy projektowaniu systemów należy więc stosować narzędzia umożliwiające weryfikację poszczególnych faz projektu pod kątem spełniania wymagań, także wydajnościowych. Celowe jest więc rozwijanie najważniejszych metod formalnych tak, by umożliwiały one dokładniejsze badanie projektowanych systemów już we wczesnych fazach projektu co pozwoli na skrócenie czasu realizacji całego projektu oraz obniżenie jego kosztowności.

1.2. Metody specyfikacji, miejsce Map Stanów

Ważną fazą w cyklu życia oprogramowania jest jego specyfikacja. Może być ona wykonana metodami nieformalnymi jako ciąg zdań wyrażonych w języku naturalnym. Przy takim podejściu trudno się jednak ustrzec przed rozwlekłością, a równocześnie niejednoznacznością opisu i wynikającymi stąd błędami kodowania. Ponadto trudne jest sprawdzenie poprawności produktu względem specyfikacji. Z tego względu powstały i wciąż są rozwijane formalne języki i techniki specyfikacji. Pośrednim elementem są języki półformalne jak *PDL* (*Program Design Language*) zwany także Structured English oraz diagramy i drzewa decyzyjne gdzie pewne elementy opisu podawane są w języku naturalnym. W pełni formalnymi językami są np. *CSP* (*Communicating Sequential Processes*) [31] i *CCS* (*Calculus of Communicating Systems*) [46] stosowane do opisu procesów współbieżnych, czy *SDL* (*Specification and Description Language*) [56] stworzony dla opisu systemów komutujących w telefonii. Z tych języków wywodzą się zarówno języki kodowania (np. *Ada*, *Occam*) jak i nowocześniejsze języki specyfikacji (np. *Language Of Temporal Ordering Specifications — LOTOS* [34]). Spore znaczenie mają też takie formalne języki jak *Estelle* [35], *VDM* oraz notacja *Z*. Do języków formalnych zalicza się także Sieci Petriego wysokiego poziomu

(np. Kolorowane Sieci Petriego [36]), które także używane są do specyfikowania i projektowania systemów. Wymienić tu wreszcie należy diagramy i tablice stanów Automatów Skończenie Stanowych, które są bezpośrednim poprzednikiem Map Stanów. Porównanie wielu formalnych technik i języków specyfikacji znajduje się w pracy [14] natomiast praca [20] zawiera przegląd narzędzi komputerowych używanych w inżynierii oprogramowania (*CASE*).
Mapy Stanów (ang. Statecharts) zostały wprowadzone i zdefiniowane przez D. Harela w pracy [28]. Stanowią one rozszerzenie notacji diagramów stanów Automatów Skończenie Stanowych (ang. Finite State Machines — FSM) o hierarchię stanów, równoległość i rozpowszechnianie komunikatów. Rozszerzenia te miały na celu zwiększenie wygody posługiwania się omawianą notacją oraz jej czytelności. Omawiany tu formalizm stworzony został jako narzędzie służące do zapisu specyfikacji systemów oraz ich projektowania. Przewidywany był także jako wspólny język informatyków — projektantów systemów i inżynierów znających procesy technologiczne, ale nie znających informatyki. Od czasu wprowadzenia, Mapy Stanów znalazły szerokie zastosowanie przy specyfikacji i modelowaniu oprogramowania. Wiele narzędzi wspomagających projektowanie używa ich jako metody definiowania dynamiki systemu np. STATEMATE [24], lub UML [12], [17], [65]. Obecnie Mapy Stanów są uważane za podstawową metodę opisu zachowania się projektowanych systemów w technologiach zorientowanych obiektowo.

1.3. Cel, teza pracy

Praca ma na celu opracowanie metod oceny wydajności systemów projektowanych przy użyciu technik obiektowych. Cel ten ma być osiągnięty poprzez wprowadzenie rozszerzeń czasowych i stochastycznych do szeroko stosowanego w metodologii obiektowej formalizmu Map Stanów.

Tezą pracy jest stwierdzenie, że możliwe jest wprowadzenie do formalizmu Map Stanów takich rozszerzeń, które pozwolą stosować przy projektowaniu

systemów Inżynierię Wydajności Oprogramowania. Umożliwi to ocenę aspektów wydajnościowych projektu w każdej jego fazie począwszy od wstępnej specyfikacji. Podejście takie zwiększy efektywność projektowania systemów o nałożonych wymaganiach odnośnie ich wydajności.

1.4. Struktura pracy

Rozdział 2 przedstawia istniejące wersje Map Stanów. Zaprezentowane są w nim podstawowe diagramy wprowadzone przez Harela, znane z literatury rozszerzenia związane z czasem oraz niektóre narzędzia komputerowe implementujące Mapy Stanów.

Rozdział 3 przedstawia propozycję sposobu wykorzystania Map Stanów do oceny wydajności poprzez ich konwersję do pewnej klasy sieci Petriego.

Druga propozycja, opisana w rozdziale 4, wprowadza bezpośrednio do modelu Map Stanów, losowe opóźnienia przejść oraz deterministyczne czasy przeterminowania przejść i czas życia zdarzeń. Przedstawiona jest formalna algebraiczna definicja składni i semantyki.

Trzecia propozycja (rozdz. 5) rozszerza podstawowy model Map Stanów o losowe opóźnienia otwarcia przejść, a także generacji zdarzeń. Zmienne losowe opisujące je mają rozkłady wykładnicze. Dzięki wprowadzeniu do modelu priorytetów przejść możliwe jest uniknięcie niedeterminizmu. Dla tak rozszerzonych Map Stanów możliwe jest zbudowanie łańcucha Markowa z czasem ciągłym i opracowanie analitycznej metody oceny wydajności także zaprezentowanej w omawianym rozdziale.

Modyfikacja Markowowskich Map Stanów wprowadzająca dowolne rozkłady zmiennych losowych pokazana jest w rozdziale 6. Zdefiniowane tam Wydajnościowe Mapy Stanów mogą być wykorzystane do symulacyjnych analiz projektowanych systemów. Podejście symulacyjne zaprezentowane jest w rozdziale 6.6. Jest tam opisany symulator Wydajnościowych Map Stanów napisany przez autora niniejszej pracy. Przedstawione są także wyniki badań przykładowego problemu.

Wszystkie propozycje porównane są w rozdziale 7.

2. Omówienie Map Stanów

2.1. Podstawowe Mapy Stanów Harela

2.1.1. Higrafy

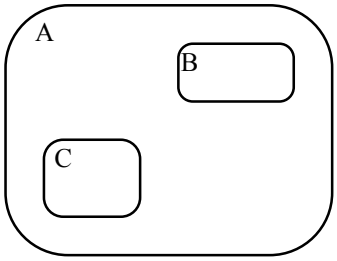
D. Harel zaproponował użycie do specyfikacji systemów reaktywnych formalizmu nazwanego *higrafami*. Higrafy stanowią połączenie idei i notacji hipergrafów z kołami Eulera i ich późniejszym rozwinięciem — diagramami Venna. Zwykłe multigrafy skierowane, używane zazwyczaj do opisu automatów skończenie stanowych składają się z punktów symbolizujących stany oraz strzałek obrazujących przejścia między nimi. Żaden z tych elementów nie posiada powierzchni, ani też żadne znaczenie nie jest przypisane ich lokalizacji. Inaczej rzecz się ma w przypadku higrafów. Tu położenie poszczególnych obiektów graficznych oraz ich wzajemne przenikanie ma tak samo duże znaczenie, jak połączenia węzłów za pomocą łuków. Co więcej, podobnie jak w hipergrafach, krawędzie mogą łączyć nie pary węzłów, a ich zbiory. Jednym z zastosowań higrafów są zaproponowane przez Harela Mapy Stanów (ang. Statecharts). Są one rozwinięciem o hierarchię stanów, równoległość akcji i rozpowszechnianie komunikatów, metod specyfikacji automatów skończenie stanowych. Podobnie jak przy graficznym opisie tych automatów w postaci diagramów stanów mamy do czynienia ze stanami i przejściami między nimi. Występują tu jednak dodatkowe elementy i konstrukcje, o których będzie mowa w następnych podrozdziałach.

2.1.2. Stany

Stany oznaczane są na Mapie Stanów za pomocą prostokątów z zaokrąglonymi rogami. W celu umożliwienia jednoznacznej identyfikacji i rozróżniania stanów, każdy prostokąt ma przypisaną unikalną nazwę, którą zwyczajowo umieszcza się w górnym lewym rogu danego prostokąta.

2.1.3. Hierarchia i typy stanów

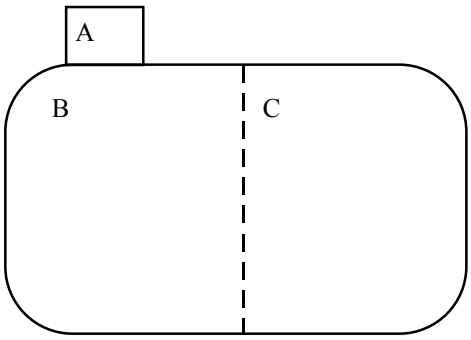
Umieszczenie danego prostokąta wewnątrz innego oznacza, że stan opisywany nim jest podstanem drugiego stanu. Innymi słowy wnętrze danego prostokąta wraz z elementami, które zawiera jest jego uszczegółowieniem. Możliwa jest także odwrotna interpretacja, która stanowi, że nadstany ogarniają wspólne właściwości podstanów i generalizują te własności.



Rys. 2.1 Przykład oznaczania hierarchii stanów

Rysunek 2.1 przedstawia przykład omawianej notacji. Mamy tutaj *nadstan* **A** oraz dwa jego *podstany* **B** i **C**. Ten rodzaj hierarchii nazywamy hierarchią typu *XOR* gdyż przebywanie systemu w nadstanie implikuje jego przebywanie w dokładnie jednym z podstanów. Typ nadstanu w tym rodzaju hierarchii określamy jako *XOR*.

Wprowadzona została także hierarchia typu *AND* oznaczana za pomocą linii przerywanych dzielących prostokąt przedstawiający stan na kilka części.



Rys. 2.2 Przykład hierarchii typu *AND*

Fakt przebywania systemu w stanie **A** pociąga za sobą konieczność jego przebywania także jednocześnie w obu stanach potomnych **B** i **C**. Ten rodzaj

zapisu przedstawia równoległość (lub ortogonalność). Nadstan w tym rodzaju hierarchii ma typ *AND*. Stany bezpośrednio potomne tego stanu są do siebie *ortogonalne*. Jeśli stan $\mathbf{S_1}$ jest ortogonalny do $\mathbf{S_2}$, to każdy stan potomny $\mathbf{S_1}$ jest ortogonalny do $\mathbf{S_2}$ oraz do każdego potomka $\mathbf{S_2}$**.** W przykładzie z rys. 2.2 oznaczałoby to, że stany **B** i **C** są ortogonalne, a także każdy stan potomny stanu **B** byłby ortogonalny do każdego potomka stanu **C**.

Relacja hierarchii jest niezwrotna, antysymetryczna i przechodnia. Ma ona też narzucone pewne ograniczenia:

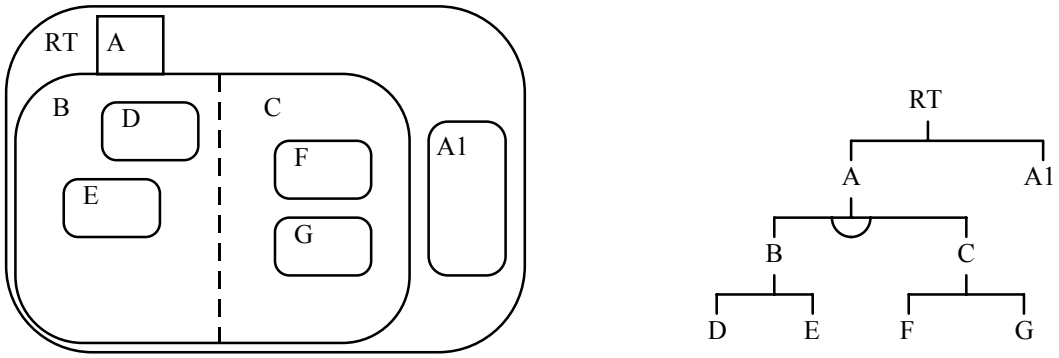
— stan rodzic całego projektowanego systemu nazywany *korzeniem* (ang. *root*) musi być typu *XOR*,

— każdy stan z wyjątkiem korzenia ma dokładnie jednego rodzica,

— każdy ze stanów potomnych (bezpośrednio) stanu typu *AND* musi mieć typ *XOR*,

— ponadto każdy stan nie posiadający stanów potomnych ma typ *PRIM*.

Relację hierarchii możemy przedstawić za pomocą drzewa jak na rys. 2.3.



Rys. 2.3 Przykład drzewa hierarchii stanów

2.1.4. Przejścia

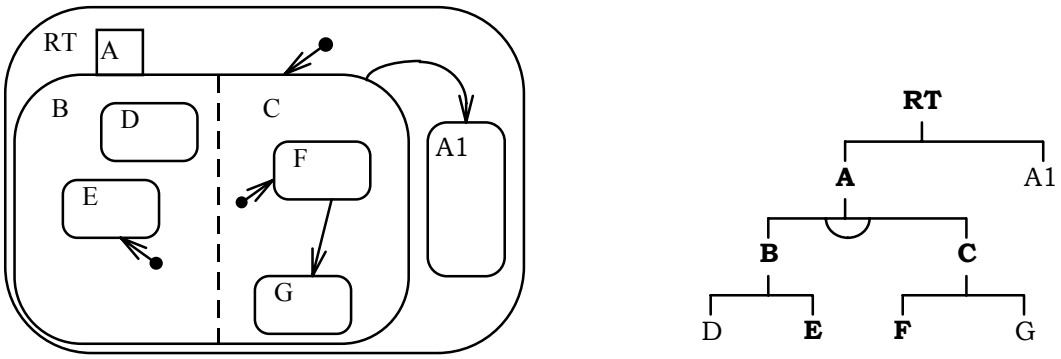
W poprzednim punkcie omówione zostały stany i zależności pomiędzy nimi. Do opisu dynamiki projektowanego systemu potrzebny jest jednak jeszcze jeden element — przejścia. *Przejściami* nazywamy zmiany stanu projektowanego lub specyfikowanego systemu zachodzące pod wpływem określonych *zdarzeń*, przy spełnieniu określonych *warunków* i z wykonaniem

pewnych *akcji*. Przejścia na Mapie Stanów rysowane są jako strzałki zaczynające się i kończące na krawędziach prostokątów oznaczających stany. Niedopuszczalne są przejścia zaczynające się lub kończące w stanach bezpośrednio potomnych stanu typu *AND*, a także w korzeniu. Każda strzałka może być opatrzona *etykietą* specyfikującą zdarzenia i warunki umożliwiające zajście przejścia oraz akcje wykonywane w czasie jego realizacji. W podstawowej wersji Map Stanów ogólny format etykiety jest następujący:

zdarzenie$_1$ **op** zdarzenie$_2$ ... [warunek$_1$ **op** warunek$_2$ ...]/akcja$_1$;akcja$_2$...

gdzie **op** jest jednym z operatorów logicznych (and, or, not). W wyrażeniach dopuszczalne są też nawiasy określające kolejność działań.

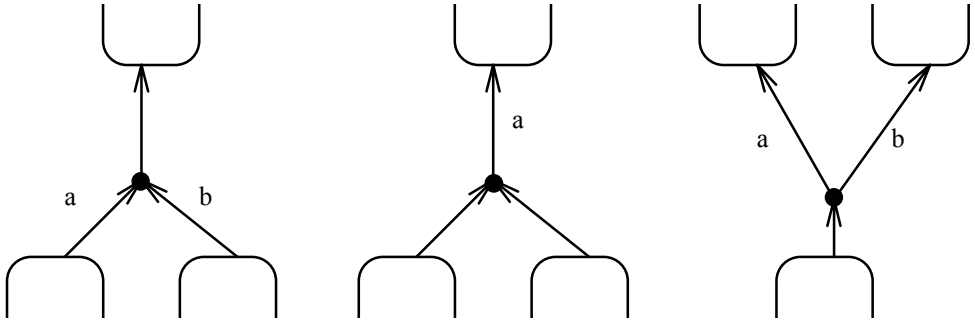
Jeden ze stanów potomnych każdego stanu typu *XOR* jest stanem *domyślnym*, co jest zaznaczane za pomocą strzałki rozpoczynającej się zaciemnionym kółkiem. W przypadku, gdy dany stan jest jedynym potomkiem stanu typu *XOR*, to jest on stanem domyślnym, czego się zwykle dodatkowo nie zaznacza. Stan domyślny jest tym stanem, do którego system przechodzi w momencie wejścia do nadstanu. Rysunek 2.4 przedstawia przykład zaznaczenia stanów domyślnych oraz kilku przejść między stanami. Na drzewie hierarchii wytłuszczone są nazwy stanów domyślnych.



Rys. 2.4

2.1.5. Łączenie i rozdzielanie przejść

Przejścia nie muszą być oznaczane tylko jedną strzałką lecz mogą się składać z kilku odcinków. Konstrukcję taką nazywamy *przejściem złożonym*. Jeżeli kilka przejść ma wspólny stan źródłowy albo stan przeznaczenia, można je rysować jako wiązkę strzałek o wspólnym końcu albo początku. Ponadto jeśli w etykietach poszczególnych strzałek występuje to samo zdarzenie, warunek lub akcja, to elementy te mogą być umieszczane we wspólnej etykiecie. Jeżeli etykiety występują przy kilku odcinkach przejścia złożonego, to etykietę całości tworzymy poprzez wzięcie iloczynu logicznego zdarzeń z poszczególnych części, iloczynu warunków, oraz sumy zbiorów wykonywanych akcji. Poniżej pokazano przykłady omawianych konstrukcji.

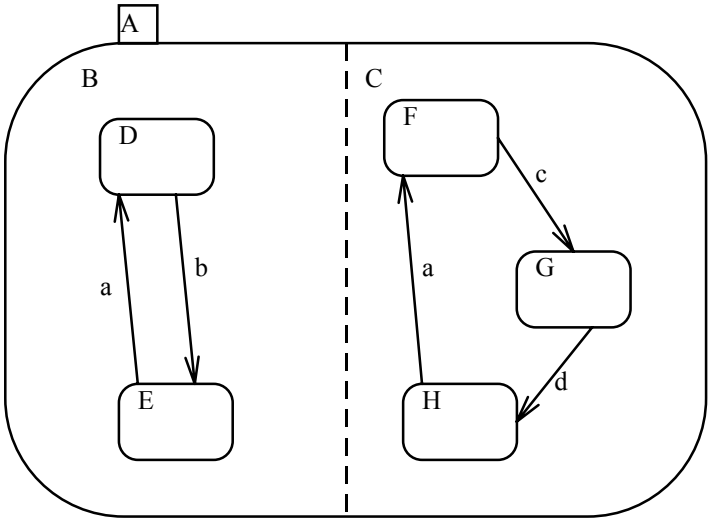


Rys. 2.5 Przykład łączenia i rozdzielania przejść

Konstrukcji tych oprócz faktoryzacji powtarzających się fragmentów etykiet lub odcinków przejść, używa się także dla zaznaczenia przejścia z lub do stanów ortogonalnych. Wtedy jednak konstrukcja omawiana ma nieco inne znaczenie. W sytuacji przedstawionej na rys. 2.5 mamy za każdym razem po dwa przejścia złożone, z których tylko jedno może zajść w danym momencie gdyż wzajemnie się one wykluczają. W przypadku zaś gdy źródłem przejścia są stany ortogonalne lub gdy przejścia kończą się w stanach ortogonalnych, mamy do czynienia z jednym przejściem złożonym, które może być albo wykonana w całości, albo wcale.

2.1.6. Ortogonalność

Specyfikację, za pomocą Map Stanów, dużych i złożonych systemów, gdzie niektóre procesy mogą odbywać się niezależnie od siebie, bardzo ułatwia zastosowanie zapisu stanów typu *AND*. Obrazuje on sytuację gdy system może przebywać jednocześnie w kilku stanach ortogonalnych. W poniższym przykładzie z rys. 2.6a są to (**D**,**F**), (**D**,**H**), (**E**,**G**) i in. Przejścia wewnątrz stanów **B** i **C** zachodzą niezależnie od siebie, a więc także mogą zajść równolegle. Z tego powodu ortogonalność łączy się często z równoległością niezależnych procesów. Drugą, obok niezależności, składową pojęcia ortogonalności jest *synchronizacja*. Polega ona na tym, że jedno zdarzenie może wyzwolić więcej niż jedno przejście. Gdy przykładowy system przebywa w stanach (**E**,**H**), to pojawienie się zdarzenia **a** powoduje zajście przejść przeprowadzających system do stanów (**D**,**F**).

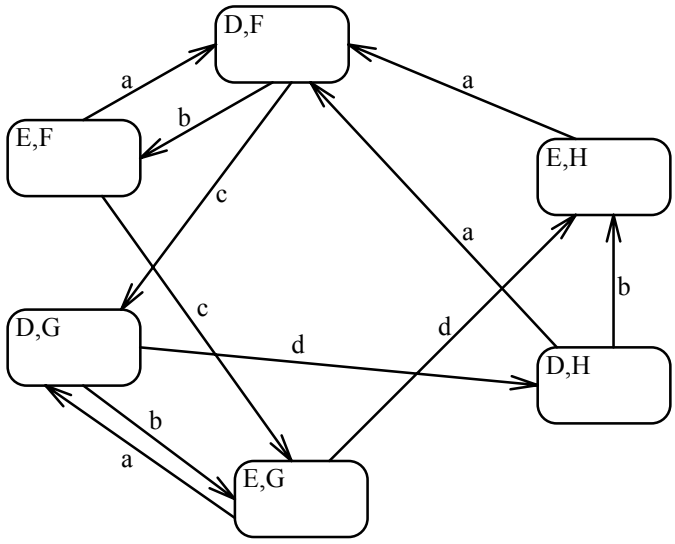


Rys. 2.6a

Ten sam system można przedstawić za pomocą klasycznej płaskiej struktury Diagramów Stanów jak na rys. 2.6b. Każdy zestaw stanów, w którym system może jednocześnie przebywać musimy zamienić na odpowiedni stan w strukturze płaskiej. Powoduje to wykładniczy wzrost ilości stanów. Gdyby poddać takiej konwersji system zawierający w dwóch

składowych ortogonalnych po tysiąc stanów, dałoby to w wyniku milion stanów wynikowych.

Podobnie jak ilość stanów rośnie podczas konwersji ilość przejść. Musimy w wynikowej strukturze umieścić wszystkie przejścia, które dotyczą stanów, z jakich powstały stany wynikowe. Na przykład, ponieważ na rys. 2.6a mamy przejście między stanami **D** i **E** etykietowane zdarzeniem **b**, więc na rys. 2.6b musimy umieścić trzy wyzwalane tym zdarzeniem przejścia ((**D**,**F**)→(**E**,**F**), (**D**,**G**)→(**E**,**G**) i (**D**,**H**)→(**E**,**H**)). Aby nadmiernie nie komplikować rysunku nie zostały na nim umieszczone przejścia wyzwalane dwoma zdarzeniami jak np. $(\mathbf{E},\mathbf{F}) \xrightarrow{\mathbf{a\ and\ c}} (\mathbf{D},\mathbf{G})$, które też powinny się na nim pojawić aby obie specyfikacje były rzeczywiście równoważne. Nawet bez tych dodatkowych (jeszcze pięciu) przejść specyfikacja używająca ortogonalności jest czytelniejsza od płaskiej.



Rys. 2.6b Płaska specyfikacja systemu z rys. 2.6a

2.1.7. Łącznik historii

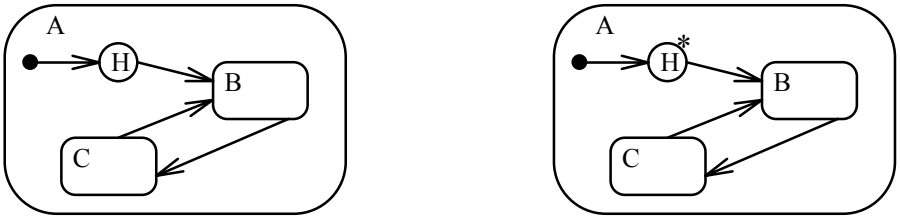
Przy projektowaniu systemów spotykamy się czasem z potrzebą wejścia nie do podstanu domyślnego, lecz do tego, w którym układ przebywał poprzednio. Przykładami mogą być choćby systemy operacyjne z przełączaniem zadań. Aby umożliwić projektantowi dogodne rozwiązanie tego typu prob-

lemu Harel wprowadził do Map Stanów symbol *łącznika historii*. Zaznacza się go jako literę H w kółku umieszczonym na drodze przejścia wprowadzającgo do stanu, którego ów łącznik dotyczy. Znaczenie tego symbolu jest następujące:

— gdy stan osiągany jest po raz pierwszy, to wejście następuje do podstanu domyślnego,

— przy kolejnych wejściach osiągany jest ten podstan, w którym system przebywał przed opuszczeniem stanu z historią.

Opisane zachowanie dotyczy tylko bezpośredniego potomstwa stanu, do którego wejście następuje poprzez opisany łącznik historii. Istnieje także drugi łącznik tzw. *historii głębokiej* dotyczący wszystkich stanów potomnych tego, którego łącznik dotyczy. Zaznacza się go podobnie jak poprzednio omawiany dodając do niego gwiazdkę.



Rys. 2.7 Przykład zaznaczania płytkiej i głębokiej historii

Oba łączniki używane są w niektórych narzędziach wspomagających projektowanie natomiast nie są uwzględniane przy próbach formalnego definiowania składni i semantyki różnych wersji Map Stanów.

2.1.8. Elementy dodatkowe w pakiecie STATEMATE

W pakiecie STATEMATE zastosowano pewne rozszerzenia w stosunku do podstawowej wersji Map Stanów zwiększające ich ekspresyjność.

— Wprowadzone zostały *zmienne* różnych typów, które mogą być używane w wyrażeniach tworzących warunki i które można zmieniać w akcjach.

— Dodano możliwość parametryzowania stanów tworząc *stany ogólne* (ang. *generic*). Stany te pełnią podobną rolę jak podprogramy w proceduralnych

językach programowania. Jeśli różne fragmenty projektowanego systemu mają taką samą strukturę, z dokładnością do parametrów jak nazwy akcji lub zdarzeń, to można je zastąpić stanami ogólnymi definiując wewnętrzną strukturę tylko raz.

— Oprócz przejść wprowadzono także *reakcje*, które nie mają żadnego symbolu graficznego. Są one skojarzone z poszczególnymi stanami poprzez opis tekstowy postaci identycznej z etykietami przejść. Ich znaczenie jest podobne do przejść zaczynających się i kończących w danym stanie lecz podczas wykonywania reakcji nie następuje wyjście ze stanu i powrót do niego.
— Stosowane są dwa alternatywne modele czasu:
  — *synchroniczny*, w którym wykonanie każdego przejścia (lub grupy przejść ortogonalnych) zajmuje jednakowy kwant czasu,
  — *asynchroniczny*, w którym czas trwania przejścia przyjmowany jest za zerowy, a upływ czasu warunkowany jest tylko zdarzeniami zewnętrznymi pojawiającymi się w przypadkowych momentach.

2.1.9. Przyjęte w dalszej części pracy ograniczenia modelu

Ze względu na dość dużą złożoność podstawowego modelu Map Stanów, w bieżącej pracy analizowany będzie model ograniczony.

W etykietach przejść nie będą rozpatrywane warunki. To ograniczenie nie jest istotne gdyż można wprowadzić zdarzenie polegające na spełnieniu danego warunku i w ten sposób go zamodelować.

Akcje będą ograniczone tylko do tych, które generują zdarzenia. Jedynie te akcje mają wpływ na analizę dynamiki projektowanego systemu.

Nie będą rozpatrywane *reakcje* gdyż z dobrym przybliżeniem można je zastąpić przez przejścia zaczynające się i kończące w danym stanie.

Pominięte zostaną zmienne.

Nie będą rozpatrywane stany ogólne. Zostały one wprowadzone w systemie STATEMATE dla wygody projektanta i pominięcie ich nie zubaża ekspresyjności modelu.

Przejścia nie mogą przecinać granic stanów. Oznacza to, że każde przejście musi zaczynać się i kończyć wewnątrz tego samego stanu. Ograniczenie to pozwala na uniknięcie niejednoznaczności określenia priorytetów strukturalnych. Jak wykazał Peron w pracy [51] zawsze można kosztem rozbudowania specyfikacji zamienić Mapę Stanów z przejściami przecinającymi granice stanów na równoważną mapę bez takich przejść.

Rozpatrywany będzie tylko asynchroniczny model czasu.

2.2. Istniejące rozszerzenia związane z czasem

W literaturze dotyczącej przedmiotu omawiane są pewne rozszerzenia Map Stanów dodające do nich określone elementy związane z upływem czasu. Stosuje się je do przeprowadzania analitycznych badań zachowania się projektowanych systemów w czasie. Jako wyjściowy model, do którego dodaje się dodatkowe czynniki, stosowane są podstawowe Mapy Stanów z wyżej wymienionymi ograniczeniami.

2.2.1. Hybrydowe i czasowe Mapy Stanów

Y. Kesten i A. Pnueli zdefiniowali w pracy [38] *Czasowe* i *Hybrydowe Mapy Stanów*. Z każdym przejściem w tym modelu skojarzony jest *przedział czasowy* $\{l,u\}$[1]. Oznacza on czas, w którym przejście musi być aktywowane aby mogło zostać wykonane. Jeżeli z przejściem związany jest przedział [0,0], to nazywamy je *natychmiastowym*. Wszystkie przejścia wyzwalane zdarzeniem muszą być natychmiastowe. Przejścia czasowe mogą być natomiast opatrzone *warunkami*, które muszą być spełnione przez cały czas z przedziału $\{l,u\}$.

Czas życia zdarzeń jest zerowy. Oznacza to, że pojawienie się zdarzenia może odnieść skutek tylko w tym samym momencie. Jeżeli jednak jest możliwe zajście sekwencji przejść bez upływu czasu, to dane zdarzenie może brać udział w wyzwalaniu więcej niż jednego z nich.

Hybrydowe Mapy Stanów mają dodane *zmienne ciągłe*, których wartość może się zmieniać w funkcji czasu. Poszczególne stany w tej wersji mogą mieć skojarzone ze sobą równania różniczkowe zmieniające wartości zmiennych ciągłych. Pozwala to na specyfikowanie systemów hybrydowych — cyfrowo analogowych.

W pracy [38] zdefiniowano dodatkowo tekstowy język opisu Map Stanów *Statext*, który jest równoważny językowi graficznemu.

2.2.2. Czasowe Mapy Stanów

A. Peron i A. Maggiolo–Schettini w artykule [50] wprowadzają inną wersję *Czasowych Map Stanów*. W stosunku do podstawowych Map Stanów mają one dodane opóźnienia przejść, czasy przeterminowania przejść oraz czas życia zdarzeń. Analiza projektowanych systemów prowadzona jest w gęstej dziedzinie czasu określonego nieujemnymi liczbami wymiernymi. Poniżej przedstawione jest krótkie omówienie tych rozszerzeń.

Opóźnienia

Każde przejście może mieć określony w etykiecie czas opóźnienia. Liczony jest on od momentu wejścia systemu do stanu, z którego startuje przejście. Może być ono wykonane, gdy pojawi się wyzwalająca je oferta zdarzeń i nastąpi to po upływie czasu dłuższego od zadanego opóźnienia. Czasy te modelują wykonywanie jakichś czynności w stanie, po zakończeniu których możliwe jest opuszczenie go danym przejściem.

---

[1] Zapis $\{l,u\}$ oznacza odpowiednio otwarte lub domknięte przedziały: $[l,u]$, $[l,u)$, $(l,u]$ lub (l,u).

Czasy przeterminowania

Przejścia mogą mieć podany czas przeterminowania. Podobnie jak przy opóźnieniach liczy się go od momentu wejścia do stanu źródłowego rozpatrywanego przejścia. Omawiany parametr określa czas, po którym dane przejście nie może być już wykonane. Modeluje to sytuację, gdy w stanie wykonywane są jakieś czynności, których wynik się dezaktualizuje po pewnym czasie. Oczywiście po wyjściu ze stanu źródłowego i powrocie do niego czas liczy się od początku. Uwaga ta dotyczy także poprzedniego parametru. Czas przeterminowania musi być dłuższy od czasu opóźnienia. Dopuszczalne jest określenie jego wartości jako nieskończenie wielkiej. Oznacza to brak możliwości przeterminowania przejścia.

Czas życia zdarzeń

Akcje generujące zdarzenia mają określony czas życia wysyłanych zdarzeń. Oznacza to, że mogą one wyzwolić jakieś przejścia tylko przez ten czas. Jeżeli dla żadnego przejścia nie zostanie spełniony warunek czasowy związany z opóźnieniem i przeterminowaniem, wtedy dane zdarzenie jest tracone. Modeluje to sytuację, gdy wynik jakiegoś działania, odbywającego się niekoniecznie w jednym stanie, jest aktualny przez określony czas.

2.2.3. Stochastyczne Mapy Stanów

W artykule [8] wprowadziłem *Stochastyczne Mapy Stanów*. Omawiam je dokładniej w rozdziale 4. Są one modyfikacją Czasowych Map Stanów przedstawionych w pracy [50]. W rozszerzeniu tym przejście wychodzące ze stanu *s* może zajść, jeśli system przebywał w tym stanie przynajmniej przez czas t_{del} będący realizacją zmiennej losowej o dystrybuancie F i nie dłużej niż czas t_{out}. Czas opóźnienia z modelu opisanego w [50] jest więc tutaj realizacją zmiennej losowej o pewnym rozkładzie tworzoną dla każdego przejścia

w momencie wejścia systemu do stanu, z którego ono startuje. Czas t_{out} jest czasem przeterminowania danego przejścia.

Jeżeli w etykiecie przejścia nie podano dystrybuanty zmiennej losowej to jest ono *przejściem natychmiastowym*. Pominięcie specyfikacji t_{out} oznacza przyjęcie jego wartości za nieskończoną.

Podczas realizacji przejścia mogą być wykonywane akcje polegające na generowaniu zdarzeń wewnętrznych. Te zdarzenia mogą mieć zadany *czas życia*, po upływie którego są usuwane ze zbioru zaistniałych zdarzeń. Brak określenia czasu życia zdarzenia oznacza, że jest ono aktywne do następnego kroku. Zdarzenia po wyzwoleniu przejścia znikają nawet, gdy wykonaniu go nie towarzyszył upływ czasu.

Czas rozpatrywany tu, podobnie jak w pracy [50], jest gęsty, a konkretnie wyrażony nieujemnymi liczbami wymiernymi.

2.2.4. Elementy czasu w pakiecie STATEMATE i języku UML

Pakiet STATEMATE rozszerza podstawowe Mapy Stanów o dwa elementy związane z czasem. Pierwszym z nich jest odłożenie w czasie wyzwolenia danego łuku poprzez generację zdarzenia *timeout*(ev,t). Zdarzenie to ma miejsce gdy upłynął czas *t* licząc od ostatniego wystąpienia zdarzenia *ev*. Odliczanie czasu może zostać przerwane gdy nastąpi wyjście ze stanu startowego rozważanego łuku wzdłuż innego łuku. Stanie się tak nawet gdy system powróci do tego stanu. Jeśli w trakcie odliczania czasu wystąpi ponownie zdarzenie *ev*, wtedy czas liczony jest od początku. Modelować możemy w ten sposób działania, których czas trwania nie może przekroczyć zadanej wartości, a także aktywności związane ze stanem, których czas trwania jest zadany.

Akcja *schedule*(ac,t) powoduje odłożenie wykonania akcji *ac* o czas *t* licząc od momentu wykonania akcji *schedule*. Ta akcja jest nieprzerywalna. Odliczanie czasu musi się zakończyć, a akcja *ac* musi zostać wykonana. W ten sposób

można modelować opóźnienia w systemie nie związane z pobytem w jakimś stanie.
W obu przypadkach czas jest podawany jako wyrażenie arytmetyczne, które może zawierać realizacje zmiennych losowych.

Język UML wprowadza nieco inne elementy czasu do modelu Map Stanów. Wprowadzone są tu dwa zdarzenia określające opóźnienie wykonania łuku. Są to *after*(t) oraz *when*(t) różniące się sposobem liczenia czasu. W pierwszym przypadku jest to czas od momentu wejścia do stanu, w drugim — czas bezwzględny liczony od startu systemu. Opóźnienia te nie są związane z żadnym dodatkowym zdarzeniem. Czas podaje się w formie wyrażenia arytmetycznego mogącego zawierać realizacje zmiennych losowych.
Wewnątrz stanów mogą być specyfikowane akcje nie związane z upływem czasu ale także aktywności, których czas trwania jest niezerowy. Nie jest on zadawany wprost lecz zależy od rzeczywistego czasu wykonania aktywności zdefiniowanej inną Mapą Stanów. Aktywność może zostać przerwana wskutek wyjścia ze stanu, w którym jest określona. Po zakończeniu aktywności następuje wyjście ze stanu wzdłuż łuku domyślnego czyli takiego, dla którego nie podano zdarzenia wyzwalającego.
Można specyfikować zdarzenia odłożone wewnątrz stanów. Zdarzenia takie uaktywniają się dopiero po wejściu systemu do stanu, w którym nie są zdefiniowane jako odłożone. Można w ten sposób modelować sytuację gdy pewne wyniki (obliczeń, operacji technologicznych) są dostępne dopiero po wykonaniu pewnej ilości kroków pośrednich. Sytuacja taka występuje na przykład przy przetwarzaniu potokowym lub systolicznym.

2.3. Istniejące narzędzia

Język Map Stanów użyty został w wielu narzędziach wspomagających specyfikację lub projektowanie systemów informatycznych lub automatyki. Obecnie Mapy Stanów są stosowane jako jeden z głównych modeli

opisujących dynamikę i zachowanie się systemów w metodach projektowania obiektowego. Wymienić tu można takie metodologie jak: Booch one [11], OMT czy UML [12],[65]. Także w wielu narzędziach komputerowo wspomaganego projektowania zastosowano model Map Stanów. Należą tu narzędzia do wspomnianej już metodologii projektowania obiektowego UML (Rational Rose, select Enterprise, STP UML, Rapsody). Także takie pakiety jak "Better state" czy STATEMATE stosują jako jeden z głównych modeli Mapy Stanów.

3. Konwersja Map Stanów w Kolorowane Sieci Petriego

3.1. Wprowadzenie

Przy badaniu różnych aspektów jakości, w tym także wydajnościowych, projektowanych lub istniejących systemów często stosuje się sieci Petriego. Rozwinięta jest bogata teoria tych sieci oraz metody ich analizy [48], [52], [55]. Dla różnych rozszerzeń czasowych sieci Petriego istnieją analityczne metody badania wydajności opisywanych układów. M. Ajmone-Marsan i in. opisują klasę uogólnionych stochastycznych sieci Petriego [1]. Zastosowano w nich dwa rodzaje przejść:

— czasowe o czasie odpalania opisanym zmienną losową o rozkładzie wykładniczym,

— natychmiastowe o zerowym czasie odpalania.

Do analizy wydajności stosowana jest w nich teoria procesów i łańcuchów Markowa [57].

W opisywanych poniżej badaniach przyjąłem topologiczny model kolorowanej sieci Petriego (ang. Coloured Petri Net, CPN), opisanej przez K. Jensena [36]. Umożliwia to badanie w różnych aspektach systemu specyfikowanego za pomocą Map Stanów . Rozwinięta jest dla sieci Petriego teoria pozwalająca badać osiągalność stanów, a także występowanie blokad czy zakleszczeń.

Wyboru rodzaju sieci Petriego dokonano ze względu na łatwość opisu pewnych zachowań Map Stanów w tej klasie sieci. Szczególnie interesującymi własnościami sieci kolorowanych są: możliwość współistnienia różnych rodzajów znaczników (różnych kolorów) oraz możliwość wprowadzenia hierarchii, która nie zmieniając semantyki sieci (jak udowodniono w [32]), często znacznie upraszcza zapis.

Wprowadzając czas do modelu sieci Petriego (bez zmiany topologii badanego układu) zyskujemy możliwość badania maksymalnych czasów

odpowiedzi systemu (stosując sieci czasowe) oraz średnich czasów odpowiedzi i przepustowości systemu (używając sieci stochastycznych). Ponieważ w pracy zajmuję się aspektami wydajnościowymi, więc dalej rozpatrywać będę tylko sieci stochastyczne. Są nimi Regularne Stochastyczne Sieci Petriego wprowadzone przez C. Dutheilleta i S. Haddada w pracy [18] i będące zastosowaniem do sieci kolorowanych rozszerzeń wprowadzonych przez M. Ajmone-Marsana i in. w artykule [1].

Wyniki przedstawione w tym rozdziale były częściowo publikowane w pracy [7].

3.2. Konwersja topologiczna

3.2.1. Poddawane konwersji Mapy Stanów

Omawiane w tym rozdziale Mapy Stanów są bogatszym podzbiorem Map Stanów opisanych przez Harela i Naamada [24] niż ma to miejsce w pozostałych rozdziałach. Dopuszczalne są tu przejścia przecinające granice stanów oraz różne łączniki przejść, w tym łącznik historii.

3.2.2. Stosowane Sieci Petriego

Kolorowane sieci Petriego są zgodne ze specyfikacją zawartą w [36]. Formalnie sieć taka jest ciągiem:

CPN= (Σ, P, T, A, N, C, G, E, IN),

gdzie:

— Σ jest zbiorem *palet* kolorów,

— P jest zbiorem *miejsc*,

— T jest zbiorem *przejść*,

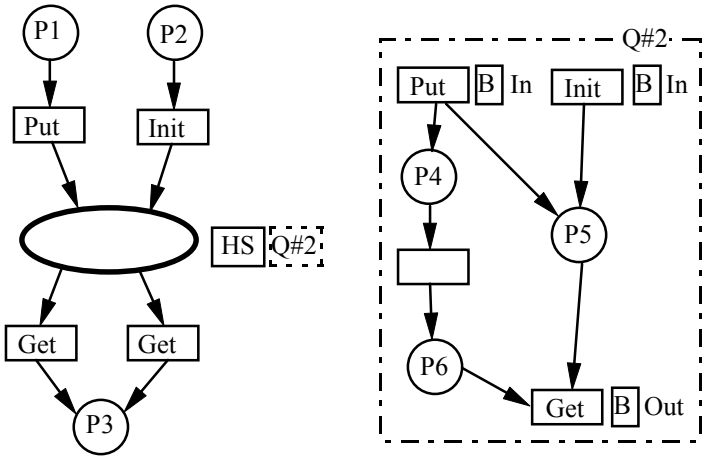
— A jest zbiorem *łuków*,

— N jest funkcją, która przyporządkowuje każdemu łukowi parę miejsce-przejście lub przejście-miejsce, z którymi jest on incydentny,

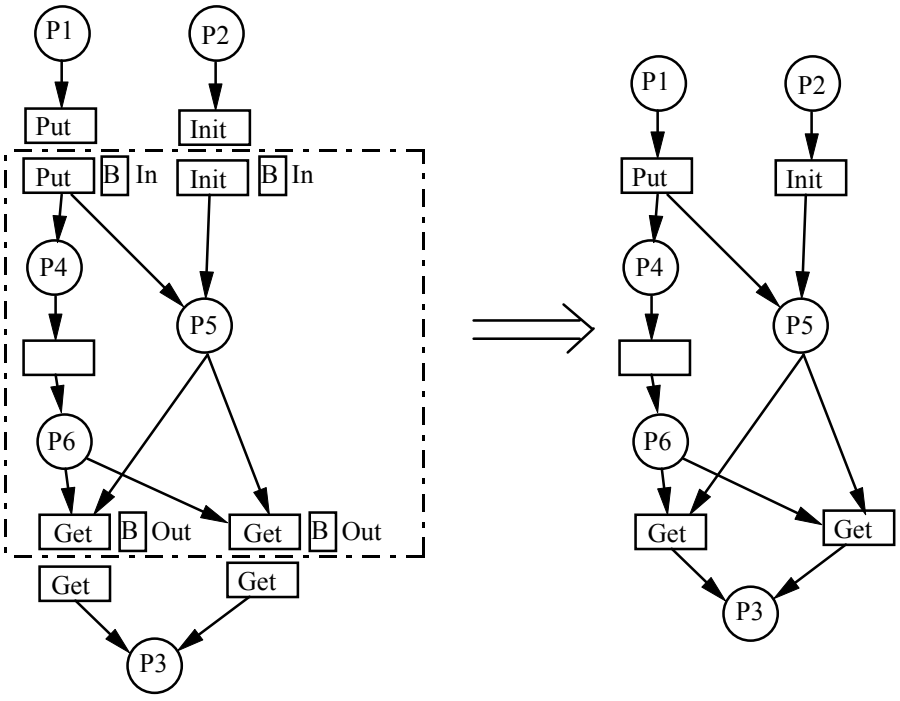
— C jest funkcją przyporządkowującą każdemu miejscu paletę kolorów znaczników, które można umieszczać w tym miejscu,
— G jest funkcją przyporządkowującą każdemu przejściu wyrażenie logiczne (*dozór*),
— E jest funkcją przypisującą każdemu łukowi wyrażenie określające wielozbiór znaczników przenoszonych przez dany łuk,
— IN jest funkcją przyporządkowującą każdemu miejscu wyrażenie określające wielozbiór znaczników znajdujących się w tym miejscu w znakowaniu początkowym.

W dalszej części rozdziału zastosowano dwie palety kolorów oznaczone jako **T** oraz **E**. Paleta **T** składa się z kolorów **w** (≈ work) oraz **h** (≈ history). Paleta **E** zawiera nazwy wszystkich zdarzeń, które mogą się pojawić w systemie. Zadeklarowano także zmienne $\mathbf{t_i}$ jako wielozbiory palety **T** oraz zmienną **e** będącą wielozbiorem palety **E**. W przykładach niektórych konwersji zastosowano rozszerzenia hierarchiczne Kolorowanych Sieci Petriego wprowadzone i szczegółowo opisane w [32]. Formalna definicja tych rozszerzeń jest dość skomplikowana i nie będzie tu przytoczona. Wyjaśnienia wymagają takie pojęcia jak *fuzja* oraz *hierarchiczna substytucja* (ang. *Hierarchy Substitution*). W przykładach zastosowano *fuzje globalne* oznaczane na rysunkach jako **FG *nazwa***. Oznaczają one, że wszystkie miejsca albo przejścia oznaczone tym symbolem z taką samą nazwą fuzji są jednym miejscem albo jednym przejściem, a łuki dochodzące lub wychodzące z niego są sumą mnogościową łuków dochodzących lub wychodzących z miejsc lub przejść należących do fuzji. Drugim zastosowanym rozszerzeniem jest hierarchiczna substytucja miejsca oznaczona jako **HS *nazwa***. Oznaczone tym symbolem miejsce, zwane *miejscem zastępczym* (ang. *substitution place*), jest rozrysowywane szczegółowo jako osobna podsieć. Samo miejsce zastępcze jest skrótowym oznaczeniem tej szczegółowej podsieci, którą projektant chce ukryć na wyższym poziomie hierarchii. Autorzy tego rozszerzenia ([32]) objaśniają jego znaczenie przez analogię do abstrakcyjnych

typów danych, przy czym szczegółowa podsieć oznacza konkretny typ danych czyli realizację typu abstrakcyjnego. Przejścia sąsiednie do miejsca zastępczego nazywane są *przejściami–gniazdkami* (ang. *socket transition*) przez analogię do gniazdek w układach elektronicznych. Spełniają one podobną rolę jak parametry aktualne funkcji w klasycznych proceduralnych językach programowania lub jak podstawka pod układ scalony w układach elektronicznych. Na arkuszu uszczegóławiającym miejsce zastępcze znajdują się *przejścia–porty* (ang. *port transition*) opatrzone symbolem **B In**, **B IO** lub **B Out**. Są one przy spłaszczaniu struktury hierarchicznej sklejane z przejściami–gniazdkami. Odwołując się znów do podanych przez autorów analogii, można te przejścia przyrównać do parametrów formalnych funkcji znanych z proceduralnych języków programowania lub do nóżek układów scalonych. Rysunek 3.1 pokazuje przykład zastosowania miejsca zastępczego (po lewej) i jego uszczegółowienia (po prawej) natomiast na rysunku 3.2 przedstawiony jest sposób zamiany tej konstrukcji w równoważną płaską strukturę.



Rys. 3.1 Miejsce zastępcze



Rys. 3.2 Semantyka miejsca zastępczego

Dla zwiększenia przejrzystości rysunków tam, gdzie nie są konieczne, pominięto znakowania początkowe. Pominięto także wyrażenia przy tych łukach, które powinny być etykietowane jako **1'(w)**.

3.2.3. Zdarzenia i akcje

Przyjmijmy, że łuki oznaczające przejścia w Mapach Stanów etykietowane są dwoma ciągami nazw zdarzeń oddzielonymi od siebie ukośną kreską. Pierwszy ciąg zawiera zdarzenia, które muszą zajść aby dane przejście zostało wyzwolone. Niektóre wersje Map Stanów oprócz zdarzeń mają jeszcze, ujęte w nawiasy kwadratowe, *warunki*, jednak jak pokazano w [24], wyrażenia **zdarzenie** oraz **[warunek]** mogą być traktowane tak samo z punktu widzenia składni map stanów. Z tego powodu, dla uproszczenia zapisu, w dalszej części używane będą wyłącznie zdarzenia.

Drugi ciąg nazw występujący w etykiecie zawiera akcje. Są one wykonywane równocześnie podczas wykonywania przejścia. Akcje generują zdarzenia o takich samych nazwach, które będą dostępne w kolejnym mikrokroku.

Dla zamodelowania w Sieci Petriego mechanizmów związanych z generacją, rozpowszechnianiem i wykorzystaniem zdarzeń wprowadzono paletę **E** kolorów będących nazwami wszystkich zdarzeń występujących w specyfikowanym systemie. Wprowadzono także dwa dodatkowe miejsca przechowujące znaczniki o tych kolorach. Jedno z nich oznaczone jako **EO** zawiera ofertę zdarzeń dla danego mikrokroku, drugie — **PE** — zdarzenia w danym mikrokroku wygenerowane. Oba te miejsca muszą być widoczne na wszystkich poziomach hierarchii, a więc należy utworzyć fuzje globalne dla każdego z nich. Znaczniki odpowiadające danemu zdarzeniu generowane są w takiej ilości aby mogły być wykorzystane wszędzie tam, gdzie są potrzebne. Liczbę tę można określić jako *maksymalny stopień ortogonalności Mapy Stanów*. Definiujemy go jako maksymalny stopień ortogonalności korzenia Mapy Stanów, a ten określamy poniższym wzorem podstawiając korzeń za *s*:

$$\mathrm{N}(s) = \begin{cases} 1 & gdy\ typ(s) = PRIM \\ \sum_{i=1}^{n} \mathrm{N}(s_i) & gdy\ typ(s) = AND \\ \max(\mathrm{N}(s_i)) & gdy\ typ(s) = XOR \end{cases}$$

gdzie:

— n jest liczbą podstanów danego stanu,

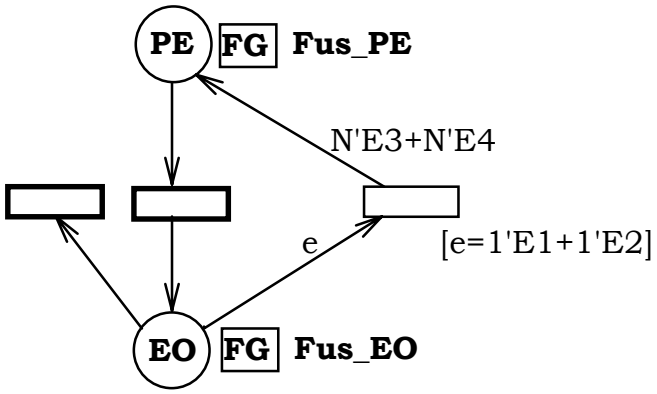
— s_i dla i=1,…,n jest oznaczeniem kolejnych podstanów.

W momencie gdy przy danej ofercie zdarzeń nie jest już możliwe żadne przejście, usuwamy wszystkie pozostałe w miejscu **EO** znaczniki, a następnie przepisujemy zawartość miejsca **PE** do **EO**. Modelujemy w ten sposób mikrokroki Mapy Stanów. Nie określamy tu szczegółów tych operacji.

W Mapach Stanów przejścia zaczynające się w nadstanach mają pierwszeństwo przed przejściami wychodzącymi z podstanów. Dla zamodelowania tego mechanizmu wprowadźmy priorytety przejść w sieci Petriego. Nie zwiększa to wprawdzie ekspresyjności Kolorowanych Hierarchicznych Sieci Petriego, w których priorytety można zamodelować w inny sposób, jednak

zdecydowano się na takie rozwiązanie ze względu na prostotę i elegancję zapisu.

Na rys. 3.3 przedstawiono konwersję przejścia etykietowanego **E1**,**E2/E3**,**E4**. Zamieniane jest ono na przejście wyprowadzające z miejsca **EO** po jednym znaczniku odpowiadającym zdarzeniom **E1** i **E2** oraz wprowadzające do miejsca **PE** po N znaczników odpowiadających zdarzeniom **E3** i **E4** gdzie N jest określonym wyżej maksymalnym stopniem ortogonalności. Na rysunku przedstawiono także symbolicznie wspomniane wyżej przejścia modelujące mikrokroki.



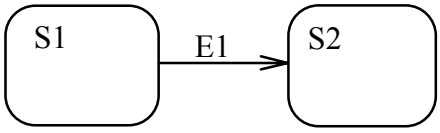
Rys. 3.3 Generacja i wykorzystanie zdarzeń

W dalszej części rozdziału nie będą przedstawiane miejsca **PE** i **EO** ani sąsiednie do nich łuki. Pozostawione natomiast będą dozory przejść zapisane dla przejrzystości rysunków w skrótowej formie. Zamiast pełnego zapisu "**[e=1'E1]**" będzie występowało samo "**[E1]**" natomiast wyrażenia typu "**[e=1'E1+1'E2]**" zastąpione będą iloczynem logicznym zdarzeń: "**[E1 & E2]**".

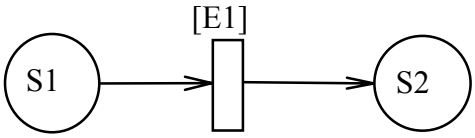
3.2.4. Stany na jednym poziomie hierarchii

Podstawową konstrukcją w języku Map Stanów jest przejście pomiędzy dwoma równorzędnymi (czyli znajdującymi się na jednym poziomie hierarchii) stanami. Przedstawione jest ono na rys. 3.4. Układ opisany tą, bardzo prostą, siecią przechodzi ze stanu **S1** do stanu **S2** pod wpływem

zdarzenia **E1**. Powyższe przejście w sieci Petriego może być opisane w sposób przedstawiony na rys. 3.5.



Rys. 3.4

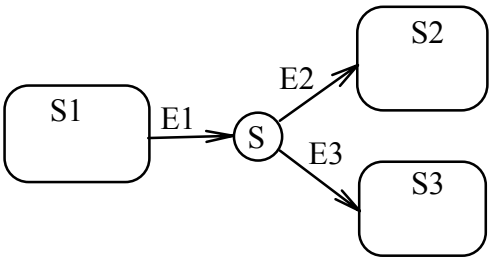


Rys. 3.5

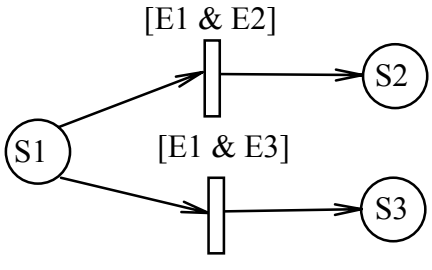
Zastosowano na nim analogiczne oznaczenia: miejsce **Si** odpowiada stanowi **Si**. Zdarzenie **E1** zastąpiono tu dozorem przejścia. Fakt przebywania w stanie **Si** modelowany jest za pomocą znacznika o kolorze **w** przebywającego w miejscu **Si**.

3.2.5. Łącznik zdarzeń

Często wykorzystywaną w Mapach Stanów konstrukcją jest zastosowanie łącznika zdarzeń lub warunków. Oba łączniki rozpatrywane są tu wspólnie ze względu na taką samą ich topologię i semantykę. Konwersji dokonujemy tu w następujący sposób. Każdy stan **Si** zamieniamy na odpowiadające mu miejsce **Si**. Następnie wyznaczamy wszystkie pełne przejścia złożone (Full Compound Transition [24]) łączące poszczególne stany.



Rys. 3.6



Rys. 3.7

Każde z nich zamieniamy na jedno przejście w sieci Petriego. Opatrujemy je dozorem będącym iloczynem logicznym zdarzeń znajdujących się wzdłuż

danego przejścia złożonego w Mapie Stanów. Przykładowa konwersja przedstawiona jest na rys. 3.6 i rys. 3.7.

3.2.6. Hierarchia stanów XOR

Hierarchia stanów *XOR* w Mapach Stanów pozwala na dekompozycję projektowanego automatu na różne poziomy szczegółowości. Pozwala uzyskać przejrzysty opis złożonego systemu. Z tego powodu jest ona jedną z podstawowych i często używanych konstrukcji w Mapach Stanów. Z konstrukcją tą związane jest kilka elementów podlegających konwersji:

a) przejście z nadstanu,

b) przejście z jednego z podstanów,

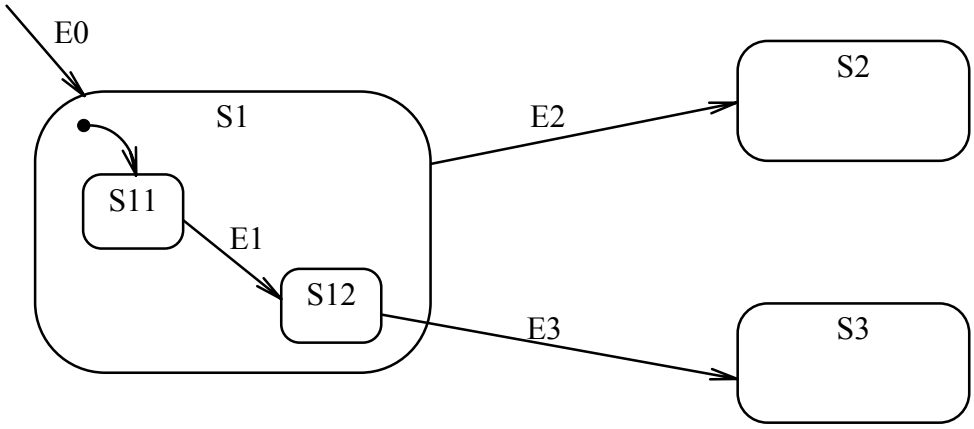
c) przejście do nadstanu.

Postępowanie przy konwersji poszczególnych elementów przedstawione będzie na przykładzie (rys. 3.8 i rys. 3.9). W sieci Petriego zastosowano składnię miejsc zastępczych (ang. Substitution Places) w odniesieniu do **S1**.

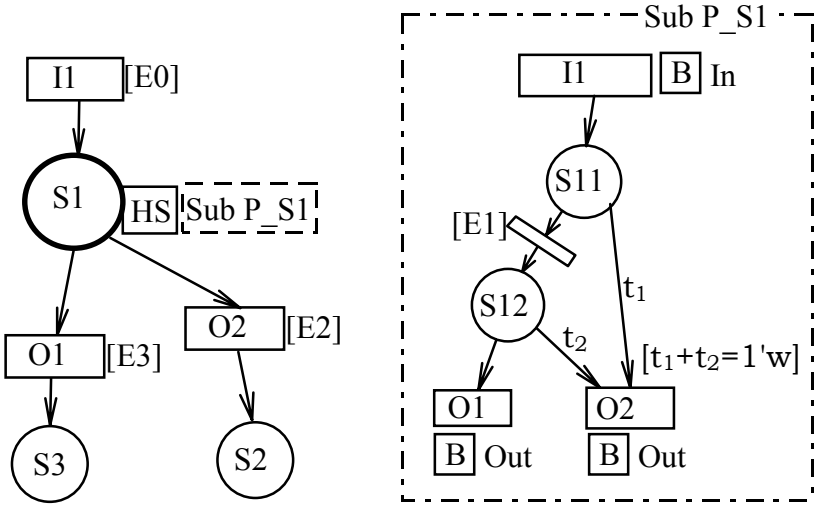
Łuk etykietowany jako **E0** wraz z przejściem domyślnym tworzy pełne przejście złożone do stanu **S11**. Odwzorowujemy je w sieci Petriego za pomocą przejścia oznaczonego na rys. 3.9 jako **I1**, które jest przejściem–gniazdkiem sąsiednim do miejsca **S1**. Na stronie **P_S1** wyprowadzamy z przejścia–portu **I1** łuk do miejsca **S11**. Analogicznie postępujemy, jeśli w Mapie Stanów mamy przejście do któregoś ze stanów podrzędnych. Odpowiednie dozory przejść w sieci Petriego umieszczamy tylko na stronie nadrzędnej.

Przejście ze stanu podrzędnego oznaczone jest w przykładzie jako **E3**. Transformacji dokonujemy analogicznie jak w przypadku przejść do stanów podrzędnych. Różnica polega tylko na zastosowaniu przejść–gniazdek wychodzących z miejsca zastępczego, a nie wchodzących. W omawianym przykładzie jest to przejście **O1**.

Przejście etykietowane **E2** zaczyna się w stanie nadrzędnym. Oznacza więc wyjście z dowolnego ze stanów podrzędnych. W sieci Petriego zastępujemy je przejściem–gniazdkiem (tu **O2**) sąsiednim do miejsca zastępczego **S1** opatrzonym dozorem **E2**. Na stronie uszczegóławiającej miejsce **S1** umieszczamy łuk z każdego miejsca na tej stronie do przejścia–gniazdka **O2**. Wyrażenia tych łuków zawierają zmienne $\mathbf{t_i}$ zadeklarowane jako wielozbiory palety **T**. Dozór przejścia–portu **O2** zapewnia odpalenie go gdy w jednym z miejsc **S11** lub **S12** znajdzie się znacznik o kolorze **w**.



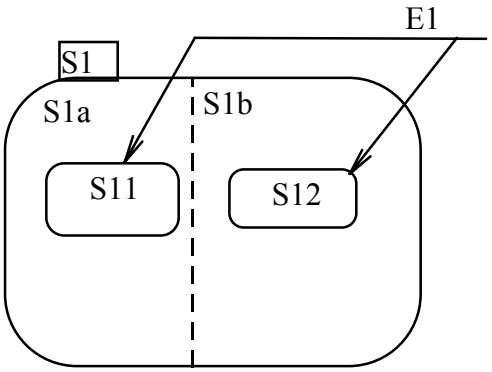
Rys. 3.8



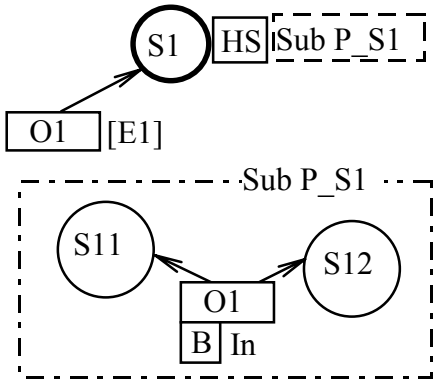
Rys. 3.9

3.2.7. Wejście do stanów ortogonalnych

W przypadku jednoczesnego wejścia do wszystkich stanów parami ortogonalnych, jak na rys. 3.10, zastępujemy stany podrzędne odpowiednimi miejscami. Stan nadrzędny staje się miejscem zastępczym. Bezpośrednich podstanów stanu typu *AND* nie modelujemy w sieci Petriego gdyż w Mapach Stanów nie są dopuszczalne przejścia z ani do takich stanów. Z przejścia-portu odpowiadającego przekształcanemu wyprowadzamy tyle łuków, ile było stanów ortogonalnych i doprowadzamy je do odpowiednich miejsc. Przykład takiej konwersji pokazany jest na rys. 3.11.

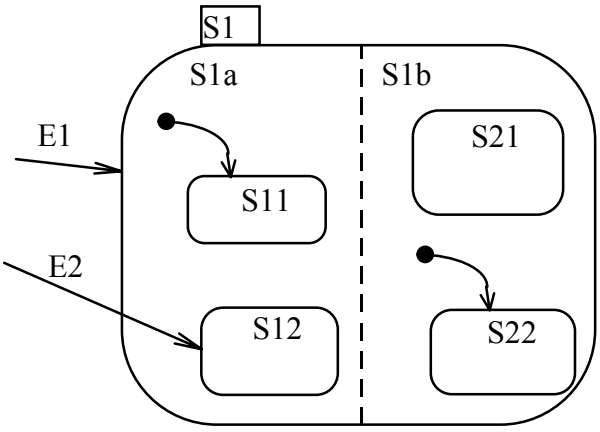


Rys. 3.10

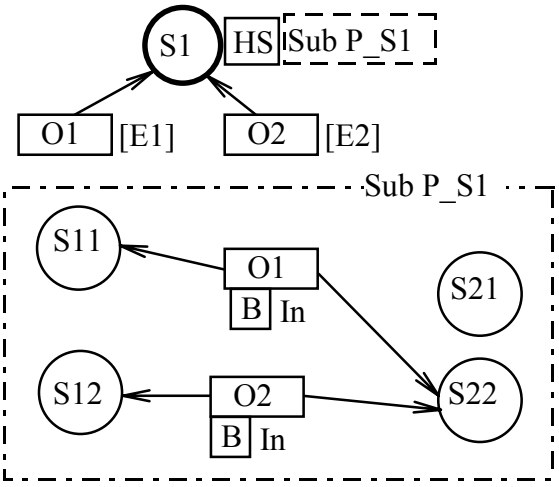


Rys. 3.11

Nieco bardziej skomplikowana sytuacja występuje, gdy mamy przekształcić wejście do stanu nadrzędnego lub do części stanów podrzędnych.



Rys. 3.12



Rys. 3.13

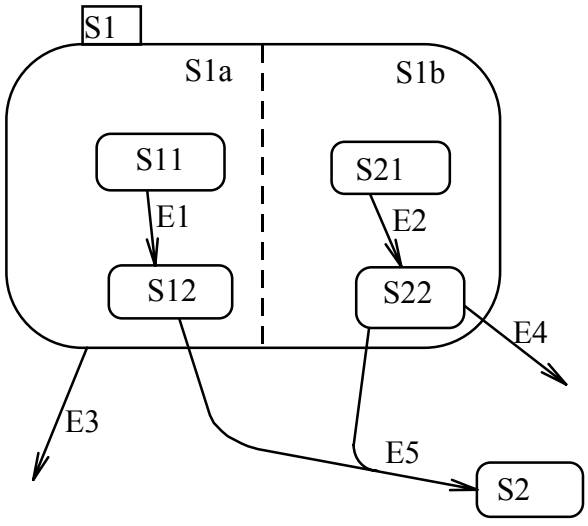
Zgodnie z semantyką Map Stanów podaną w [24] przejścia takie musimy uzupełnić odpowiednimi przejściami domyślnymi. Po tej czynności uzyskujemy sytuację analogiczną do poprzednio omówionej, czyli przejścia do wszystkich podstanów parami ortogonalnych. Przykład opisywanego przekształcenia znajduje się na rys. 3.12 i rys. 3.13. Po uzupełnieniu przejść uzyskujemy sytuację następującą:

E1→ {**S11 S22**}, natomiast **E2**→ {**S12 S22**}.

3.2.8. Wyjście ze stanów ortogonalnych

Możliwe są trzy rodzaje wyjść ze stanów równoległych. Na rys. 3.14 pokazane są przykłady każdego z nich. Przejście wyzwalane zdarzeniem **E3** jest wyjściem z nadstanu, **E4** jest wyjściem z części podstanów ortogonalnych podczas gdy przejście wyzwalane zdarzeniem **E5** jest wyjściem ze wszystkich podstanów ortogonalnych.

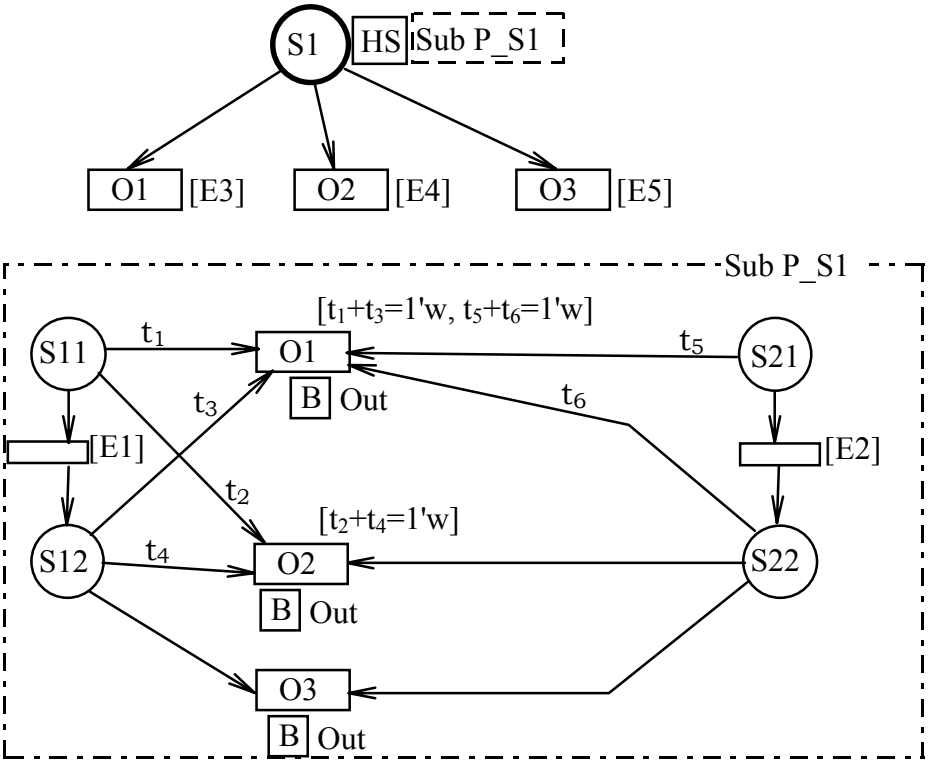
Przystępując do konwersji tych przejść najpierw tworzymy miejsce zastępcze odpowiadające nadstanowi z Mapy Stanów i wyprowadzamy z niego odpowiednie przejścia opatrując je dozorami. Dalszy tok postępowania pokazany będzie na przykładzie.



Rys. 3.14

Przejście ze wszystkich stanów ortogonalnych przekształcane jest na przejścia–porty wychodzące ze wszystkich miejsc odpowiadających stanom ortogonalnym, czyli w przykładzie, **S12** i **S22**. Na stronie nadrzędnej połączone są do przejścia-gniazdka oznaczonego tutaj symbolem **O3** i opatrzonego dozorem **[E5]**.

Kolejną możliwością wyjścia ze stanów równoległych jest przejście z części stanów podrzędnych lub ze stanu nadrzędnego. Przykłady takich przejść także znajdują się na rys. 3.14. Dla każdego podstanu będącego startowym rozważanego przejścia tworzymy w sieci Petriego odpowiadające mu miejsce i jeden łuk do przejścia-portu wyprowadzający z niego znacznik.

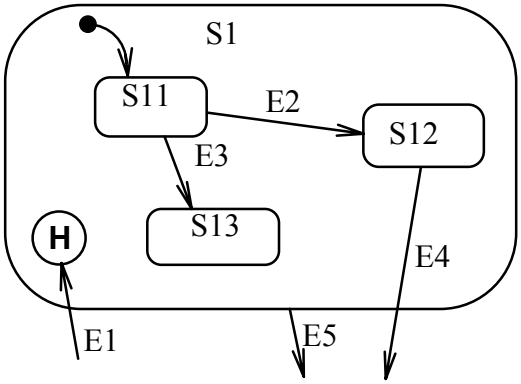


Rys. 3.15

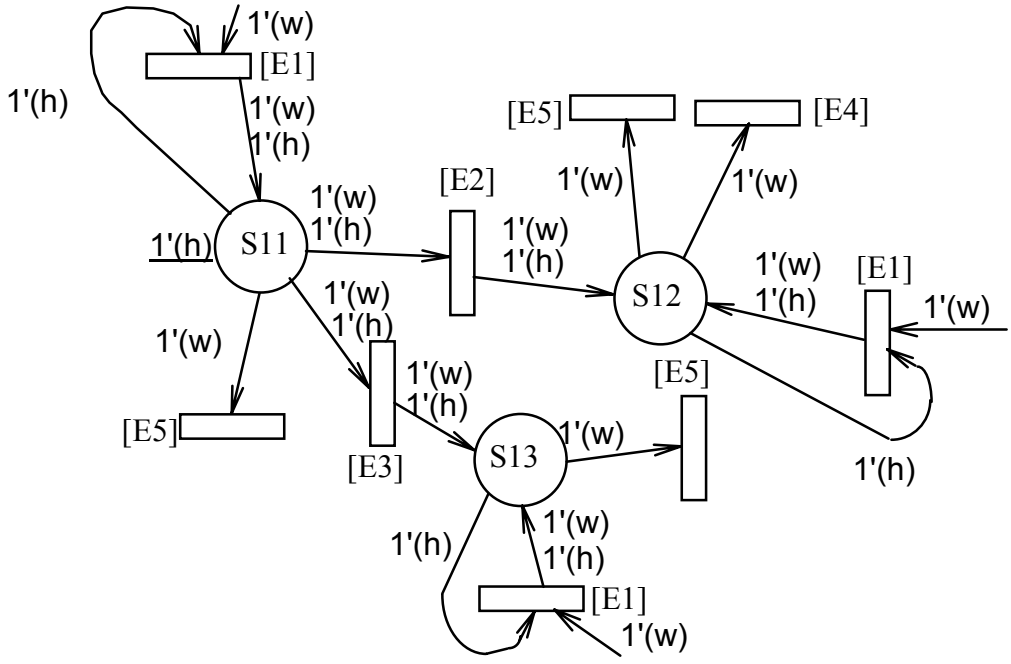
Z każdego miejsca odpowiadającego podstanom pozostałych komponentów ortogonalnych wyprowadzamy do tego przejścia łuk etykietowany zmienną $\mathbf{t_i}$. Samo przejście-gniazdko natomiast opatrujemy dozorem wymagającym aby suma znaczników wyprowadzanych z miejsc modelujących podstany każdego komponentu ortogonalnego była równa jeden. Wynik przekształcenia przykładowej Mapy Stanów pokazano na rys. 3.15.

3.2.9. Hierarchia z historią stanów

Ważną konstrukcją języka diagramów Harela jest wejście do stanu z hierarchią za pośrednictwem łącznika historii. Umożliwia ona powrót do stanu, w którym układ, opisywany Mapą Stanów, znajdował się przed opuszczeniem stanu nadrzędnego. Stan zawierający łącznik historii modelujemy w kolorowanej sieci Petriego z użyciem znaczników o dwóch kolorach. Jeden z nich określa stan układu, a drugi jego pamięć.



Rys. 3.16



Rys. 3.17

Przy wychodzeniu poza stan nadrzędny, zabierany jest tylko znacznik o kolorze **w** natomiast znacznik o kolorze **h** zawsze pozostaje wewnątrz stanu nadrzędnego, czyli w którymś z miejsc modelujących stany podrzędne.

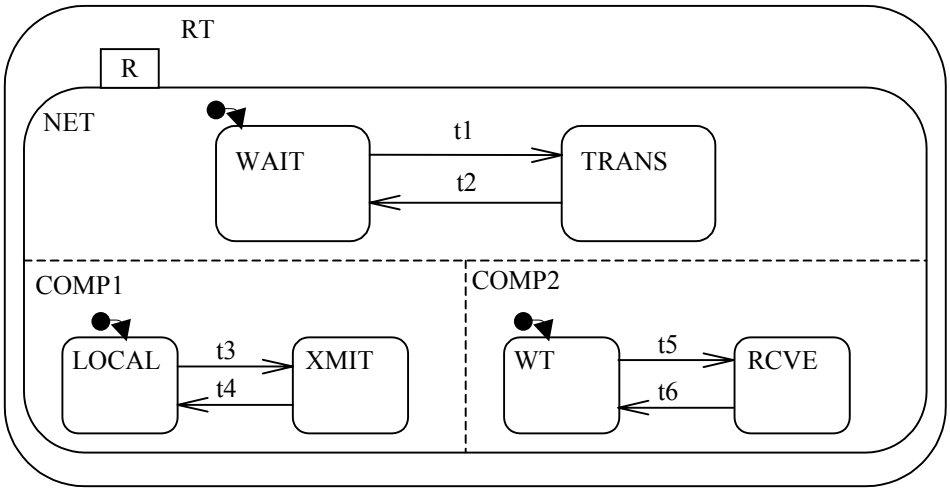
Przejście domyślne modelowane jest poprzez znakowanie początkowe (w podanym przykładzie jest to umieszczenie znacznika **h** w miejscu **S11**). Pełen przykład transformacji opisywanej tu konstrukcji znajduje się na rys. 3.16 i rys. 3.17.

3.3. Elementy związane z czasem

W poprzednim podrozdziale opisano propozycję sposobu, w jaki należy przekształcać topologię Map Stanów w strukturę sieci Petriego. Jest to sposób ogólny mający zastosowanie tak dla Map Stanów bez czasu, jak i z wprowadzonymi opóźnieniami określonymi deterministycznie lub stochastycznie. Przekształcane są one w odpowiednie rodzaje sieci Petriego. Regularne Stochastyczne Sieci Petriego (*RSPN*) opisane w pracy [18] mają dwa rodzaje przejść. Jedne z nich, natychmiastowe, mają zerowy czas odpalania, natomiast drugie, czasowe charakteryzują się czasem odpalania opisanym zmienną losową o rozkładzie wykładniczym. Stosując opisaną w poprzednim rozdziale konwersję można zastosować je do wyjściowych Map Stanów. Rozpatrywać wtedy możemy model Map Stanów o dwóch rodzajach przejść:

— natychmiastowych o czasie wykonania równym zero oraz
— czasowych o czasie wykonania opisanym zmienną losową o rozkładzie wykładniczym z parametrem λ.

Wszystkie przejścia natychmiastowe musimy zamienić na takie same w sieci Petriego, a wszystkie czasowe, na również czasowe o identycznych parametrach jak w wyjściowej Mapie Stanów. Wprowadźmy jeszcze jedno przekształcenie aby umożliwić modelowanie dynamiki związanej ze zdarzeniami zewnętrznymi. Przyjmujemy, że zdarzenia te pojawiają się w momentach opisanych zmiennymi losowymi o rozkładach wykładniczych. Dodajemy więc miejsce **EE** i dla każdego zdarzenia zewnętrznego, przejście czasowe do miejsca **EO**. Ponadto za każdym razem, gdy znacznika odpowiadającego danemu zdarzeniu nie ma ani w miejscu **EE**, ani w **EO**, wkładany jest znacznik do miejsca **EE** umożliwiając regenerację tego zdarzenia.



t1:srq t2:,5/rrq;sack

t3:send/srq t4:sack

t5:rrq t6:,1

Rys. 3.18 Mapa Stanów opisująca system transmisji

Na rysunku 3.18 przedstawiony jest przykład prostego systemu transmisji zapisanego za pomocą Map Stanów. Etykiety przejść zapisane są w następującym formacie: *trig*, λ/*act*

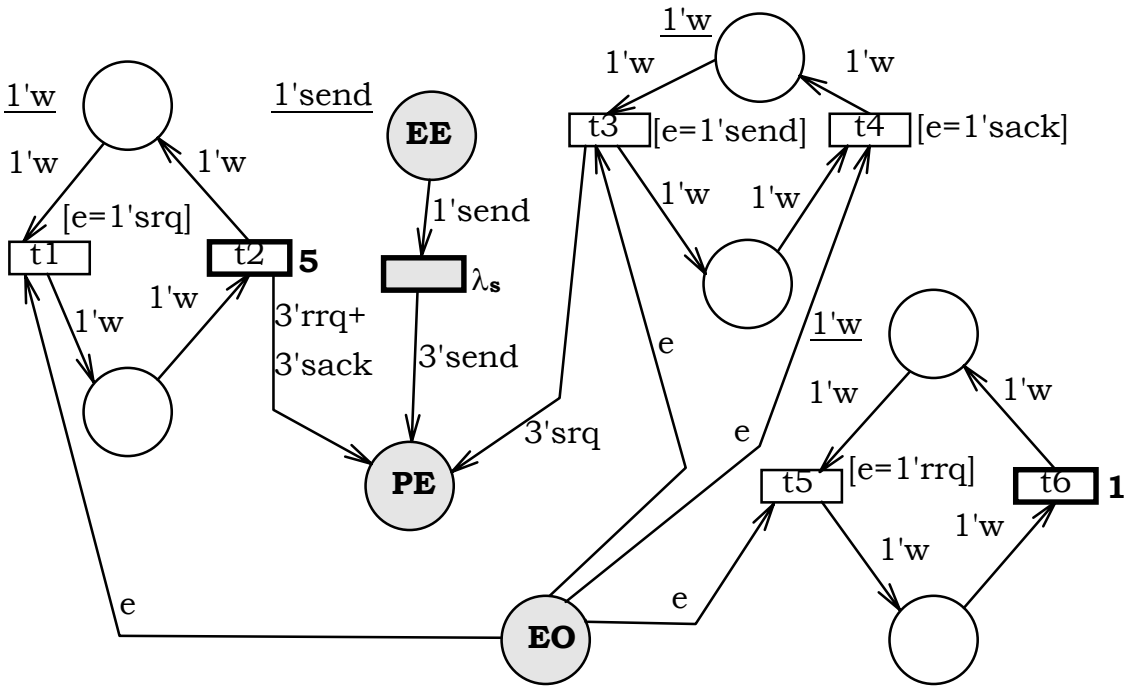
gdzie *trig* oznacza zdarzenia wyzwalające,

λ jest parametrem zmiennej losowej opisującej opóźnienie,

act jest listą generowanych zdarzeń.

Każdy element etykiety można pominąć.

Konwersja w Sieć Petriego przedstawiona jest na rys. 3.19. Zastosowano płaską strukturę sieci, bez hierarchii, gdyż w tym przypadku daje to czytelniejszy rysunek. Przejścia zaznaczone cienką linią są natychmiastowe, grubą — czasowe. Szare elementy sieci są związane z generacją zdarzeń. Pominięto elementy związane z regeneracją zdarzeń zewnętrznych oraz symulujące mikrokroki jak opisano w rozdziale 3.2.3. Maksymalny stopień ortogonalności modelowanej Mapy Stanów wynosi 3, dlatego dla każdego zdarzenia generowane są po trzy znaczniki.



Rys. 3.19 System transmisji opisany za pomocą *RSPN*

3.4. Wnioski

Mapy Stanów są wygodnym językiem specyfikacji systemów reaktywnych. Sieci Petriego umożliwiają analityczne podejście do badania jakości projektowanych systemów. Z ich pomocą badać można różne aspekty opracowywanych systemów. Rozwinięta jest teoria dotycząca badań nad osiągalnością stanów oraz istnieniem zakleszczeń. Są to bardzo ważne zagadnienia przy projektowaniu układów automatyki czy systemów produkcyjnych. Rozszerzenia czasowe Sieci Petriego umożliwiają badanie czasów odpowiedzi systemu na pobudzenia co jest bardzo istotne w systemach czasu rzeczywistego. W niektórych systemach ważna jest ich wydajność. Tutaj też można zastosować teorię Sieci Petriego, a konkretnie ich stochastycznych rozszerzeń, zwłaszcza Regularnych Stochastycznych Sieci Petriego.

Istnieje jeszcze inny niż Kolorowane Hierarchiczne Sieci Petriego model rozszerzonych Sieci Petriego zwany Place Chart Nets opisany w pozycji [39] dopuszczający stosowanie podobnej jak w Mapach Stanów hierarchii. Zamiana Map Stanów na ten formalizm wydaje się być łatwiejsza. Jego

twórcy sugerują nawet modelowanie Map Stanów za pomocą Place Chart Nets. Autorowi nie jest jednak znany dowód równoważności tego modelu z Kolorowanymi Sieciami Petriego ani żadne jego stochastyczne rozszerzenie, które umożliwiałoby badanie wydajności systemów. Dokonywano prób konwersji Map Stanów w pewną Sieć Petriego w pracy [19]. Nie jest tam jednak jasno określone jaki to rodzaj sieci (nazywana jest ona Kolorowaną Siecią Petriego). Wydaje się ona być połączeniem idei Kolorowanych Hierarchicznych Sieci Petriego z Place Chart Nets.

Postępowanie opisane w rozdziale ma pewną niedogodność związaną z koniecznością dokonywania konwersji Map Stanów w inny formalizm. Prawdopodobnie krok ten udałoby się zautomatyzować, jednak bardziej atrakcyjny wydaje się inny kierunek badań polegający na bezpośredniej formalizacji składni i semantyki rozszerzonych o elementy czasu Map Stanów. Przedstawiony będzie on w kolejnych podrozdziałach.

4. Stochastyczne Mapy Stanów

4.1. Wprowadzenie

W poprzednim podrozdziale przedstawiona została propozycja podejścia do analizy wydajności systemów opisywanych Mapami Stanów poprzez ich konwersję do pewnej klasy sieci Petriego. Teraz pokazany będzie projekt bardziej bezpośredniego podejścia. Prowadzenie bezpośrednich teoretycznych analiz systemów specyfikowanych za pomocą Map Stanów wymaga uprzedniego zdefiniowania ich formalnej semantyki. Różne warianty semantyk zostały przedstawione w pracach [25], [43], [51], [50], [54], [66]. W artykułach [43], [50] wprowadzone jest czasowe rozszerzenie Map Stanów. Polega ono na rozpatrywaniu gęstej dziedziny czasu, dodaniu opóźnień i czasów przeterminowania do przejść oraz przyjęciu niezerowego czasu życia zdarzeń.

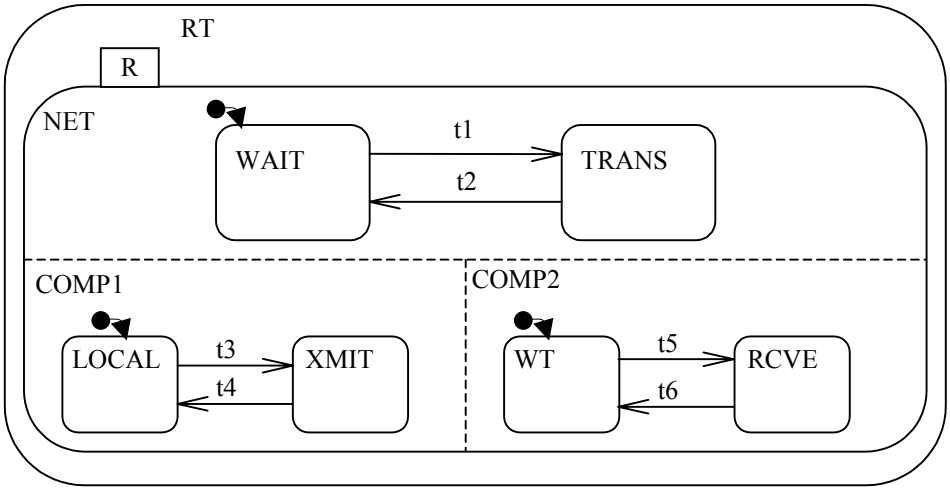
Podejście to wydaje się być niewystarczającym dla oceny wydajności. Dalsza część rozdziału przedstawia kolejne rozszerzenie czasowych Map Stanów opisanych w pracy [50]. Wprowadzona jest tu losowa długość opóźnień związanych z przejściami. Umożliwi to bardziej precyzyjną ocenę zależności czasowych w projektowanych systemach niż zadawanie stałych wielkości opóźnień gdyż te ostatnie z konieczności dotyczą czasów maksymalnych. Wyniki przedstawione w tym rozdziale były też publikowane przez autora w pracy [8].

4.2. Stochastyczne Mapy Stanów

Stochastyczne Mapy Stanów są rozszerzeniem znanego języka graficznej specyfikacji systemów reaktywnych.

W omawianej tu wersji Map Stanów dodano pewne elementy uwzględniające upływ czasu. Przejście wychodzące ze stanu *s* może zajść jeśli system przebywał w tym stanie przynajmniej przez czas t_{del} będący

realizacją zmiennej losowej o dystrybuancie F i nie dłużej niż czas t_{out}. Jeżeli nie podano dystrybuanty to przejście jest natychmiastowe ($t_{del} \equiv 0$). Pominięcie specyfikacji t_{out} oznacza przyjęcie jego wartości za nieskończoną. Podczas realizacji przejścia mogą być wykonywane akcje polegające na generowaniu zdarzeń wewnętrznych. Te zdarzenia mogą mieć zadany czas życia, po upływie którego są usuwane ze zbioru zaistniałych zdarzeń. Brak określenia czasu życia zdarzenia oznacza, że jest ono aktywne do następnego kroku (zdefiniowanego w rozdziale 4.4). Czas rozpatrywany tu, podobnie jak w pracy [50], jest gęsty. Jest on wyrażony nieujemnymi liczbami wymiernymi.



t1:srq t2:,($\mathbf{U}_{1,3}$)/rrq;sack

t3:send/srq t4:sack

t5:rrq t6:,($\mathbf{U}_{0,1}$)

Rys. 4.1 Przykładowa Stochastyczna Mapa Stanów

Na rys. 4.1 jest przedstawiony przykład Stochastycznej Mapy Stanów. Obrazuje on transmisję danych między komputerem COMP1 a komputerem COMP2 poprzez sieć NET. Etykiety przejść zapisane są zgodnie z notacją wprowadzoną w punkcie 4.3.3. W odpowiedzi na zewnętrzne żądanie transmisji (zdarzenie *send*) komputer pierwszy generuje bez opóźnienia zdarzenie *srq* (zlecenie wykonania transmisji przez sieć), które powoduje wykonanie przejścia *t1* w komponencie NET. Wszystko to dzieje się w jednym

kroku, a co za tym idzie, w jednej chwili. Po upływie losowego czasu z przedziału [1,3) sieć wraca do stanu WAIT (przejście *t2*) z wygenerowaniem zlecenia odbioru czyli zdarzenia *rrq*, które powoduje natychmiastowe wykonanie przejścia *t5* i zmianę stanu COMP2 na RCVE. Jednocześnie generowane jest zdarzenie *sack* czyli potwierdzenie wykonania transmisji. Powoduje to powrót urządzenia COMP1 do stanu LOCAL. Jest to drugi krok. Możliwe jest kolejne przejście chociaż komputer COMP2 nie jest jeszcze gotowy do odbioru. Gotowość ta zostanie przywrócona po wykonaniu przejścia *t6*, co nastąpi w kroku trzecim po upływie losowego czasu z przedziału [0,1).

4.3. Składnia

Mapa Stanów, zdefiniowana jak w pracy [54], jest ciągiem $Z = \langle \Pi, S, T \rangle$, gdzie:

— Π jest skończonym zbiorem *zdarzeń elementarnych*,

— S jest skończonym zbiorem *stanów*,

— T jest skończonym zbiorem *przejść*.

Własności przytoczone w tym rozdziale dotyczące podstawowych cech Map Stanów są wzorowane na pracy [54], idea czasowego rozszerzenia zaczerpnięta została z pracy [50], natomiast rozszerzenia stochastyczne są wynikiem własnej pracy autora.

4.3.1. Zdarzenia

Zdarzenia mogą pochodzić od środowiska, w którym system pracuje lub mogą być generowane przez sam system. W tym drugim przypadku mogą mieć określony czas życia. Jeśli zdarzeniu nie przypiszemy czasu życia, to przyjmuje się, że jest ono aktywne do zakończenia najbliższego kroku. Określamy także zbiór $\overline{\Pi}$ zawierający negacje zdarzeń elementarnych: $\overline{\Pi} = \{\overline{e} \mid e \in \Pi\}$, przy czym $\overline{e}$ oznacza niezaistnienie zdarzenia *e*.

4.3.2. Stany i ich właściwości

Stany w Mapach Stanów reprezentowane są jako prostokąty o zaokrąglonych rogach. Hierarchia stanów zaznaczana jest poprzez umieszczanie prostokątów symbolizujących podstany wewnątrz figur oznaczających nadstany. Formalnie hierarchia opisywana jest za pomocą *funkcji hierarchii*: $\rho : S \to 2^S$, która przyporządkowuje każdemu stanowi zbiór jego bezpośrednich podstanów. Na przykład dla mapy z rys. 4.1 mamy ρ(R)={NET,COMP1,COMP2}. Stan *s*, dla którego $\rho(s) = \varnothing$ nazywamy prostym. Takimi stanami są np. WAIT lub LOCAL.
Definiujemy także funkcje pomocnicze:

$$\rho^0(s) = \{s\};\quad \rho^{i+1}(s) = \bigcup_{s' \in \rho(s)} \rho^i(s'), dla\ i \geq 0, \tag{4.1}$$

$$\rho^+(s) = \bigcup_{i \geq 1} \rho^i(s);\quad \rho^*(s) = \bigcup_{i \geq 0} \rho^i(s), \tag{4.2}$$

$$\tilde{\rho}^*(x,y) \Leftrightarrow \left(x \in \rho^*(y) \vee y \in \rho^*(x)\right) \tag{4.3}$$

Funkcje ρ^+ i ρ^* oznaczają odpowiednio niezwrotne i zwrotne domknięcia przechodnie funkcji ρ. Wyznaczają one relacje przodek-potomek. Funkcja $\tilde{\rho}^* : S \times S \to \{T, F\}$ określa, czy jeden ze stanów będących jej parametrami jest potomkiem drugiego. Mapy Stanów muszą spełniać następujące warunki:

$$\underset{root \in S}{\exists!}\ \rho^*(root) = S, \tag{4.4}$$

$$\underset{s \in S}{\forall}\ s \notin \rho^+(s), \tag{4.5}$$

$$\left(s' \notin \rho^+(s) \wedge s \notin \rho^+(s')\right) \Rightarrow \left(\left(\rho^*(s) \cap \rho^*(s') \neq \varnothing\right) \Rightarrow s = s'\right). \tag{4.6}$$

Oznaczają one, że:

— istnieje dokładnie jeden taki stan nazywany *root*, który jest przodkiem wszystkich innych stanów,

— żaden stan nie jest własnym potomkiem,

— każdy stan (oprócz root) ma dokładnie jednego przodka na każdym poziomie hierarchii.

Każdy stan jest typu *XOR*, *AND* lub *PRIM*, co jest określone za pomocą *funkcji typu* $\phi : S \to \{AND, XOR, PRIM\}$. Na funkcję tę nałożone są następujące ograniczenia:

$$\phi(root) = XOR\,, \tag{4.7}$$

$$\underset{s \in S}{\forall} \left(\phi(s) = AND \Rightarrow \underset{s' \in \rho(s)}{\forall} \phi(s') = XOR \right). \tag{4.8}$$

Wzór (4.8) oznacza, że bezpośrednie podstany stanu typu *AND* muszą być typu *XOR*.

W każdym stanie typu *XOR* wyróżniamy jeden z jego podstanów (podstan domyślny) oznaczając go na diagramie strzałką. Definiujemy częściową funkcję $\delta : S \to S$ wyznaczającą domyślny podstan każdego stanu typu *XOR*.

Najbliższym wspólnym przodkiem (ang. Least (or Lowest) Common Ancestor) dwóch stanów nazywamy najmniejszy stan zawierający je. Formalnie zapisujemy to za pomocą funkcji $lca : S \times S \to S$.

$$\underset{x,y \in S}{\forall} lca(x,y) = s \Leftrightarrow \left(\{x,y\} \subseteq \rho^*(s) \wedge \underset{s' \in S}{\forall} \left(\{x,y\} \subseteq \rho^*(s') \Rightarrow s \in \rho^*(s') \right) \right) \tag{4.9}$$

Analogicznie definiujemy *lca*(X), czyli najbliższego wspólnego przodka zbioru stanów.

$$\underset{X \subseteq S}{\forall} lca(X) = s \Leftrightarrow \left(X \subseteq \rho^*(s) \wedge \underset{s' \in S}{\forall} \left(X \subseteq \rho^*(s') \Rightarrow s \in \rho^*(s') \right) \right) \tag{4.10}$$

Dwa stany nazywamy *ortogonalnymi* jeśli żaden z nich nie jest potomkiem drugiego oraz najbliższy wspólny przodek tych stanów jest typu *AND*. Relacja ta opisuje sytuację, gdy dwie aktywności mogą zachodzić jednocześnie nie

wykluczając się wzajemnie. Formalnie relację ortogonalności zapisujemy następująco:

$$\underset{x,y\in S}{\forall} x\perp y \Leftrightarrow \left(\neg\hat{\rho}^{*}(x,y)\wedge\phi(lca(x,y))=AND\right) \tag{4.11}$$

W naturalny sposób możemy rozszerzyć relację ortogonalności na zbiór stanów. Zbiór X jest ortogonalny, jeśli jego elementy są parami ortogonalne:

$$X\perp X \Leftrightarrow \left(\underset{x,y\in X}{\forall} x=y\vee x\perp y\right) \tag{4.12}$$

Zbiór stanów nazywamy *spójnym*, gdy jego elementy są parami powiązane relacją przodek-potomek lub są ortogonalne. Zbiór spójny zawiera stany, w których system może przebywać jednocześnie. Formalnie logiczną funkcję $consistent{:}2^{S}\rightarrow\{T,F\}$ zapisujemy:

$$\underset{X\subseteq S}{\forall} consistent(X) \Leftrightarrow \left(\underset{x,y\in X}{\forall}\hat{\rho}^{*}(x,y)\vee x\perp y\right) \tag{4.13}$$

Spójny zbiór stanów, do którego nie można dodać żadnego stanu nie niszcząc spójności nazywamy zbiorem maksymalnie spójnym lub *konfiguracją*:

$$\underset{\substack{X\subseteq S\\ consistent(X)}}{\forall}\Xi(X) \Leftrightarrow \underset{s\in S-X}{\forall}\neg consistent(X\cup\{s\}) \tag{4.14}$$

W celu rozszerzenia danego spójnego zbioru stanów X do konfiguracji definiujemy funkcje określające reguły dołączania stanów. Domyślne dopełnienie w dół (ozn. $\Downarrow$) tworzy zbiór poprzez dodawanie dla każdego stanu typu *AND* wszystkich jego podstanów, a dla stanu typu *XOR* - podstanu domyślnego, o ile w zbiorze wyjściowym nie było innego podstanu:

$$\underset{X\subset S}{\forall}\Downarrow X=Y \Leftrightarrow \underset{s\in Y}{\forall}\begin{pmatrix}(\phi(s)=XOR)\Rightarrow(X\cap\rho(s)=\varnothing\Rightarrow\delta(s)\in Y),\\ (\phi(s)=AND)\Rightarrow\rho(s)\in Y\end{pmatrix} \tag{4.15}$$

Twierdzenie 4.1

Domyślne dopełnienie w dół spójnego zbioru stanów jest spójne.

Dowód

Aby zaburzyć spójność zbioru stanów należałoby dodać do niego drugi podstan jakiegoś stanu typu *XOR*. Możliwość taką wyklucza wzór (4.15) definiujący domyślne dopełnienie w dół. Wynika stąd, że operator ten nie zaburza spójności. *QED*

Aby zdefiniować pełne dopełnienie domyślne zdefiniujemy najpierw ścieżkę między dwoma stanami, czyli zbiór stanów będących jednocześnie podstanami jednego i nadstanami drugiego z nich. Ścieżkę między stanami s i s' zapisujemy jako $Z^{s}_{s'}$. Na przykład $Z^{R}_{XMIT} = \{R,COMP1,XMIT\}$. Formalnie ścieżkę opisuje wzór (4.16).

$$\underset{\substack{s,s' \in S \\ s' \in \rho(s)}}{\forall} Z^{s}_{s'} = \left\{ s'' : s'' \in \rho^{*}(s) \wedge s' \in \rho^{*}(s'') \right\} \tag{4.16}$$

Domyślne dopełnienie zbioru X (ozn. $\Updownarrow X$) definiuje poniższy wzór:

$$\Updownarrow X = \Downarrow \left(\bigcup_{s \in X} Z^{root}_{s} \right) \tag{4.17}$$

Twierdzenie 4.2

Domyślne dopełnienie spójnego zbioru X tworzy konfigurację, do której należą wszystkie stany ze zbioru X.

Dowód

Aby udowodnić pierwszą część tezy powyższego twierdzenia, należy wykazać, że uzyskany zbiór stanów jest spójny, oraz że dodanie jakiegokolwiek stanu spójność tę zniszczy.

Jeśli stan **s1** jest spójny ze stanem **s2**, to jest spójny także z każdym stanem należącym do ścieżki $Z_{s_2}^{root}$, co wynika wprost z definicji funkcji *consistent* oraz faktu, że stan ortogonalny do innego jest też ortogonalny do jego rodzica. Z powyższych rozważań wynika także, iż wszystkie zbiory stanów sumowane we wzorze definiującym dopełnienie są ze sobą spójne. Z powyższego oraz twierdzenia 4.1 wynika, że domyślne dopełnienie spójnego zbioru stanów jest spójne.

Załóżmy teraz, że istnieje stan **s3**, który dodany do uzyskanego zbioru stanów X' nie zaburza jego spójności. Oznacza to, że stan ten jest spójny z każdym stanem zbioru X' i jednocześnie nie należy do niego. Wszystkie stany potomne stanów typu *AND* należących do X' należą też do X', a więc stanu **s3** należy poszukiwać wśród podstanów stanu typu *XOR*. Jednak dla każdego stanu typu *XOR* należącego do X', jeden z jego podstanów też należy do X'. Pozostałe podstany oraz ich potomne są niespójne z nim. Jak więc widać stan **s3** o zadanych cechach nie istnieje. Kończy to dowód pierwszej części twierdzenia.

Dowód drugiej części jest oczywisty. Żaden z operatorów wzoru definicyjnego nie usuwa stanów, a więc każdy stan ze zbioru X należy do jego domyślnego dopełnienia. *QED*

Stan systemu w danej chwili czasu opisany jest poprzez *czasową konfigurację*. Jest ona czwórką: $C^t = \langle C, ent, test, t_{del} \rangle$, gdzie

$$\begin{aligned} &C \subseteq S, \Xi(C), \\ &ent : C \to Q, \\ &test : Q \to 2^{\Pi}, \\ &t_{del} : T \to Q. \end{aligned} \tag{4.18}$$

W powyższym wzorze C jest spójnym podzbiorem zbioru stanów spełniającym warunek Ξ(C) (4.14) czyli konfiguracją, *ent* jest funkcją przypisującą każdemu stanowi z konfiguracji C czas wejścia do niego, *Q* oznacza zbiór

nieujemnych liczb wymiernych, *test* jest funkcją tworzącą zbiór zdarzeń widocznych w danym momencie czasu, natomiast t_{del} oznacza częściową funkcję przypisującą przejściom wychodzącym ze stanów należących do konfiguracji C wylosowane wartości czasu opóźnienia.

4.3.3. Przejścia

Przejścia na Mapie Stanów zaznaczamy za pomocą strzałek pomiędzy stanami. Stany, w których zaczynają się przejścia nazywamy źródłowymi, a te, w których się kończą — docelowymi. Zbiory stanów źródłowych oraz docelowych każdego przejścia muszą być ortogonalne (warunek (4.19)). Zbiory te są określone poprzez funkcje *src,targ*:$T \rightarrow 2^S$. Wymagamy, aby funkcje te spełniały następujące warunki:

$$\forall_{t\in T}\ \forall_{x,y\in src(t)}\ x \perp y \text{ oraz } \forall_{t\in T}\ \forall_{x,y\in targ(t)}\ x \perp y \tag{4.19}$$

$$\forall_{t\in T}\left(lca(t) \not\subset src(t) \wedge lca(t) \not\subset targ(t)\right) \tag{4.20}$$

$$\forall_{t\in T}\ \phi(lca(t)) = OR \tag{4.21}$$

Poprzez *lca(t)* oznaczamy *lca(src(t)∪ targ(t))*.

Podobnie jak dla stanów, określamy spójność przejść. Przejścia są spójne, gdy zawarte są w stanach ortogonalnych. O przejściach niespójnych mówimy, że są w konflikcie.

$$consistent(t_1, t_2) \Leftrightarrow \left(t_1 = t_2 \vee lca(t_1) \perp lca(t_2)\right) \tag{4.22}$$

Dla dowolnego zbioru przejść określmy funkcję $consistentset : 2^T \rightarrow 2^T$ generującą zbiór przejść spójnych z elementami argumentu funkcji.

$$consistentset(T') = \{t \in T \mid \forall_{t'\in T'} consistent(t, t')\} \tag{4.23}$$

Przejścia są opatrzone etykietami postaci:

trig,(F,t_{out})/<act_1,t_1>;<act_2,t_2>;… gdzie:

trig jest kombinacją zdarzeń elementarnych i ich negacji,

F — dystrybuanta zmiennej losowej opisującej opóźnienie przejścia,

t_{out} — czas przeterminowania,

act_i — akcja polegająca na generacji zdarzenia wewnętrznego o czasie życia t_i dla i=1,2,….

Każdy z elementów etykiety może być pominięty.

Dla przejścia *t* oznaczmy:

rand(t) — realizacja zm. losowej opóźnienia przejścia,

act(t) — zbiór par <act_i,t_i> dla i=1,2,…,

jeśli *a*=<act_i,t_i>, wtedy *a.act*=act_i, *a.t*=t_i.

4.4. Semantyka

Zachowanie systemu opisanego Mapą Stanów określone jest poprzez serię *kroków* począwszy od konfiguracji startowej. Każdy krok składa się z serii mikrokroków zapoczątkowanej zdarzeniami zewnętrznymi bądź upływem czasu. Zanim zdefiniujemy mikrokrok zajmiemy się funkcją En(T,τ). Jest to funkcja określająca (niejednoznacznie) zbiór przejść, które mogą być wykonane jednocześnie w czasie τ, konfiguracji C^t i przy zaistnieniu danego zbioru zdarzeń. Konfigurację oraz zbiór zdarzeń traktujemy jako stałe w danym mikrokroku i dlatego nie są one parametrami funkcji En.

$$En(T,\tau) = relevant(C^t,\tau) \cap consistentset(T) \cap triggered(test(\tau)) \qquad (4.24)$$

We wzorze tym *T* oznacza zbiór wybranych (nie określamy jak) przejść. Funkcja *relevant* wyznacza zbiór przejść wychodzących ze stanów należących do bieżącej konfiguracji czasowej i mogących być wykonanymi w momencie τ.

$$relevant(C^t,\tau)=\left\{\begin{array}{l} t\in T \mid src(t)\subset C \wedge \\ \tau\in\left[ent(src(t))+t_{del}(t),ent(src(t))+t_{out}(t)\right] \end{array}\right\} \qquad (4.25)$$

Funkcja *consistentset* określa zbiór przejść spójnych ze zbiorem wybranym. Funkcja *triggered* wyznacza zbiór pzejść przygotowanych do wykonania przez zbiór zdarzeń *test*(τ) zawierający zdarzenia zewnętrzne i te spośród wewnętrznych, których czas życia nie upłynął do chwili τ.

Ψ jest dopuszczalnym mikrokrokiem, gdy (dyskusja p. praca (54)):

$$\begin{array}{l} \underset{T'\subset\Psi}{\forall} En(T',\tau)\cap(\Psi-T')\neq\varnothing, \\ \Psi=En(\Psi,\tau). \end{array} \qquad (4.26)$$

Zbiór Ψ można też wyznaczyć w sposób równoważny za pomocą poniższej procedury:

$\Psi=\varnothing$;
loop : *if* $(\Psi=En(\Psi,\tau))$ *then* SUCCESS;
elseif $(\Psi\subset En(\Psi,\tau))$ *then* $\Psi=\Psi\cap(t\in(En(\Psi,\tau)-\Psi))$;
else FAIL;
fi
GOTO *loop*; (4.27)

W wyniku wykonania mikrokroku system przechodzi do czasowej konfiguracji $C^{t'}$. Uzyskujemy ją z konfiguracji C^t poprzez usunięcie stanów, które system opuścił, dodanie stanów, które osiągnął, modyfikację funkcji *ent* i *test* oraz wylosowanie opóźnień dla przejść wychodzących z nowych stanów. Formalnie zapiszemy to poprzez zdefiniowanie funkcji *NextTConf*:

$$
\begin{aligned}
&NextTConf(C^t, \Psi) = \langle C', ent', test', t'_{del} \rangle \text{ gdzie:} \\
&C' = \Updownarrow\left(\left(C - \bigcup_{t\in\Psi} \rho^+(lca(t))\right) \cup \bigcup_{t\in\Psi} targ(t)\right), \\
&\forall_{s\in C\cap C'} ent'(s) = ent(s)\,;\ \forall_{s\in C'-C} ent'(s) = \tau, \\
&\forall_{\tau'\geq\tau} test'(\tau') = test(\tau') \cup \{e = a.act \mid a \in act(t), t\in\Psi, \tau'\leq\tau+a.t \vee a.t=0\}, \\
&\forall_{s\in C'\cap C} \forall_{t:src(t)=s} t'_{del}(t) = t_{del}(t); \\
&\forall_{s\in C'-C} \forall_{t:src(t)=s} t'_{del}(t) = rand(t).
\end{aligned}
\tag{4.28}
$$

Krok definiujemy jako najdłuższą sekwencję mikrokroków, w których przejścia nie powtarzają się. Cały krok odbywa się w chwili τ. Krok następny może być wykonany dopiero w chwili τ'>τ. Może być on wyzwolony zewnętrznie lub wewnętrznie. Pierwszy przypadek zachodzi, gdy w czasie τ' pojawiają się zdarzenia zewnętrzne, które wyzwalają przejścia. Przypadek drugi związany jest z upływem czasu i zachodzi, kiedy upływa czas t_{del} dla niepustego zbioru przejść przygotowanych (ze względu na konfigurację i zdarzenia) do wykonania. Dokładniejsze omówienie zagadnienia czasu oraz problemów z nim związanych znajduje się w pracy [50].

4.5. Podsumowanie

W rozdziale 4 przedstawiona została składnia i semantyka stochastycznego rozszerzenia Map Stanów. Zastosowano wariant semantyki wprowadzony w pracy [54] bazujący na pojęciach kroku i mikrokroku. Zdefiniowano czasową konfigurację systemu, etykiety z dodanymi elementami czasu, nową postać funkcji zezwalającej *En*. Podano definicję mikrokroku uwzględniającą elementy stochastyczne modelu.

Formalny zapis semantyki powinien ułatwić tworzenie narzędzi programistycznych wspomagających proces analizy wydajności systemów opisanych za pomocą Map Stanów.

Możliwe jest także dalsze ustochastycznienie modelu poprzez wprowadzenie losowego czasu życia zdarzeń oraz time-outów. Interesujące byłoby też dodanie prawdopodobieństw wyboru przejść na wypadek konfliktu.

Rozdział kolejny przedstawia nieco zmodyfikowaną wersję Map Stanów pozbawioną niedeterminizmu wyboru przejść.

5. Analityczna metoda oceny wydajności systemów

5.1. Wstęp

Poprzedni rozdział przedstawia propozycję stochastycznego rozszerzenia Map Stanów na potrzeby badań nad wydajnością projektowanych systemów. Zdefiniowano w nim składnię i semantykę operacyjną tego rozszerzenia. Są one przedstawione w formie algebraicznej jako zestaw funkcji i relacji. Podejście tam zaprezentowane ma pewne wady. Niektóre funkcje użyte do zdefiniowania semantyki, aczkolwiek poprawne matematycznie, są trudne do zaimplementowania. Algorytmizację utrudnia też fakt istnienia niejednoznaczności wyboru przejść w definicji kroku. Ponadto propozycja ta, z racji nieokreślenia rozkładów prawdopodobieństwa stosowanych w modelu zmiennych losowych, nadaje się raczej do badań symulacyjnych. Przykład takich badań przedstawiony będzie w rozdziale 6.6. Teraz zaś zostanie zaprezentowane analityczne podejście do oceny wydajności.

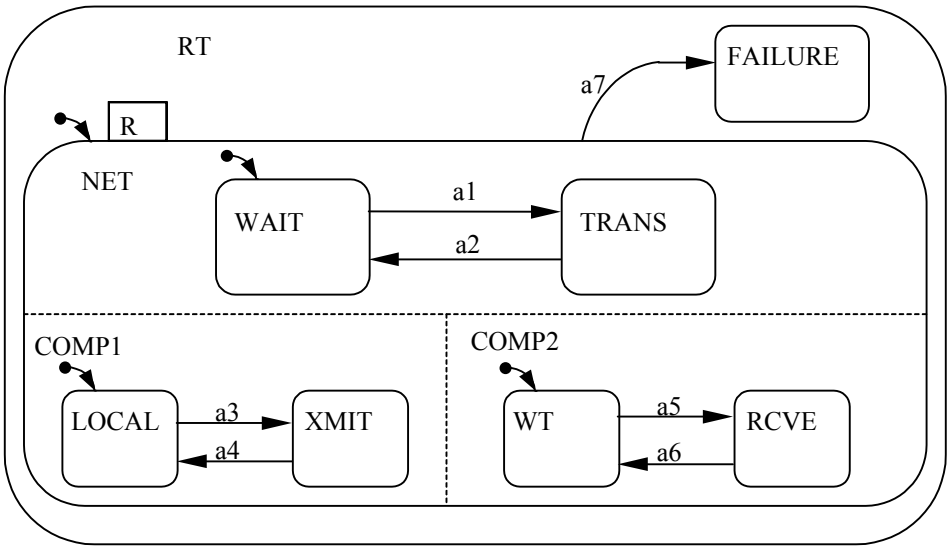
5.2. Markowowskie Mapy Stanów

5.2.1. Wprowadzenie

W bieżącym rozdziale przedstawione zostaną Mapy Stanów wzbogacone o elementy czasu w postaci opóźnień opisanych zmiennymi losowymi o rozkładzie wykładniczym. Rozkład taki charakteryzuje się właściwością braku pamięci co pozwala na zastosowanie teorii procesów Markowa do opisu zachowania się projektowanego lub badanego systemu w czasie. Zamieszczona zostanie także propozycja algebry procesowej dla Markowowskich Map Stanów. Propozycja ta wzorowana jest na algebrze procesowej dla Map Stanów wprowadzonej przez Useltona i Smolkę w pracy [67]. Główne tezy tego podrozdziału publikowane były też w pracach [5] i [6], których jestem współautorem.

5.2.2. Przykład wprowadzający

Na rys. 5.1 pokazany jest przykład Markowowskiej Mapy Stanów. Podobnie jak w przykładzie z poprzedniego rozdziału, mapa ta pokazuje w uproszczony sposób transmisję pomiędzy dwoma komputerami COMP1 i COMP2 poprzez sieć NET.



a1:<∞,1,{srq},∅,<>>

a2:<1,1, ∅,{rrq},<sack,9.3>>

a3:<∞,1,{send},{srq},<>>

a4:<∞,1,{sack},∅,<>>

a5:<∞,1,{rrq},∅,<>>

a6:<5,1,∅,∅,<>>

a7:<∞,1,{fail},∅,<>>

Rys. 5.1 Przykład Markowowskiej Mapy Stanów

Zawarta jest w niej komórka R typu *AND*, która z kolei zawiera trzy ortogonalne podkomórki. Podkomórka COMP1 reprezentuje pierwszy komputer (nadający), podkomórka COMP2 — komputer drugi (odbierający), a podkomórka NET — sieć transmitującą dane pomiędzy komputerami. Pozostałe podkomórki zawarte w podkomórkach COMP1, COMP2, NET oraz RT, są typu *PRIM*. Każda podkomórka tego typu oznacza tu stan odpowiadającego jej urządzenia. Na przykład podkomórka WAIT reprezentuje stan sieci, w którym oczekuje ona na dane. Łuki zaczynające się w zaczernionych kółkach wskazują na komórki domyślne. Łuki są wzajemnie

jednoznacznie identyfikowane przez ich nazwy. Z nazwami łuków lub z samymi łukami kojarzymy *etykiety łuków*. Etykiety złożone są z pięciu elementów. Pierwszy z nich, λ, jest parametrem *opóźnienia otwarcia* łuku. Opóźnienie to jest realizacją zmiennej losowej o rozkładzie wykładniczym z parametrem λ. Gdy λ=∞, wtedy opóźnienie jest równe zero. Dany łuk może być użyty do przejścia między sąsiadującymi komórkami dopiero po upływie czasu opóźnienia otwarcia. Drugi element, *p*, opisuje *priorytet* łuku. Jest on istotny tylko wtedy, gdy więcej niż jeden łuk wychodzi z jednej komórki. W przypadku, gdy więcej niż jeden taki łuk może zostać wykonany, brany jest ten o najwyższym priorytecie. Trzecim parametrem jest zbiór *E*, zawierający *zdarzenia wyzwalające* dany łuk. Następny, czwarty element opisuje zbiór *zdarzeń natychmiastowych* generowanych podczas wykonywania danego łuku. Ostatni wreszcie, piąty element etykiety jest skończoną listą *zdarzeń odroczonych* czyli zdarzeń generowanych po upływie losowego czasu po wykonaniu danego łuku. Jako przykład zostanie opisana etykieta *a2*. Powinna być ona odczytywana następująco: „Łuk o nazwie *a2* jest łukiem opóźnionym (λ=1) o priorytecie równym 1 (ten parametr jest nieistotny z powodu braku kolidujących łuków). Łuk jest wyzwalany pustym zbiorem zdarzeń czyli zostanie wyzwolony natychmiast po otwarciu. W czasie wykonywania tego łuku zostaje wygenerowane zdarzenie "rrq" oraz zaplanowane odroczone zdarzenie "sack". Samo zdarzenie wystąpi po upłynięciu czasu będącego realizacją zmiennej losowej o rozkładzie wykładniczym z parametrem λ = 9.3.”

Specyfikowany system pracuje w następujący sposób. Urządzenie COMP1 odpowiadając na zewnętrzne zdarzenie "send", bez opóźnienia przechodzi do stanu XMIT generując przy tym zdarzenie "srq". Następnie sieć NET zmienia swój stan z WAIT na TRANS. Oczekuje ona w tym stanie na upłynięcie czasu opóźnienia opisanego rozkładem wykładniczym z parametrem λ=1. Po tym czasie sieć wraca do stanu WAIT poprzez łuk *a2* planując zdarzenie "sack", które wystąpi po upływie czasu opisanego zmienną losową o rozkładzie

wykładniczym z parametrem λ=9.3. Niezwłocznie po pojawieniu się zdarzenia "sack", COMP1 powraca do stanu LOCAL. W czasie przejścia wzdłuż łuku *a2* jest też generowane zdarzenie "rrq", które z kolei jest rozpoznawane przez urządzenie COMP2 powodując natychmiastowe przejście ze stanu WT do RCVE. Po upływie losowego czasu o rozkładzie wykładniczym z parametrem λ=5, wykonywane jest przejście wzdłuż łuku *a6* i urządzenie COMP2 powraca do wyjściowego stanu WT.

Gdy wystąpi jakiś błąd podczas pracy systemu i pojawi się zdarzenie "fail", system natychmiast przejdzie do stanu FAILURE. Oznacza to, że pojawienie się tego zdarzenia przerwie pracę systemu w stanie R ponieważ łuki wychodzące ze stanu R mają wyższy priorytet strukturalny niż łuki wychodzące ze stanów potomnych.

5.2.3. Mapy Stanów

Definicja 5.1 [5]

Mapę Stanów *S* definiujemy jako ciąg 6-elementowy:

$S = <BoxN, \rho, \Psi, \delta, ArcN, Arc>$

gdzie:

- *BoxN* jest skończonym zbiorem *nazw* komórek.
- $\rho \subseteq BoxN \times BoxN$ jest relacją *hierarchii*: $<b_1, b_2> \in \rho$ oznacza, że b_2 jest komórką potomną — "dzieckiem" "rodzica" b_1. Zbiór *BoxN* i relacja ρ definiują drzewo komórek. Korzeń drzewa, *r*, nie ma rodziców, a liście drzewa nie mają dzieci. Definiujemy też ρ^* jako zwrotne przechodnie domknięcie .
- $\Psi : BoxN \rightarrow \{PRIM, XOR, AND\}$ jest funkcją, która każdej komórce nadaje *typ*. Korzeń musi być typu *XOR*, liście są typu *PRIM*, a pozostałe komórki mogą być typu *XOR* lub *AND*.
- $\delta : BoxN \rightarrow 2^{BoxN}$ jest funkcją określającą dla każdej komórki jej podkomórkę *domyślną*. Wartością funkcji dla komórki typu *XOR* jest

zbiór zawierający dokładnie jeden element spośród jej dzieci, podczas gdy dla komórki typu *AND* jest to zbiór wszystkich jej dzieci. Wartością funkcji dla komórek typu *PRIM* jest zbiór pusty. Rozszerzeniem omawianej funkcji jest $\Delta : BoxN \rightarrow 2^{BoxN}$ zdefiniowana następująco: $b \in \Delta(b)$, a dla komórek $b' \in BoxN$ takich, że $<b, b'> \in \rho^*$ mamy $b' \in \Delta(b)$ wtedy i tylko wtedy gdy $\delta(b') \subseteq \Delta(b)$.

— *ArcN* jest skończonym zbiorem *nazw* łuków. $BoxN \cap ArcN = \emptyset$.

— $Arc \subseteq BoxN \times ArcN \times BoxN$ jest zbiorem *łuków*. Łuk $\alpha \in Arc$ jest trójką $<b_1,a,b_2>$, gdzie: $source(\alpha) = b_1$, $target(\alpha) = b_2$ oraz $name(\alpha) = a$. Wprowadzono tu funkcje *source,target*: $Arc \rightarrow BoxN$ oznaczające, odpowiednio, komórkę początkową i końcową danego łuku. Funkcja *name*: $Arc \rightarrow ArcN$ daje nazwę danego łuku. Przyjmuje się, że łuki są jednoznacznie identyfikowane przez ich nazwy i dlatego można wprowadzić odwrotną funkcję $name^{-1}(a) = \alpha$.

Założenie 5.1

Łuki nie mogą przecinać granic komórek, czyli $<b,a,b'> \in Arc$ implikuje, że b i b' mają wspólnego rodzica typu *XOR*.

Powyższe założenie jest często przyjmowane przy próbach formalnego zdefiniowania Map Stanów. Ułatwia ono formalny zapis modelu i eliminuje niejednoznaczności definicji priorytetów strukturalnych ([24]). Założenie to nie ogranicza ekspresyjności modelu, jak udowodniono w pracy [50].

Zostało wprowadzone wyrażenie $lca(b_1,b_2)$ dla oznaczenia *najbliższego wspólnego przodka* (ang. *least common ancestor*) komórek b_1 i b_2 w drzewie $<BoxN,\rho>$, tzn., $<lca(b_1,b_2),b_i> \in \rho^*$ (i=1,2), i nie istnieje inna komórka b, taka że $<b,b_i> \in \rho^*$, oraz $<lca(b_1,b_2),b> \in \rho^*$.

Komórki $b_1,b_2 \in BoxN$ są *ortogonalne*, co zapisujemy jako $b_1 \perp b_2$, jeśli żadna nie jest przodkiem drugiej oraz $\Psi(lca(b_1,b_2)) = AND$.

Podobnie, dwa łuki $<b_1,a_1,b'_1>$ i $<b_2,a_2,b'_2>$ są *ortogonalne*, co zapisujemy

$<b_1,a_1,b'_1> \perp <b_2,a_2,b'_2>$,

gdy komórki $lca(b_1,b'_1)$ i $lca(b_2,b'_2)$ są ortogonalne.

Przy założeniu (5.1), że łuki nie mogą przecinać granic komórek, dla łuków ortogonalnych, $b_1 \perp b_2$ implikuje, że $b'_1 \perp b'_2$.

Niech $B \subseteq BoxN$ będzie podzbiorem komórek. Wtedy zbiór

$relevant(B) = \{a \in ArcN \mid a = name(\alpha), source(\alpha) \in B\}$

jest zbiorem nazw łuków wychodzących z komórki B, nazywanych relewantnymi do niej.

Definicja 5.2 [5]

a) *Strukturalną konfiguracją* Mapy Stanów S jest podzbiór komórek $B \subseteq BoxN$, taki że $r \in B$ oraz dla każdej komórki $b \in B$ gdy $\Psi(b) = AND$ wtedy wszystkie dzieci należą do B, a gdy $\Psi(b) = XOR$ wtedy dokładnie jedno z dzieci znajduje się w B. Zbiór B wraz z relacją hierarchii ρ ograniczoną do tego zbioru tworzą poddrzewo $<B, \rho \cap B \times B>$, drzewa komórek. *Startowa konfiguracja strukturalna* Mapy Stanów S jest zdefiniowana jako $B_{init} = \Delta(r)$.

b) *Strukturalnym przejściem* T z konfiguracji strukturalnej B do konfiguracji strukturalnej B' jest maksymalny zbiór nazw parami ortogonalnych łuków, dla których komórkami startowymi są elementy zbioru B.

Ponieważ zbiór $BoxN$ jest skończony, oraz konfiguracja strukturalna jest podzbiorem tego zbioru, można sformułować następujący wniosek:

Wniosek 5.1

Zbiór konfiguracji strukturalnych Markowowskiej Mapy Stanów jest skończony.

Przejścia strukturalne zależą od konfiguracji strukturalnych Map Stanów i od środowiska zewnętrznego, które generuje zdarzenia powodujące możliwe przejścia wzdłuż łuków.

Definicja 5.3 [5]

Markowowska Mapa Stanów **MMS** jest parą:

MMS = <*S, L*>

gdzie:

— *S* jest Mapą Stanów,

— *L* jest *funkcją etykietującą*, która daje interpretację łuków. Z każdym łukiem $\alpha \in Arc$, lub z odpowiadającą mu nazwą łuku *a*, skojarzona jest etykieta *l*. Będziemy zapisywać $L(\alpha)$ lub $L(a)$, przy czym $L(\alpha) = L(a)$ gdy $a = name(\alpha)$.

— Każda etykieta jest ciągiem:

$l = <\lambda, p, E, \{e'_1, \ldots, e'_m\}, <<e_1, \lambda_1>, \ldots, <e_n, \lambda_n>>>$

gdzie:

— $\lambda \in R_+ \cup \{\infty\}$ jest parametrem *opóźnienia otwarcia* łuku. Gdy $\lambda=\infty$ wtedy nie ma żadnego opóźnienia, w przeciwnym razie, opóźnienie $\tau>0$ jest realizacją zmiennej losowej o rozkładzie wykładniczym z parametrem $\lambda \in R_+$, co zapisujemy *rand*(λ). $p \in$ Nat jest *priorytetem* łuku.

— $E \subseteq EVENTS$ jest zbiorem *zdarzeń wyzwalających* dany łuk, gdzie *EVENTS* jest skończonym zbiorem nazw zdarzeń.

— $\{e'_1, \ldots, e'_m\}$, gdzie $e'_i \in EVENTS$ ($m \geq 0$), jest skończonym zbiorem *natychmiastowych zdarzeń wewnętrznych*, które są generowane podczas wykonywania danego łuku.

— $<<e_1,\lambda_1>, \ldots, <e_n,\lambda_n>>$ gdzie

$e_i \in EVENTS$ i $\lambda_i \in R_+$ dla i = 1, ..., n ($n \geq 0$), jest skończoną listą *odroczonych zdarzeń wewnętrznych* skojarzonego z etykietą łuku, które są inicjowane podczas wykonywania danego łuku. λ_i jest parametrem zmiennej losowej o rozkładzie wykładniczym. Zmienna ta wyraża opóźnienie generacji zdarzenia e_i po wykonaniu łuku.

Opóźnienia otwarcia zostały wprowadzone do Map Stanów w celu umożliwienia modelowania pewnych aktywności zachodzących w komórce źródłowej danego łuku zanim będzie mogła być opuszczona wzdłuż tego łuku. Zdarzenia natychmiastowe i odroczone zostały wprowadzone dla zamodelowania rezultatów akcji podejmowanych podczas przechodzenia pomiędzy komórkami przez dany łuk.

Wprowadźmy selektory dla elementów składowych etykiety. Jeżeli

$l = <\lambda, p, E, \{e'_1, \ldots, e'_m\}, <<e_1, \lambda_1>, \ldots, <e_n, \lambda_n>>>$

wtedy oznaczamy:

$l.delay = \lambda$, $l.priority = p$, $l.trigger = E$, $l.immediate = \{e'_1, \ldots, e'_m\}$,

$l.deferred = <<e_1, \lambda_1>, \ldots, <e_n, \lambda_n>>$

Założenie 5.2

Aby zmniejszyć niedeterminizm przyjęto, że łuki opuszczające daną komórkę mają różny priorytet tzn. jeśli

$source(\alpha_1) = source(\alpha_2)$ to $L(\alpha_1).priority \neq L(\alpha_2).priority$.

Pozwala to na dokonanie jednoznacznego wyboru łuku wtedy, gdy więcej niż jeden łuk wychodzący z danej komórki jest wyzwolony.

Pojęcie konfiguracji strukturalnej Markowowskich Map Stanów jest identyczne z wyżej definiowanym dla Map Stanów (def. 5.2).

5.2.4. Algebra Procesowa Map Stanów

W tym rozdziale przedstawiona zostanie konfiguracja strukturalna Map Stanów w formie algebraicznej. Zasadniczo algebra ta jest wzorowana na podejściu zaprezentowanym przez Useltona i Smolkę w pracy [67]. Wprowadzono jednak pewne modyfikacje, takie jak w [5]. Dla *Algebry Procesowej Map Stanów* (StPA), wprowadzona zostaje następująca definicja składni wyrażeń procesowych *PROC* w notacji podobnej do BNF:

$P ::= \mathbf{0} \mid a; P \mid P_1 + P_2 \mid P_1 \| P_2 \mid P_1[P_2] \mid p \mid \langle p :: \Xi \rangle$

gdzie

— P, P_1, P_2 są meta zmiennymi nad zbiorem *PROC*,

— $a \in ArcN$,

— $p \in ProcN$ gdzie *ProcN* jest zbiorem nazw procesów,

— Ξ jest *specyfikacją rekursywną*,

— **0** jest *procesem pustym*,

— *a;P* jest *prefiksowaniem akcją*,

— $P_1 + P_2$ jest zapisem *wyboru niedeterministycznego*,

— $P_1 \| P_2$ jest *złożeniem równoległym*, które skleja akcje P_1 i P_2 lub je synchronizuje,

— $P_1[P_2]$ jest *uszczegółowieniem komórki*, w którym komórka reprezentowana przez P_1 jest uszczegółowiona przez P_2 tak, że akcje należące do P_2 mogą zachodzić dopóki nie może być wzięta żadna z akcji należących do P_1.

Zachodzą następujące oczywiste równania składniowe:

$P + \mathbf{0} = P$, $P \| \mathbf{0} = P$, $P[\mathbf{0}] = P$.

$p \in ProcN$ jest zmienną rekursywną, a $\langle p :: \Xi \rangle$ jest definicją procesu ze zmienną p związaną ze specyfikacją rekursywną Ξ. Zmienna p jest *wolna* jeśli nie jest związana. Term P nazywamy *zamkniętym* jeśli nie zawiera wolnych zmiennych, w przeciwnym przypadku jest on *otwarty*.

Specyfikacja rekursywna Ξ jest zbiorem równań rekurencyjnych

$\{p_i = P_i \mid i \in I\}$,

w którym tylko zmienne w $\{p_i \mid i \in I\}$ są wolne w każdym P_i. Zmienna p jest *chroniona* w termie P gdy występuje wyłącznie w podwyrażeniu typu *a;P'*. Gdy wszystkie zmienne są chronione w termach t_i należących do specyfikacji rekursywnej Ξ to cała specyfikacja Ξ jest chroniona. Wymaga się aby poprawne specyfikacje rekursywne były chronione. $\langle p :: \Xi \rangle$ jest wyrażeniem

procesowym uzyskanym przez zamianę w otwartym termie P każdego p_i na definicję procesu $<p_i :: \Xi>$.

W dalszej części używana będzie następująca notacja:

$$\sum_{1 \le i \le n} P_i$$

dla wyrażenia postaci $P_1 + \ldots + P_n$, i podobnie

$$\prod_{1 \le i \le n} P_i$$

dla wyrażeń typu $P_1 \parallel \ldots \parallel P_n$.

Założenie 5.3

Składniki P_1 i P_2 wyrażenia wyboru $P_1 + P_2$ nie mogą być wyrażeniami równoległymi, tzn. P_i nie może mieć postaci $P'_i \parallel P''_i$, gdzie P'_i, P''_i (i = 1, 2) są wyrażeniami procesowymi.

Poniższa funkcja rekurencyjna definiuje transformację komórki b w odpowiadające jej wyrażenie procesowe:

$$\phi(b) = \begin{cases} b & \Psi(b) = PRIM \\ b[< \phi(\delta(b)) :: \Xi_b >] & \Psi(b) = XOR \\ b[\prod_{<b,b'> \in \rho} < \phi(b') :: \{b' = \mathbf{0}\} >] & \Psi(b) = AND \end{cases}$$

Ξ_b — specyfikacja rekursywna — jest zbiorem definicji procesowych:

$$\Xi_b = \bigcup_{<b,b'> \in \rho} \{b' = \Phi_{b'}\}$$

gdzie ciało $\Phi_{b'}$ procesu b' jest zdefiniowane następująco. Jeśli nie istnieje żaden łuk wychodzący z komórki b' wtedy ciało $\Phi_{b'}$ jest reprezentowane przez pusty proces **0**, w przeciwnym przypadku reprezentowane jest przez następujące wyrażenie procesowe:

$$\Phi_{b'} = \sum_{<b',a,b''> \in Arc} a;\phi(b'')$$

Funkcja ϕ zdefiniowana tutaj różni się od tej z [67] dla komórek typu *AND*.

Transformacja startowej konfiguracji strukturalnej B_{init} Mapy Stanów jest zdefiniowana jako:

$$\phi(B_{init}) =< \phi(r) \mid \{r = \mathbf{0}\} >$$

gdzie $\phi(B_{init})$ przeciąża funkcję ϕ, natomiast r jest korzeniem B.

Uwaga

Zbiór wyrażeń procesowych jest bogatszy od zbioru konfiguracji strukturalnych Map Stanów, co oznacza, że istnieją takie wyrażenia procesowe, które nie reprezentują żadnych konfiguracji strukturalnych Map Stanów. Na przykład wyrażenie procesowe:

$$(a_1; P_1 \parallel a_2; P_2) + a_3; P_3$$

nie może być interpretowane jako poprawna konfiguracja strukturalna, ponieważ składnik $(a_1; P_1 \parallel a_2; P_2)$ w powyższym wyrażeniu powinien być prefiksowany łukiem. Ponadto, procesy $a_1; P_1$ i $a_2; P_2$ powinny być interpretowane jako podstany ortogonalne pewnego stanu typu *AND*, którego cały składnik $(a_1; P_1 \parallel a_2; P_2)$ powinien być uszczegółowieniem. Dodatkowo jeszcze procesy odpowiadające podstanom ortogonalnym nie mogą być prefiksowane gdyż nie jest dopuszczalne wejście do podstanów stanu typu *AND*.

Przykład (cd.)

Przyjmijmy, że Mapa Stanów z rys. 5.1 jest w stanie startowym. Jej konfigurację strukturalną możemy zapisać w terminologii wyrażeń procesowych następująco:

ϕ(S) = <ϕ(RT) :: {RT = **0**}>

ϕ(RT) = RT[<ϕ(R) :: Ξ_{RT} >]

ϕ(R) = R[<ϕ(NET) :: {NET=**0**}> || <ϕ(COMP1) :: {COMP1=**0**}> ||
<ϕ(COMP2) :: {COMP2=**0**}>]

ϕ(NET) = NET[<WAIT :: Ξ_{NET}>]

ϕ(COMP1) = COMP1[<LOCAL :: Ξ_{COMP1}>]

ϕ(COMP2) = COMP2[<WT :: Ξ_{COMP2}>]

Specyfikacje rekursywne są następujące:

Ξ_{RT} = {R = a7; FAILURE}

Ξ_{NET} = {WAIT = a1; TRANS} $\cup$ {TRANS = a2; WAIT}

Ξ_{COMP1} = {LOCAL = a3;XMIT} $\cup$ {XMIT = a4; LOCAL}

Ξ_{COMP2} = {WT = a5; RCVE} $\cup$ {RCVE = a6; WT}

Term ϕ(S) może być też zapisany jak poniżej:

ϕ(S) = < RT[< R[<NET[<WAIT::Ξ_{NET}>]::{NET=**0**}> ||

<COMP1[<LOCAL::Ξ_{COMP1}>] :: {COMP1=**0**}> ||

<COMP2[<WT::Ξ_{COMP2}>]::{COMP2=**0**}>] :: Ξ_{RT}>] :: {RT = **0**}>

lub w formie skróconej (bez specyfikacji rekursywnych) jako:

ϕ(S) =RT[R[NET[WAIT] || COMP1[LOCAL] || COMP2[WT]]]

5.2.5. Konfiguracje Markowowskich Map Stanów

Konfiguracja strukturalna B Markowowskiej Mapy Stanów w formie wyrażenia procesowego jest definiowana jako $P = \phi(B)$. Startowa konfiguracja Markowowskiej Mapy Stanów jest zdefiniowana jako $P_{init} = \phi(B_{init})$.

Nieformalnie, przejścia w Markowowskich Mapach Stanów zależą nie tylko od konfiguracji strukturalnej i środowiska zewnętrznego, które generuje zdarzenia zewnętrzne będące przyczyną przejść wzdłuż łuków, ale także od konfiguracji czasowej (zdefiniowanej poniżej) opisującej przyczyny powodujące przejścia w dziedzinie czasu.

Poniżej zdefiniowana będzie Konfiguracja Markowowskiej Mapy Stanów jako para składająca się z konfiguracji strukturalnej i czasowej.

Definicja 5.4 [5]

Konfiguracja czasowa T Markowowskiej Mapy Stanów w chwili τ dla konfiguracji strukturalnej *P* jest zdefiniowana jako trójka:

$$T(P) = <OA, IE, DE>$$

gdzie:

— $OA \subseteq ArcN$ jest zbiorem *nazw otwartych łuków*, które są relewantne do strukturalnej konfiguracji *P*,

— $IE \subseteq EVENTS$ jest zbiorem *dostępnych zdarzeń wewnętrznych*,

— $DE = <e_1,\lambda_1>, \ldots, <e_n,\lambda_n>$ jest listą *odroczonych zdarzeń wewnętrznych*, gdzie $\lambda_i \in R_+$ (dla i = 1, ... , n, i $n \geq 0$).

Startowa konfiguracja czasowa dla startowej konfiguracji strukturalnej P_{init} jest zdefiniowana następująco:

$$T(P_{init}) = <OA_{init}, IE_{init}, DE_{init}>$$

gdzie:

— $OA_{init} = \{a \mid a \in relevant(P_{init}), L(a).delay = \infty\}$

— $IE_{init} = \varnothing$

— $DE_{init} = < >$, gdzie $< >$ oznacza pustą listę.

Przykład (cd.)

Konfiguracją Markowowskiej Mapy Stanów S w stanie startowym i chwili $\tau = 0$ jest:

$<\phi(S), <\{a1,a3,a5\},\varnothing,<>>>$

Mapa Stanów zanurzona jest w *środowisku* określonym jako funkcja:

$$EE : TIME \rightarrow 2^{EVENTS}$$

gdzie *TIME* jest zbiorem chwil czasowych, a *EVENTS* jest zbiorem zdarzeń. Zbiór $EE(\tau)$ będzie nazywany *zbiorem zdarzeń zewnętrznych* w chwili τ. Jeśli nie będzie to powodowało niejednoznaczności, zbiór ten będzie zapisywany jako *EE*.

Suma zbiorów dostępnych zdarzeń zewnętrznych i wewnętrznych $EO = EE \cup IE$, nazywana jest *ofertą zdarzeń* w danej chwili.

Definicja 5.5

Konfiguracją Markowowskiej Mapy Stanów w danej chwili jest para:

$<P, T(P)>$

P jest konfiguracją strukturalną, a $T(P)$ jest konfiguracją czasową dla P.

Przyjmijmy, że na liście zdarzeń odroczonych dane zdarzenie może wystąpić tylko raz.

Założenie 5.1

Dla każdego zdarzenia $e_i \in EVENTS$ oraz konfiguracji $<P,T(P)> \in RCONF$, na liście zdarzeń odroczonych DE może wystąpić co najwyżej jedna para zawierająca zdarzenie e_i.

Zgodnie z wnioskiem 5.1, zbiór konfiguracji strukturalnych jest skończony. Wprost z definicji konfiguracji czasowej (def. 5.4) wynika, że zbiory IE oraz OA, jako podzbiory zbiorów skończonych, są skończone. Skończoność listy DE wynika z definicji Markowowskich Map Stanów (def. 5.3), skończoności zbioru zdarzeń oraz powyższego założenia. Reasumując można sformułować następujący wniosek.

Wniosek 5.2

Zbiór konfiguracji Markowowskiej Mapy Stanów jest skończony.

Zdefiniujmy teraz kilka pomocniczych funkcji używając indukcji strukturalnej nad wyrażeniami procesowymi. Funkcje te będą użyte przy definiowaniu semantyki przejść.

Funkcja $relevant(P)$ definiuje, dla danej konfiguracji strukturalnej $P \in PROC$, zbiór nazw łuków wychodzących z komórek należących do konfiguracji strukturalnej reprezentowanej wyrażeniem procesowym P.

Funkcja *enabled*(P,OA,EO), dla danej konfiguracji strukturalnej $P \in PROC$ i zbioru nazw otwartych łuków $OA \subseteq ArcN$, definiuje zbiór nazw łuków, które są wyzwolone przez ofertę zdarzeń $EO \subseteq EVENTS$ i wzdłuż których można dokonać przejścia do nowej konfiguracji strukturalnej. Dany łuk mający nazwę $a \in ArcN$ może być wyzwolony tylko wtedy gdy $L(a).trigger \subseteq EO$. Oczywiście zbiór *enabled*$(P,OA,EO) \subseteq$ *relevant*(P).

Funkcja *mprior*(P,OA,EO) wybiera ze zbioru *enabled*(P,OA,EO) maksymalny podzbiór składający się z parami ortogonalnych łuków o najwyższych priorytetach.

Wreszcie funkcja *remove*(P,OA,EO) określa taki podzbiór zbioru *relevant*(P), który zawiera nazwy łuków przeznaczonych do usunięcia, ze zbioru łuków otwartych, po wykonaniu przejścia z P do nowej konfiguracji strukturalnej, wzdłuż zbioru łuków *mprior*(P,OA,EO).

Poniższa tabela podaje formalne definicje omawianych funkcji poprzez indukcję strukturalną.

Tabela 5.1

P	*relevant*(P)	*enabled*(P,OA,EO)	*mprior*(P,OA,EO)	*remove*(P,OA,EO)
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$a;\ P_1$	$\{a\}$	**if** $a \in OA$ and $L(a).trigger \subseteq EO$ **then** $\{a\}$ **else** $\emptyset$ **fi**	**if** $a \in$ *enabled*(P,OA,EO) **then** $\{a\}$ **else** $\emptyset$ **fi**	*mprior*(P,OA,EO)
$P_1 + P_2$	*relevant*(P_1) $\cup$ *relevant*(P_2)	*enabled*(P_1,OA,EO) $\cup$ *enabled*(P_2,OA,EO)	**if** *mprior*$(P_1,OA,EO) \succ$ *mprior*(P_2,OA,EO) **then** *mprior*(P_1,OA,EO) **else** *mprior*(P_2,OA,EO) **fi** [1]	**if** *mprior*$(P,OA,EO) \neq \emptyset$ **then** *relevant*(P) **else** $\emptyset$ **fi**

[1] Zauważmy, że zgodnie z przyjętymi ograniczeniami na klasę wyrażeń procesowych (Założenie 5.3), każdy zbiór *mprior*(P_i, OA, EO) zawiera najwyżej jeden element a_i (i = 1, 2). Relacja *mprior*$(P_1,OA,EO) \succ$ *mprior*(P_2,OA,EO) jest spełniona wtedy i tylko wtedy gdy *mprior*$(P_i,OA,EO) = \{a_i\}$ dla i = 1, 2, i jednocześnie spełniona jest relacja $L(a_1).priority > L(a_2).priority$ albo gdy *mprior*$(P_1,OA,EO) \neq \emptyset$, i *mprior*$(P_2,OA,EO) = \emptyset$.

$P_1 \parallel P_2$	*relevant*(P_1) ∪ *relevant*(P_2)	*enabled*(P_1,*OA*,*EO*) ∪ *enabled*(P_2,*OA*,*EO*)	*mprior*(P_1,*OA*,*EO*) ∪ *mprior*(P_2,*OA*,*EO*)	*remove*(P_1,*OA*,*EO*) ∪ *remove*(P_2,*OA*,*EO*)
$P_1[P_2]$	*relevant*(P_1) ∪ *relevant*(P_2)	*enabled*(P_1,*OA*,*EO*) ∪ *enabled*(P_2,*OA*,*EO*)	**if** *mprior*(P_1,*OA*,*EO*)≠∅ **then** *mprior*(P_1,*OA*,*EO*) **else** *mprior*(P_2,*OA*,*EO*) **fi** [1]	**if** *mprior*(P_1,*OA*,*EO*)≠∅ **then** *relevant*(*P*) **else if** *mprior*(P_2,*OA*,*EO*) ≠ ∅ **then** *relevant*(P_2) **else** ∅ **fi fi** [2]
p	*relevant*(P_p) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej	*enabled*(P_p,*OA*,*EO*) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej	*mprior*(P_p,*OA*,*EO*) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej	*remove*(P_p) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej
<*p* ::Ξ>	*relevant*(P_p) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej Ξ	*enabled*(P_p,*OA*,*EO*) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej Ξ	*mprior*(P_p,*OA*,*EO*) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej Ξ	*remove*(P_p) gdzie $p = P_p$ jest równaniem ze specyfikacji rekursywnej Ξ

5.2.6. Przejścia między konfiguracjami

Badamy przejścia między konfiguracjami przy zadanym środowisku. Analogicznie do konfiguracji strukturalnych i czasowych będą także rozróżniane *przejścia strukturalne* i *czasowe*. Przejścia strukturalne opisują zmiany konfiguracji strukturalnych (odpowiadają one prejściom w Mapach Stanów), natomiast przejścia czasowe opisują zmiany konfiguracji czasowych. Oba rodzaje przejść są ściśle ze sobą powiązane. Oznacza to, że

[1] Definicje pokazują rolę priorytetu strukturalnego P_1 nad P_2. Jeśli *mprior*(P_1,*OA*,*EO*) jest zbiorem niepustym, to oznacza to, że istnieje wyzwolony łuk na wyższym poziomie hierarchii. W tym przypadku po wyjściu z P_1, wszystkie łuki relewantne do P_1 i wszystkie relewantne do P_2 powinny zostać usunięte ze zbioru nazw otwartych łuków *OA* (krok czwarty algorytmu w podrozdziale 6.4). Gdy natomiast *mprior*(P_1,*OA*,*EO*) jest zbiorem pustym wtedy tylko łuki relewantne do P_2 powinny zostać usunięte ze zbioru *OA*.

przejścia strukturalne mogą powodować zmiany konfiguracji czasowych, natomiast przejścia czasowe mogą wywoływać przejścia strukturalne.

Zgodnie z [54], grupujemy sekwencje przejść w kroki i mikrokroki. Krok definiuje się jako sekwencję mikrokroków, które zachodzą w jednej chwili. Pierwszy mikrokrok w sekwencji tworzącej krok zaczyna się pierwszym przejściem strukturalnym występującym po dowolnym przejściu czasowym. Ostatni mikrokrok w kroku jest tym, po którym nie jest możliwe wykonanie żadnego przejścia strukturalnego. Tak więc zarówno przed jak i bezpośrednio po wykonaniu kroku możliwe są tylko przejścia czasowe.

Niech $<P, T(P)>$, gdzie $T(P) = <OA, IE, DE>$, będzie konfiguracją Markowowskiej Mapy Stanów w chwili τ. Załóżmy, że żadne przejście strukturalne nie jest możliwe w chwili τ. W tej sytuacji może zajść przejście czasowe albo może pojawić się zdarzenie zewnętrzne.

Są dwie przyczyny przejść czasowych:

1. pojawienie się zdarzenia wewnętrznego jako wynik upływu czasu opóźnienia dla któregoś z listy zdarzeń odroczonych,
2. upłynięcie czasu opóźnienia pewnego łuku o niezerowym czasie co czyni ten łuk otwartym.

Niech $<e_i,\lambda_i>$ (dla pewnego i= 1, …, n) będzie elementem z listy zdarzeń odroczonych:

$DE = <<e_1,\lambda_1>, \ldots, <e_n,\lambda_n>>$

Pojawienie się wewnętrznego zdarzenia odroczonego $e_i \in EVENTS$ wywołuje następujące przejście czasowe zapisywane:

$T(P) = internal(e_i) \Rightarrow T(P)'$

lub $<P, T(P)> = internal(e_i) \Rightarrow <P, T(P)'>$

gdzie

$T(P)' = <OA', IE', DE'>$,

oraz

$OA' = OA$, $IE' = \{e_{ij}\}$, $DE' = <<e_1,\lambda_1>, \ldots, <e_{i-1},\lambda_{i-1}>,<e_{i+1},\lambda_{i+1}>, \ldots, <e_n,\lambda_n>>$.

Tak więc po tym przejściu czasowym $EO = \{e_{ij}\}$.

Upłynięcie czasu opóźnienia otwierające pewien łuk o nazwie $a \in ArcN$ i o niezerowym czasie otwarcia jest przyczyną przejścia czasowego zapisywanego następująco:

$T(P) = open(a) \Rightarrow T(P)'$

lub $<P, T(P)> = open(a) \Rightarrow <P, T(P)'>$

gdzie

$T(P)' = <OA', IE', DE'>$,

natomiast

$OA' = OA \cup \{a\}$, $IE' = IE$, $DE' = DE$.

W dalszych rozważaniach przyjmijmy następujące założenie.

Założenie 5.2

W danej chwili τ, zbiór zdarzeń zewnętrznych $EE(\tau)$ zawiera najwyżej jedno zdarzenie $e \in EVENTS$.

Wystąpienie zdarzenia zewnętrznego nie zmienia konfiguracji, a tylko ofertę zdarzeń. Generalnie, w danej chwili może wystąpić zdarzenie zewnętrzne i zdarzenia wewnętrzne, tak więc $EO = IE \cup EE$.

Opóźnienie θ wystąpienia pierwszego zdarzenia jest wartością minimalną realizacji następujących zmiennych losowych:

1. zmiennej o rozkładzie wykładniczym z parametrem:

   $\lambda = \lambda_1 + \ldots + \lambda_n$,

   reprezentującej czas wystąpienia zdarzenia spośród zdarzeń znajdujących się na liście $DE = <<e_1, \lambda_1>, \ldots, <e_n, \lambda_n>>$ zdarzeń odroczonych,

2. zmiennej o rozkładzie wykładniczym z parametrem:

   $\lambda = \sum_{a \in relevant(P) \setminus OA} L(a).delay$,

   reprezentującej czas kiedy jeden z jeszcze nieotwartych łuków relewantnych do konfiguracji strukturalnej *P*, czyli łuków ze zbioru $relevant(P) \setminus OA$, zostanie otwarty,

3. zmiennej reprezentującej czas pojawienia się zdarzenia zewnętrznego. Gdy i ta zmienna ma rozkład wykładniczy (zakładamy, że tak), wtedy Markowowska Mapa Stanów posiada własność Markowa.

Pojedyncze przejście strukturalne może pociągać za sobą przejście czasowe. Nawiązując do terminologii używanej w literaturze dotyczącej Map Stanów, parę składającą się z przejścia strukturalnego i następującego po nim czasowego nazywamy mikrokrokiem. Sekwencja mikrokroków jest nazywana krokiem jeśli wszystkie mikrokroki zachodzą w tym samym czasie i żaden więcej mikrokrok nie jest możliwy po ostatnim z sekwencji.

W dalszych rozważaniach przyjęte są następujące założenia często przyjmowane w literaturze dotyczącej Map Stanów.

Założenia 5.3

1. Czas trwania zdarzenia jest równy zero,
2. Zdarzenie oferowane w danej chwili jest natychmiast dostępne dla wszystkich łuków,
3. Czas trwania kroku, jak również dowolnego przejścia czasowego jest zerowy.

Algorytm zdefiniowany poniżej opisuje generację sekwencji mikrokroków tworzącej krok.

Algorytm 5.1 [5]

1. Zdefiniujmy zbiór *mprior*(*P*, *OA*, *EO*). Jest to maksymalny zbiór nazw parami ortogonalnych łuków o najwyższych priorytetach relewantnych do konfiguracji strukturalnej *P*.

2. Jeśli $mprior(P, OA, EO) \neq \varnothing$ wykonaj następny krok algorytmu, w przeciwnym przypadku zakończ algorytm.
3. Zdefiniujmy przejście strukturalne

$$P \text{—}mprior(P, OA, EO) \rightarrow P'$$

stosując następujący zbiór reguł przepisywania wyrażeń procesowych:

$$a; P_1 \text{—}\{a\}\rightarrow P_1 \qquad \textit{prefiksowanie}$$

$$\frac{P_1 \text{—}\{a\}\rightarrow Q_1}{P_1 + P_2 \text{—}\{a\}\rightarrow Q_1} \qquad \frac{P_2 \text{—}\{a\}\rightarrow Q_2}{P_1 + P_2 \text{—}\{a\}\rightarrow Q_2} \qquad \textit{reguły wyboru}$$

pod warunkiem, że $a \in mprior(P_1 + P_2, OA, EO)$,

$$\frac{P_1 \text{—}\{a_1\}\rightarrow Q_1}{P_1[P_2] \text{—}\{a_1\}\rightarrow Q_1} \qquad \frac{P_2 \text{—}\{a_2\}\rightarrow Q_2}{P_1[P_2] \text{—}\{a_2\}\rightarrow P_1[Q_2]} \qquad \textit{reguły uszczegóławiania}$$

pod warunkiem, że $a_i \in mprior(P_i, OA, EO)$ dla i = 1, 2,

$$\frac{P_1 \text{—}\{a\}\rightarrow Q_1}{P_1 \,||\, P_2 \text{—}\{a\}\rightarrow Q_1 \,||\, P_2} \qquad \frac{P_2 \text{—}\{a\}\rightarrow Q_2}{P_1 \,||\, P_2 \text{—}\{a\}\rightarrow P_1 \,||\, Q_2} \qquad \textit{reguły łączenia}$$

$$\frac{\begin{array}{c} P_1 \text{—}\{a_1,\ldots,a_n\}\rightarrow Q_1 \\ P_2 \text{—}\{b_1,\ldots,b_m\}\rightarrow Q_2 \end{array}}{P_1 \,||\, P_2 \text{—}\{a_1,\ldots,a_n,b_1,\ldots,b_m\}\rightarrow Q_1 \,||\, Q_2}$$

pod warunkiem, że

$a_1,\ldots, a_n \in mprior(P_1, OA, EO)$, i $b_1,\ldots, b_m \in mprior(P_2, OA, EO)$,

$$\frac{< P_p :: \Xi > \text{—}\{a\}\rightarrow Q}{< p :: \Xi > \text{—}\{a\}\rightarrow Q} \qquad (p = P_p) \in \Xi \qquad \textit{reguła rekursji}$$

pod warunkiem, że $a \in mprior(P_p, OA, EO)$.

Niech EO będzie ofertą zdarzeń w danej chwili τ. Oczywiście, $IE \subseteq EO$, gdzie IE — element $T(P)$. Przejście strukturalne z P do P' pociąga za sobą zmianę konfiguracji czasowej $T(P)=<OA,IE,DE>$ na $T(P')=<OA',IE',DE'>$, zgodnie z niżej zdefiniowanymi regułami.

4. Modyfikuj zbiór otwartych łuków:

$$OA' = OA \setminus remove(P, OA, EO) \cup \{a \in relevant(P') \mid L(a).delay = \infty\}$$

5. Wyczyść zbiór zdarzeń zewnętrznych $EE = \varnothing$.

6. Zdefiniujmy nowy zbiór dostępnych zdarzeń wewnętrznych:

$$IE' = \bigcup_{a \in mprior(P, OA, EO)} L(a).immediate$$

7. Zdefiniujmy nowy zbiór odroczonych zdarzeń wewnętrznych:

$$DE' = DE \wedge L(a_1).deferred \wedge \ldots \wedge L(a_n).deferred$$

gdzie

^ oznacza konkatenację list, a

$mprior(P, OA, EO) = \{a_1, \ldots, a_n\}$.

Dla danej chwili τ, oraz konfiguracji $<P, T(P)>$, po wykonaniu powyższych operacji osiągana jest nowa konfiguracja $<P', T(P')>$. Zarówno $<P, T(P)>$, jak i $<P', T(P')>$ odnoszą się do tej samej chwili. Przejście:

$$<P, T(P)> = micro\text{-}step(EO) \Rightarrow <P', T(P')>$$

gdzie $P \text{---} mprior(P, OA, EO) \rightarrow P'$

reprezentuje mikrokrok.

8. Jeśli $T(P') = <OA', IE', DE'>$ i zbiór oferowanych zdarzeń $EO \neq \varnothing$ przejdź do pierwszego punktu algorytmu w celu określenia kolejnego mikrokroku, w przeciwnym przypadku krok jest zakończony.

Niech $<P_0, T(P_0)>$ będzie konfiguracją osiągniętą w wyniku przejścia czasowego i niech

$$\begin{aligned} <P_0, T(P_0)> &= micro\text{-}step(EO_0) \Rightarrow <P_1, T(P_1)> \\ &= micro\text{-}step(EO_1) \Rightarrow <P_2, T(P_2)> \\ &\ldots \\ &= micro\text{-}step(EO_{n-1}) \Rightarrow <P_n, T(P_n)> \end{aligned}$$

gdzie

$$P_0 \text{ —}mprior(P_0, OA_0, EO_0)\rightarrow P_1 \ldots \text{—}mprior(P_{n-1}, OA_{n-1}, EO_{n-1})\rightarrow P_n$$

jest taką sekwencją mikrokroków, że $<P_n, T(P_n)>$ jest konfiguracją, z której nie jest możliwe żadne więcej przejście strukturalne. Sekwencja ta tworzy krok, który będzie symbolicznie zapisywany jako:

$$<P_0, T(P_0)> = step(EO_0) \Rightarrow <P_n, T(P_n)>$$

gdzie $<P_0, T(P_0)>$ i $<P_n, T(P_n)>$ są nazywane odpowiednio *startową* i *kończącą konfiguracją* kroku.

Jest możliwe utworzenie nieskończonej sekwencji mikrokroków zachodzących w jednej chwili. Załóżmy, że istnieje pętla składająca się z łuków natychmiastowych takich, że pierwszy z nich generuje natychmiastowe zdarzenie wyzwalające drugi itd. Pętla taka jest potencjalnie nieskończona.

Łatwo jest sformułować warunek wystarczający do uniknięcia takich nieskończonych kroków. Na przykład można zabronić konstruowania pętli składających się z samych łuków natychmiastowych. Utworzenie warunku koniecznego jest bardziej złożone.

Przy niektórych podejściach do definiowania semantyki Map Stanów nakładane są ograniczenia na funkcję kroku, które mają zapobiec wyżej opisanej niedogodności. Na przykład semantyka kroku zaproponowana w pracy [25] dopuszcza wykonanie tylko jednego łuku w każdym z ortogonalnych komponentów w ciągu jednego kroku. Peron w pracy [50] proponuje inne podejście, gdzie postuluje, że żaden łuk nie może być wykonany więcej niż jeden raz w danym kroku. W pracy niniejszej nie są nakładane żadne tego typu ograniczenia na funkcję kroku. Przyjmujemy jednak następujące założenie:

Założenie 5.4

Każdy krok składa się ze skończonej sekwencji mikrokroków.

Twierdzenie 5.1

Dla danej konfiguracji startowej $<P, T(P)>$ i oferty zdarzeń EO, krok

$<P,\ T(P)> = step(EO) \Rightarrow <P',\ T(P')>$

wyznacza konfigurację kończącą $<P',\ T(P')>$ w sposób jednoznaczny.

Dowód

Na wstępie zauważmy, że w trakcie trwania kroku nie jest możliwe wystąpienie zdarzenia zewnętrznego lub odroczonego ani też otwarcie jakiegoś łuku ponieważ prawdopodobieństwo wystąpienia każdego z tych wydarzeń w zerowym czasie jest zerowe, tak więc żaden z tych przypadków nie może wprowadzić niejednoznaczności do kroku.

Aby udowodnić tezę twierdzenia, wystarczy udowodnić dla każdego mikrokroku w poniższej sekwencji tworzącej krok:

$<P,\ T(P)> = micro\text{-}step(EO_0) \Rightarrow <P_1,\ T(P_1)>$

$= micro\text{-}step(EO_1) \Rightarrow <P_2,\ T(P_2)>$

.....

$= micro\text{-}step(EO_{n-1}) \Rightarrow <P_n,\ T(P_n)>,$

gdzie $P_n = P'$, że przejście strukturalne i odpowiadające jej przejście czasowe są jednoznacznie określone. Przejście strukturalne dla i-tego mikrokroku zapisujemy następująco:

$$<Pi,\ T(P_i)> \text{---} mprior(P_i,\ OA_i,\ EO_i) \rightarrow <P_{i+1},\ T(P_{i+1})> .$$

Rozważmy teraz relację $mprior(P_i,\ OA_i,\ EO_i)$ zdefiniowaną w tabeli 5.1. Dla danej konfiguracji, zbioru otwartych łuków oraz oferty zdarzeń mamy kilka przypadków zależnych od operatorów algebry StPA.

1° Operatory procesu pustego oraz prefiksowania łuku w oczywisty sposób nie wprowadzają niejednoznaczności.

2° Operator wyboru P_1+P_2 bazuje na relacji priorytetu $\succ$. Założenie 5.2 zapewnia, że dla każdej pary P_1, P_2 jedna z relacji:
$mprior(P_1,OA,EO) \succ mprior(P_2,OA,EO)$ lub
$mprior(P_2,OA,EO) \succ mprior(P_1,OA,EO)$ jest spełniona jednoznacznie. Jest to prawdą także dla dowolnej sekwencji operatorów wyboru.

3° Operator złożenia równoległego $P_1 \| P_2$ definiuje relację *mprior* jako sumę mnogościową tych relacji dla komponentów P_1 i P_2. Wynika to z założenia, że zawsze jest brany maksymalny zbiór otwartych ortogonalnych łuków. Dowolna sekwencja operatorów złożenia równoległego tworzy taki sam zbiór łuków. Jest tak z powodu wzajemnej ortogonalności wybranych łuków.

4° Definicja relacji *mprior* dla operatora uszczegółowienia $P_1[P_2]$ przy danej ofercie zdarzeń także daje jednoznacznie określony zbiór łuków poprzez określenie priorytetu strukturalnego. Ponieważ łuki nie mogą przekraczać granic stanów, priorytet ten może być zdefiniowany jednoznacznie.

5° Specyfikacje rekursywne nie powodują niejednoznaczności.

Jednoznacznie zdefiniowana relacja *mprior* daje unikalny zbiór łuków, które mają być wykonane w czasie mikrokroku co oznacza, że przejście strukturalne i mikrokrok są określone jednoznacznie.

Przejście strukturalne z P_i to P_{i+1} pociąga za sobą zmianę konfiguracji czasowej $T(P_i) = <OA_i, IE_i, DE_i>$ na $T(P_{i+1}) = <OA_{i+1}, IE_{i+1}, DE_{i+1}>$.

Nowy zbiór otwartych łuków OA_{i+1} jest wyznaczony przez relacje *remove* i *relevant*. Relacja *relevant* zależna jest tylko od konfiguracji strukturalnej, a więc jest jednoznaczna. Dowód jednoznaczności relacji *remove* jest analogiczny do powyższego dla relacji *mprior*.

Zbiory IE_{i+1} i DE_{i+1} są konstruowane z użyciem etykiet wszystkich łuków należących do bieżącego przejścia strukturalnego, a więc są przez przejście jednoznacznie zdefiniowane.

Jak widać więc pojedynczy mikrokrok i towarzysząca mu zmiana konfiguracji czasowej są określone jednoznacznie. Ponieważ w trakcie trwania kroku nie może pojawić się zdarzenie zewnętrzne bądź odroczone, oferta zdarzeń określona jest jednoznacznie przez konfigurację czasową

osiągniętą w i-tym mikrokroku. Pozwala to na jednoznaczne określenie końca kroku. Tak więc krok wyznaczony jest w sposób jednoznaczny. *QED*

Przykład (cd.)

Niech Markowowska Mapa Stanów z rys. 5.1, otrzyma w chwili τ, zdarzenie "send". Zastosujmy wyżej opisany algorytm. Najpierw definiujemy zbiór *mprior*(*P*,*OA*,*EO*) = {a3}. Ponieważ nie jest on pusty, może zajść przejście strukturalne:

ϕ(S) —{a3}→ *P'*,

gdzie

P' = RT [R [NET [WAIT] || COMP1 [XMIT] || COMP2 [WT]]].

Elementy konfiguracji czasowej *T*(*P'*) = <OA',IE',DE'> będą następujące:

OA' = {a1, a4, a5}, *IE'* = {srq}, *DE'* = < >.

Ponadto *EE* = ∅. Teraz, ponieważ *IE'* ≠∅, wykonany zostanie następny mikrokrok dając:

P'—{a1}→ *P''*,

gdzie

P'' = RT [R [NET [TRANS] || COMP1 [XMIT] || COMP2 [WT]]],

oraz

T (*P''*) = <OA",IE",DE"> = <{a4, a5}, ∅, <>>.

Teraz, ponieważ zbiory *EE* i *IE''* są puste, krok się kończy. Następne przejście będzie następujące:

— upłynięcie czasu opóźnienia a2 wywołujące przejście czasowe:

T (*P''*) =*open*(a2)⇒ *T* (*P'''*)

5.3. Łańcuch Markowa z czasem ciągłym

5.3.1. System przejść

Określmy Etykietowany System Przejść $LTS(MSC)$ dla Markowowskich Map Stanów (MSC).

Definicja 5.6 [6]

Etykietowany System Przejść Markowowskiej Mapy Stanów jest to trójka:

$LTS(MSC) = <CONF, =LAB\Rightarrow, C_{init}>$

gdzie:

$CONF$ jest zbiorem konfiguracji: $\{<P,T(P)> \mid P \in PROC\}$,

$=LAB\Rightarrow \subseteq CONF \times CONF$, jest relacją przejść zdefiniowaną jako suma logiczna:

$\{=internal(e)\Rightarrow \mid e \in EVENTS\} \cup \{=open(a)\Rightarrow \mid a \in ArcN\} \cup$

$\cup \{=step(EO)\Rightarrow \mid EO \subseteq EVENTS\}$,

natomiast $C_{init} = <P_{init}, T(P_{init})>$

Zbiór trójek tworzących Etykietowany System Przejść zawiera wiele elementów, które odpowiadają konfiguracjom nieosiągalnym z konfiguracji startowej. Zdefiniujmy więc jego podzbiór zawierający tylko te, do których można dojść z konfiguracji C_{init} poprzez skończoną liczbę przejść. Zanim to jednak nastąpi, określmy postać przejścia w tym ograniczonym Etykietowanym Systemie Przejść. Zdefiniujmy najpierw operator składania przejść:

$\circ := =LAB\Rightarrow \times =LAB\Rightarrow \rightarrow =LAB\Rightarrow$.

Definicja 5.7

Niech $=lab_1\Rightarrow, =lab_2\Rightarrow \in =LAB\Rightarrow$ oraz $C_1, C_2, C_3 \in CONF$. Ponadto zachodzi następująca zależność: $C_1 =lab_1\Rightarrow C_2 =lab_2\Rightarrow C_3$. Przejście wynikowe zapisujemy jako: $C_1 =lab_1\Rightarrow \circ =lab_2\Rightarrow C_3$. Jeśli oba przejścia zachodzą w tym samym czasie, to konfigurację C_2 nazywamy *niestabilną*.

W chwili τ możliwe jest przejście czasowe wskutek pojawienia się zdarzenia odroczonego lub otwarcia łuku, albo przejście strukturalne czyli krok wskutek pojawienia się zdarzenia zewnętrznego. Każde przejście czasowe pociąga za sobą, być może puste, przejście strukturalne. Przyjrzyjmy się bliżej takim parom przejść.

Najpierw zajmijmy się przejściem związanym z pojawieniem się zdarzenia odroczonego $e \in EVENTS$. Wywołuje ono przejście czasowe etykietowane jako $internal(e)$ oraz następujący po niej krok z etykietą $step(\{e\})$. Parametrem kroku jest jednoelementowy zbiór $EO=\{e\}$ gdyż pojawienie się drugiego zdarzenia, zewnętrznego lub wewnętrznego, w zerowym czasie ma zerowe prawdopodobieństwo. Ze względu na zerowy czas życia zdarzeń nie ma także zdarzeń dawniejszych. Mamy więc:

$$<P,T(P)> = internal(e) \Rightarrow <P,T(P)'> = step(\{e\}) \Rightarrow <P',T(P')>$$

z niestabilną konfiguracją $<P,T(P)'>$. Wprowadźmy przejście złożone:

$$<P,T(P)> = internal(e) \Rightarrow^{\circ} = step(\{e\}) \Rightarrow <P',T(P')>$$

oznaczając: $= int(e) \Rightarrow = = internal(e) \Rightarrow^{\circ} = step(\{e\}) \Rightarrow$.

Podobna sytuacja wystąpi w przypadku otwarcia łuku. Jest to przejście opatrzone etykietą $open(a)$. Bezpośrednio po nim następuje krok etykietowany jako $step(\varnothing)$. Oferta zdarzeń dla tego kroku jest zbiorem pustym gdyż jednocześnie z otwarciem łuku nie może pojawić się ani zewnętrzne, ani wewnętrzne zdarzenie. Podobnie jak poprzednio zapisujemy:

$$<P,T(P)> = open(e) \Rightarrow <P,T(P)'> = step(\{e\}) \Rightarrow <P',T(P')>$$

z niestabilną konfiguracją $<P,T(P)'>$. Wprowadźmy przejście złożone:

$$<P,T(P)> = open(e) \Rightarrow^{\circ} = step(\{e\}) \Rightarrow <P',T(P')>$$

oznaczając: $= op(e) \Rightarrow = = open(e) \Rightarrow^{\circ} = step(\{e\}) \Rightarrow$.

Na początku samodzielnego, czyli niepoprzedzonego przejściem czasowym, kroku oferta zdarzeń może zawierać tylko jedno zdarzenie zewnętrzne. Możemy więc takie przejście zapisać jako $<P,T(P)> = step(\{e\}) \Rightarrow <P',T(P')>$.

Na koniec ograniczmy zbiór konfiguracji do tych, które mogą być osiągnięte z konfiguracji startowej C_{init}.

$RCONF=\{C \mid C \in CONF, C_{init} =lab_1\Rightarrow\circ\ldots\circ=lab_n\Rightarrow C\}$,

gdzie $=lab_i\Rightarrow$ jest jedną z wyżej zdefiniowanych przejść.

Definicja 5.8 [6]

Ograniczony Etykietowany System Przejść Markowowskiej Mapy Stanów jest trójką:

$RLTS(MSC) = <RCONF, =RLAB\Rightarrow, C_{init}>$

gdzie:

$RCONF$ jest zbiorem konfiguracji osiągalnych z konfiguracji C_{init},

$=RLAB\Rightarrow = (\{=int(e)\Rightarrow \mid e\in EVENTS\} \cup \{=op(a)\Rightarrow \mid a\in ArcN\} \cup$

$\cup \{=step(\{e\})\Rightarrow \mid e\in EVENTS\}) \cap (RCONF \times RCONF)$

jest relacją przejść ograniczoną do konfiguracji osiągalnych z konfiguracji startowej,

C_{init} jest konfiguracją startową.

5.3.2. Wyprowadzenie łańcucha Markowa

Wyżej zdefiniowany System Przejść możemy traktować jak dyskretny proces stochastyczny z ciągłym czasem. Jak pokazano wcześniej (wn. 5.2), zbiór stanów tego procesu jest skończony. Zdefiniujmy teraz, zgodnie z [6], diagram przejść łańcucha Markowa skojarzonego z wyżej określonym Etykietowanym Systemem Przejść.

- Węzłami diagramu są elementy skończonego zbioru konfiguracji $RCONF$.
- Węzłem startowym jest konfiguracja C_{init}.
- Łuki odpowiadają relacji przejść $=RLAB\Rightarrow$.
- Każdy łuk opatrzony jest etykietą postaci $<ea,\lambda>$,

  gdzie $ea \in EVENTS \cup ArcN$ i w zależności od rodzaju przejścia jest nazwą zdarzenia bądź łuku. Parametr λ jest intensywnością przejścia wzdłuż danego łuku i jest określony przez funkcję etykietującą Marko-

wowską Mapę Stanów lub przez rozkład prawdopodobieństwa dla zdarzeń zewnętrznych.

Zauważmy, że w Ograniczonym Etykietowanym Systemie Przejść wszystkie przejścia są czasowe, czyli wyrażone skończoną wartością parametru λ.

Jak już powiedziano, rozważany proces Markowa jest skończenie-stanowy, można więc zbudować macierz prawdopodobieństw przejść dla uzyskanego łańcucha Markowa. Wprowadźmy najpierw pomocniczą macierz Λ intensywności przejść. Jeżeli istnieje przejście z konfiguracji C_i do C_j, to element λ_{ij} jest równy intensywności tego przejścia, w przeciwnym wypadku $\lambda_{ij}=0$.

Macierz prawdopodobieństw przejść oznaczamy jako **W**, a jej elementy dla konfiguracji niepochłaniających są określane następująco:

$$w_{ij} = \frac{\lambda_{ij}}{\sum_{k=1}^{n} \lambda_{ik}}, \text{ gdzie } n \text{ jest rozmiarem macierzy.} \qquad (5.1)$$

Dla konfiguracji pochłaniających:

$$w_{ij} = \begin{cases} 1, gdy\ i = j \\ 0, gdy\ i \neq j \end{cases}.$$

Istnienie konfiguracji pochłaniających uniemożliwia badanie cyklicznego zachowania systemu natomiast możliwe jest uzyskanie wyników dla przypadku acyklicznego pod warunkiem wybrania konfiguracji pochłaniającej jako końcowej.

Przywołajmy znów przykład rozważany we wcześniejszej części rozdziału.

Przykład cd.

Analizując przejścia oraz możliwe konfiguracje Mapy Stanów z rys. 5.1 zgodnie z podanymi regułami konstrukcji łańcucha Markowa, otrzymujemy następujący Ograniczony Etykietowany System Przejść.

— Elementami zbioru *RCONF* są następujące konfiguracje:

C1= <RT [R [NET [WAIT] || COMP1 [LOCAL] || COMP2 [WT]]],
<{a1,a3,a5},∅,<>>>

C2= <RT [R [NET [TRANS] || COMP1 [XMIT] || COMP2
[WT]]],<{a4,a5},∅,<>>>

C3= <RT [R [NET [WAIT] || COMP1 [XMIT] || COMP2 [RCVE]]],
<{a1,a4},∅,<sack,9.3>>>

C4= <RT [R [NET [WAIT] || COMP1 [XMIT] || COMP2 [WT]]],
<{a1,a4,a5},∅,<sack,9.3>>>

C5= <RT [R [NET [WAIT] || COMP1 [LOCAL] || COMP2 [RCVE]]],
<{a1,a3},∅,<>>>

C6= <RT [R [NET [TRANS] || COMP1 [XMIT] || COMP2 [RCVE]]],
<{a4},∅,<>>>

C7= <RT [FAILURE], <∅,∅,<sack,9.3>>>

C8= <RT [FAILURE], <∅,∅,<>>>

— Konfiguracja startowa C_{init} = C1.

— Zbiór =*RLAB*⇒ zawiera następujące przejścia:

C1 =*step*({*send*})⇒ C2 C1 =*step*({*fail*})⇒ C8

C2 =*op*(*a2*)⇒ C3 C2 =*step*({*fail*})⇒ C8

C3 =*op*(*a6*)⇒ C4 C3 =*int*(*sack*)⇒ C5 C3 =*step*({*fail*})⇒ C7

C4 =*int*(*sack*)⇒ C1 C4 =*step*({*fail*})⇒ C7

C5 =*step*({*send*})⇒ C6 C5 =*op*(*a6*)⇒ C1 C5 =*step*({*fail*})⇒ C8

C6 =*op*(*a2*)⇒ C3 C6 =*op*(*a6*)⇒ C2 C6 =*step*({*fail*})⇒ C8

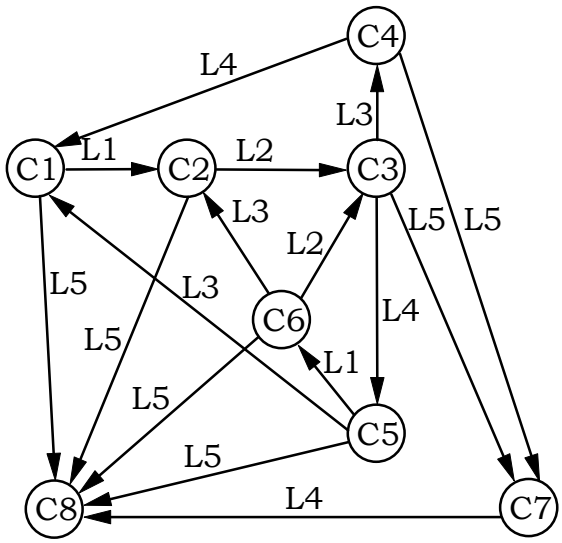
C7 =*int*(*sack*)⇒ C8

Do zdefiniowania łańcucha Markowa potrzebne są jeszcze pary <*ea*,λ> etykietujące łuki. Etykiety są zestawione w poniższej tabeli:

Tabela 5.2

nazwa	L1	L2	L3	L4	L5
ea	*send*	*a2*	*a6*	*sack*	*fail*
λ	λ_s	1	5	9.3	λ_f

gdzie λ_f i λ_s oznaczają odpowiednio intensywności dla zdarzeń zewnętrznych *fail* i *send*. Są one zależne od środowiska Markowowskiej Mapy Stanów.
Dla tak określonego procesu Markowa możemy wykreślić diagram przedstawiony na rysunku 5.2.



Rys. 5.2 Przykładowy diagram procesu Markowa

5.4. Czasowe charakterystyki badanego systemu

Uzyskany w poprzednim punkcie łańcuch Markowa umożliwia badanie charakterystyk czasowych systemu specyfikowanego za pomocą Map Stanów. Rozważymy teraz dwa przypadki najczęściej występujące w praktyce. Pierwszy z nich to przypadek acykliczny, gdzie system przetwarza jakieś dane czy zgłoszenia osiągając pewien stan końcowy. Tutaj interesuje nas średni czas przetwarzania. Drugi przypadek — cykliczny polega na przetwarzaniu zgłoszeń w pętli. Tu interesujący jest średni czas wykonania pętli zwany *średnim czasem cyklu systemu*.

5.4.1. Przypadek acykliczny

Przypadek ten polega przejściu systemu od konfiguracji startowej do końcowej. W celu wyliczenia charakterystyki czasowej określmy najpierw

macierz $\overline{\mathbf{W}}$. Uzyskujemy ją z macierzy $\mathbf{W}$ poprzez wykreślenie wiersza i kolumny odpowiadających stanowi końcowemu. Używane są tu określenia "stany" gdyż rozważamy łańcuch Markowa. Pamiętać jednak należy, że odpowiadają one ściśle konfiguracjom wyjściowej Mapy Stanów. Dla wygody przenumerujmy stany łańcucha Markowa zmniejszając o jeden, numer każdego stanu o indeksie większym od oznaczenia stanu końcowego. Podnosząc macierz $\overline{\mathbf{W}}$ do potęgi uzyskujemy opis średniej ilości wystąpień poszczególnych stanów. Ściślej, element $\overline{\mathbf{w}}_{ij}$ macierzy $\overline{\mathbf{W}}^{\mathbf{m}}$ oznacza średnią ilość wystąpień stanu *j* na pozycji *m+1* w sekwencji rozpoczynającej się od stanu *i*. Określmy teraz macierz $\mathbf{Z}$ tak, aby element $\mathbf{z}_{ij}$ oznaczał średnią ilość wystąpień stanu *j* na dowolnej pozycji w sekwencji rozpoczynającej się od stanu *i*.

$$\mathbf{Z} = \sum_{m=0}^{\infty} \overline{\mathbf{W}}^{m} \tag{5.2}$$

Szereg ten jest zbieżny do $\mathbf{Z} = \left(\mathbf{I} - \overline{\mathbf{W}}\right)^{-1}$, gdzie $\mathbf{I}$ jest macierzą jednostkową o rozmiarze równym rozmiarowi $\overline{\mathbf{W}}$.

Czas pobytu systemu w stanie *i* opisany jest zmienną losową o rozkładzie wykładniczym z parametrem

$$\tilde{\lambda}_i = \sum_{j=1}^{n-1} \lambda_{ij} ,$$

gdzie *n* jest liczbą stanów łańcucha Markowa. Wartość oczekiwana tej zmiennej, czyli średni czas pobytu systemu w stanie *i* wynosi $ET_i = 1/\tilde{\lambda}_i$. Poszukiwany średni czas dojścia systemu ze stanu startowego *i* do stanu końcowego dany jest następującą sumą:

$$\tau_i = \sum_{k=1}^{n-1} z_{ik} ET_k \ . \tag{5.3}$$

5.4.2. Przypadek cykliczny

Przypadek ten polega na przetwarzaniu przez system zgłoszeń lub danych w pętli. Oznacza to, że stan startowy osiągalny jest z niezerowym prawdopodobieństwem z każdego stanu z niego osiągalnego. Tak jak poprzednio stany łańcucha Markowa są konfiguracjami wyjściowej Mapy Stanów. Interesuje nas *stan stacjonarny* procesu. Rozkład prawdopodobieństw przebywania systemu w poszczególnych stanach uzyskujemy rozwiązując następujące równanie macierzowe: **Y**=**YW**. Ponieważ jest to układ równań liniowo zależnych, więc musimy uwzględnić jeszcze dodatkowy warunek: $\sum_{i=1}^{n} y_i = 1$.

Wykorzystując wektor **Y** oraz określony w poprzednim punkcie ET_i możemy wyliczyć *ułamek czasu* jaki system spędza w stanie *i*.

$$v_i = \frac{ET_i y_i}{\sum_{k=1}^{n} ET_k y_k}, \tag{5.4}$$

gdzie *n* jest ilością stanów łańcucha Markowa czyli ilością różnych stabilnych konfiguracji Mapy Stanów.

Dla każdego stanu w cyklu możemy określić średnią liczbę przejść przez niego pomiędzy dwoma kolejnymi przejściami przez stan odniesienia. Niech stanem odniesienia będzie stan *i*, średnia ilość przejść przez stan *j* dana jest wzorem:

$$q_{ji} = \frac{y_j}{y_i}.$$

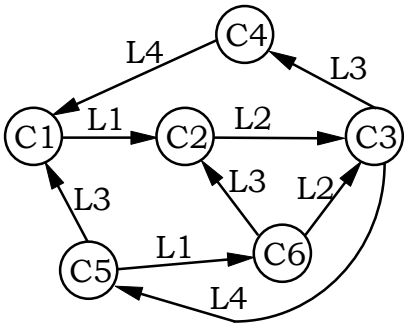
Średni czas cyklu przy stanie odniesienia *i* określa następujące równanie:

$$\gamma_i = \sum_{k=1}^{n} q_{ki} ET_k \,. \tag{5.5}$$

Uzyskany tu średni czas cyklu pozwala na dokonanie oceny wydajności systemu reaktywnego. Zdefiniowany także wyżej ułamek czasu spędzonego przez system w danym stanie pozwala na identyfikację tych stanów, na które trzeba zwrócić szczególną uwagę przy optymalizacji systemu gdyż mają największy wpływ na wydajność całego systemu.

5.4.3. Przykład obliczeniowy

Weźmy znów pod uwagę przykład Mapy Stanów analizowanej w tym rozdziale. Ponieważ interesować nas będzie cykliczne zachowanie układu w czasie normalnej pracy, usuńmy komórkę FAILURE wraz z łukiem a7. Komórka ta modeluje stan awarii systemu. Przypadkiem tym nie będziemy się teraz zajmować. Uzyskany łańcuch Markowa, przedstawiony na poniższym diagramie, redukuje się do sześciu stanów C1-C6 odpowiadającym konfiguracjom wyjściowej Mapy Stanów. Etykiety przejść są zgodne z tabelą 5.2 zamieszczoną na stronie 85.



Rys. 5.3 Zredukowany diagram Łańcucha Markowa

Powyższy, tak etykietowany, diagram opisuje łącze danych, gdzie średni czas transmisji danych przez sieć wynosi, w pewnych jednostkach, $1/\lambda_t=1$, średni czas transmisji potwierdzenia od sieci równy jest $1/\lambda_a=1/9{,}3$, a średni czas odbioru informacji przez drugi komputer wynosi $1/\lambda_r=1/5$. Cały system

zanurzony jest w środowisku generującym poissonowski potok sygnałów "*send*" ze średnim czasem oczekiwania równym $1/\lambda_s$. Macierze intensywności oraz prawdopodobieństw przejść, wyliczonych zgodnie ze wzorem (5.1), dla tego systemu są następujące:

$$\ddot{\mathbf{E}} = \begin{bmatrix} 0 & \lambda_s & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 9{,}3 & 0 \\ 9{,}3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & \lambda_s \\ 0 & 5 & 1 & 0 & 0 & 0 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}35 & 0{,}65 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 5/\lambda_s + 5 & 0 & 0 & 0 & 0 & \lambda_s/\lambda_s + 5 \\ 0 & 0{,}83 & 0{,}17 & 0 & 0 & 0 \end{bmatrix}.$$

Na początek zajmijmy się cyklicznym zachowaniem systemu. Zauważmy, że transmisja danych może się rozpocząć w konfiguracji **C1** lub **C5**. Ponieważ nie jest to jedna konfiguracja, musimy znaleźć taką, przez którą system przechodzi w czasie każdej transmisji. Konfiguracją tą jest **C3**, ją więc wybierzemy jako konfigurację odniesienia. Przy podanych parametrach czas cyklu systemu względem konfiguracji **C3** wynosi, zgodnie ze wzorem (5.5), $\gamma_3=1{,}1075+1/\lambda_s$. Ułamki czasu pobytu systemu w poszczególnych konfiguracjach wyrażają się dość skomplikowanymi wzorami. Nie będą więc tu przytoczone w postaci symbolicznej. Zamiast nich poniższa tabela przedstawia wyniki dla wybranych wartości parametru λ_s. Dla tych samych wartości podany jest też średni czas, γ_3, cyklu systemu.

Tabela 5.3

λ_s	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	γ_3
1	0.42	0.47	0.033	0.018	0.051	0.0086	2.11
0.1	0.89	0.090	0.0063	0.0034	0.011	0.00019	11.11
0.01	0.99	0.0099	0.00069	0.00037	0.0013	2.14e-6	101.1

Zauważmy, że czas cyklu systemu jest większy od średniego czasu oczekiwania na kolejny sygnał "*send*". Sugeruje to, że sygnały mogą być tracone. Obliczmy prawdopodobieństwo utraty sygnału "*send*". Jest ono równe sumie ułamków czasu przebywania systemu w konfiguracjach **C2**, **C3**,

C4, **C6** i wynosi 1-(1/(1,1075*λ_s+1)). Maleje ono wraz z parametrem λ_s. Na przykład chcemy, aby prawdopodobieństwo utraty "*send*" było mniejsze od 10^{-4}, musimy w tym celu zapewnić aby $\lambda_s<9{,}03*10^{-5}$. Wynika stąd, że czas odpowiedzi systemu, rozumiany jako czas od rozpoczęcia obsługi zdarzenia do momenty gdy możliwa jest obsługa następnego, musi być dużo krótszy od średniego czasu oczekiwania na sygnał zewnętrzny. Co natomiast należy zrobić, gdy mamy zadany średni czas oczekiwania na zdarzenie "*send*" i dopuszczalne prawdopodobieństwo jego utraty? Należy, oczywiście, skrócić czas obsługi tego zdarzenia, jednak konkretnej odpowiedzi na powyższe pytanie poszukajmy analizując acykliczne zachowanie systemu transmisji danych.

Załóżmy, że w pewnym momencie system był gotowy do przyjęcia i obsłużenia sygnału "*send*", i że sygnał ten się pojawił. System jest teraz w konfiguracji **C2** lub **C6**. Czas obsługi jest to czas dojścia systemu do jednej z konfiguracji **C1** lub **C5** traktowanych jako końcowe. Ponieważ mamy dwie konfiguracje końcowe, utwórzmy ich agregat traktując go jako konfigurację końcową. Aby obliczyć czas obsługi, wykreślmy najpierw z macierzy **W** wiersze i kolumny o numerach 1 i 5 oraz znajdźmy, korzystając ze wzoru (5.2), macierz **Z**.

$$\overline{\mathbf{W}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0{,}35 & 0 \\ 0 & 0 & 0 & 0 \\ 0{,}83 & 0{,}17 & 0 & 0 \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} 1 & 1 & \frac{\lambda_r}{\lambda_r+\lambda_a} & 0 \\ 0 & 1 & \frac{\lambda_r}{\lambda_r+\lambda_a} & 0 \\ 0 & 0 & 1 & 0 \\ \frac{\lambda_r}{\lambda_r+\lambda_t} & 1 & \frac{\lambda_r}{\lambda_r+\lambda_a} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0{,}35 & 0 \\ 0 & 1 & 0{,}35 & 0 \\ 0 & 0 & 1 & 0 \\ 0{,}83 & 1 & 0{,}35 & 1 \end{bmatrix}.$$

Zastosujmy teraz wzór (5.3) w celu wyliczenia średniego czasu dojścia systemu startującego z poszczególnych konfiguracji do konfiguracji końcowej czyli agregatu **C1** i **C5**.

$$\vec{\tau} = \left[\frac{1}{\lambda_a} + \frac{1}{\lambda_t} \;\vdots\; \frac{1}{\lambda_a} \;\vdots\; \frac{1}{\lambda_a} \;\vdots\; \frac{1}{\lambda_a} + \frac{1}{\lambda_t} \right]$$

Kolejne elementy tego wektora oraz kolumny i wiersze powyższych macierzy odpowiadają konfiguracjom **C2**, **C3**, **C4**, **C6** badanej Mapy Stanów. Ponieważ

system może startować z konfiguracji **C2** lub **C6**, powinniśmy wyliczyć sumę średnich czasów dojść ważoną prawdopodobieństwami pobytu w tych konfiguracjach znormalizowanymi do jedności. Ponieważ jednak oba interesujące nas czasy są równe, taki sam jest też średni czas dojścia systemu z dowolnej ze startowych konfiguracji do pierwszej z osiągniętych konfiguracji końcowych. Czas ten wynosi $\frac{1}{\lambda_a}+\frac{1}{\lambda_t}$, czyli przy założonych parametrach systemu, 1,1075. Wynik ten jest symetryczny względem czasów transmisji danych oraz potwierdzenia. Jeśli projektowany system nie spełnia wymagań czasowych, wtedy należy próbować zmniejszać jednocześnie oba czasy, przy czym najkorzystniej by było doprowadzić do ich równości aby niepotrzebnie nie inwestować w część systemu, która i tak nie poprawi całkowitej wydajności.

Zauważmy jeszcze, że w powyższym wyniku w ogóle nie jest uwzględniona szybkość komputera odbierającego. Nawet gdyby był on dużo wolniejszy od reszty składników systemu, i tak nie wpłynęłoby to na średni czas obsługi sygnału "*send*". Wynik powyższy pomógł więc wykryć błąd w projekcie systemu polegający na generowaniu potwierdzenia przez sieć niezależnie od tego czy dane zostały odebrane, czy nie. Po przeanalizowaniu projektu, okazuje się, że błędem jest przechodzenie od konfiguracji **C6** do **C3** gdyż wtedy nadawca otrzyma potwierdzenie odbioru mimo, że dane nie zostały odebrane. Prawdopodobieństwo takiej sytuacji oznaczonej jako p(**C6**-**C3**) równe jest iloczynowi ułamka czasu pobytu systemu w konfiguracji **C6** i prawdopodobieństwa wyjścia z niej do **C3**. Dla przyjętych wartości parametrów modelu wynosi ono:

λ_s	1	0,1	0,01
p(**C6**-**C3**)	$1,4*10^{-3}$	$3,2*10^{-5}$	$3,6*10^{-7}$

Wróćmy teraz do postawionego przedtem pytania. Mamy nałożone następujące wymagania:

– średni czas oczekiwania na "*send*" $1/\lambda_s=1000$,

– dane są transmitowane co najmniej dwa razy wolniej niż potwierdzenie,

– prawdopodobieństwo utraty sygnału "*send*" $p_{lost}<10^{-4}$,

– prawdopodobieństwo utraty sygnału mimo potwierdzenia p(**C6**-**C3**)$<10^{-7}$.

Sprawdźmy najpierw czy system wymagania te spełnia. Okazuje się, że nie, mianowicie $p_{lost}=1{,}1*10^{-3}$, pozostałe wymagania są spełnione. Rozwiążmy więc parę nierówności: $\begin{cases} p_{lost} < 10^{-4} \\ \lambda_a \geq 2 * \lambda_t \end{cases}$. Jednym z rozwiązań tej pary jest $\lambda_t=16$, $\lambda_a=32$. Przy tych parametrach mamy: $p_{lost}=9{,}4*10^{-5}$, p(**C6**-**C3**)$=6{,}3*10^{-9}$. Wymagania są teraz spełnione, musimy jednak ponad trzykrotnie zwiększyć prędkość transmisji potwierdzenia oraz szesnastokrotnie prędkość transmisji danych.

6. Wydajnościowe Mapy Stanów

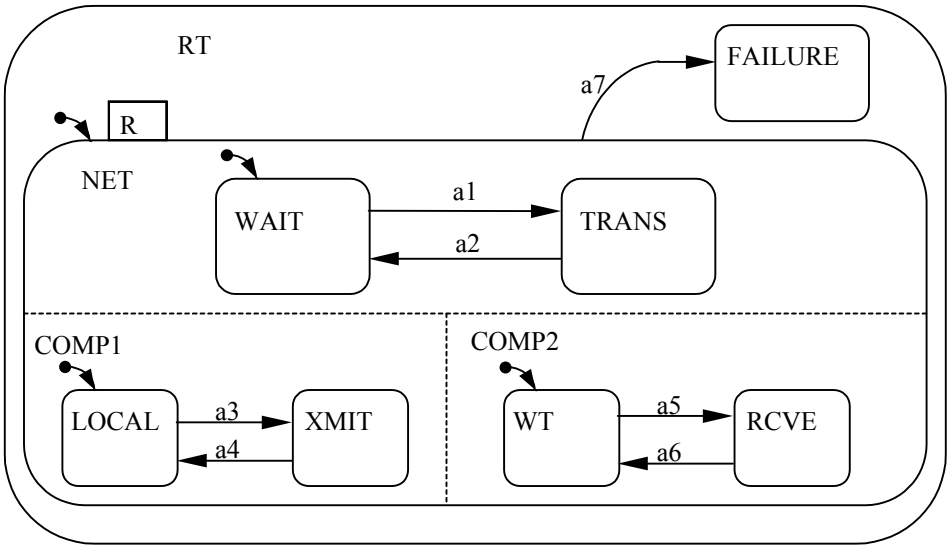
6.1. Wprowadzenie

Rozdział bieżący stanowi próbę uzyskania precyzyjnej definicji Map Stanów wzbogaconych o elementy czasu i losowości. Opisana jest tu wersja nazwana Wydajnościowymi Mapami Stanów. Jest ona rozszerzeniem opisanej w poprzednim rozdziale Markowowskiej Mapy Stanów o dowolne rozkłady zmiennych losowych. Obecny rozdział zawiera propozycję algebry procesowej dla Wydajnościowych Map Stanów. Propozycja ta wzorowana jest na algebrze procesowej dla Markowowskich Map Stanów.

6.2. Przykład wprowadzający

Przypomnijmy sobie przykład z rozdziału o Markowowskich Mapach Stanów. Wprowadźmy do niego pewne modyfikacje polegające na innym określeniu parametrów czasowych. Na rys. 6.1 pokazany jest ten sam co poprzednio system transmisji danych. Różni się on innym określeniem etykiet łuków. Etykiety złożone są z pięciu elementów. Pierwszy z nich, F, jest dystrybuantą zmiennej losowej *opóźnienia otwarcia* łuku lub, w przypadku znanych rozkładów, nazwą danego rozkładu. Gdy $F=\partial_0$ (rozkład jednopunktowy z parametrem 0), wtedy opóźnienie jest równe zero. Dany łuk może być użyty do przejścia między sąsiadującymi komórkami dopiero po upływie czasu opóźnienia otwarcia. Drugi element, *p*, opisuje, tak jak poprzednio, *priorytet* łuku. Trzecim parametrem jest zbiór *E*, zawierający *zdarzenia wyzwalające* dany łuk. Następny, czwarty element opisuje zbiór *zdarzeń natychmiastowych* generowanych podczas wykonywania danego łuku. Ostatni wreszcie, piąty element etykiety jest skończoną listą *zdarzeń odroczonych*. Jako przykład zostanie opisana etykieta łuku *a2*. Powinna być ona odczytywana następująco: „Łuk o nazwie *a2* jest łukiem z opóźnieniem generowanym zgodnie z rozkładem jednostajnym na przedziale (1,3), o priorytecie równym 1. Łuk nie jest wyzwalany żadnym zdarzeniem.

W czasie wykonywania tego łuku zostaje wygenerowane zdarzenie "rrq" oraz zaplanowane odroczone zdarzenie "sack". Samo zdarzenie "sack" wystąpi po upłynięciu czasu będącego realizacją zmiennej losowej o rozkładzie wykładniczym (F≈Exp) z parametrem λ = 9.3."



a1:<∂_0,1,{srq},∅,<>> a2:<$\mathbf{U_{1,3}}$,1, ∅,{rrq},<sack,Exp(9.3)>>

a3:<∂_0,1,{send},{srq},<>> a4:<∂_0,1,{sack},∅,<>>

a5:<∂_0,1,{rrq},∅,<>> a6:<$\mathbf{U_{0,1}}$,1,∅,∅,<>>

a7:<∂_0,1,{fail},∅,<>>

Rys. 6.1 Przykład Wydajnościowej Mapy Stanów

6.3. Mapy Stanów

Wydajnościowe Mapy Stanów nie różnią się w części strukturalnej od Markowowskich Map Stanów opisanych w poprzednim rozdziale. Odpowiednie definicje tam zawarte obowiązują w tej wersji Map Stanów. Poniższa definicja różni się inną postacią funkcji etykietującej.

Definicja 6.1

Wydajnościowa Mapa Stanów **WMS** jest parą:

WMS = <*S, L* >

gdzie:

— S jest Mapą Stanów z definicji 5.1,

— L jest *funkcją etykietującą*, która daje interpretację łuków.

Każda etykieta jest ciągiem:

$$l = <F, p, E, \{e'_1, \ldots, e'_m\}, <<e_1, F_1>, \ldots, <e_n, F_n>>>$$

gdzie:

— F to dystrybuanta zmiennej losowej opisującej *opóźnienie otwarcia* łuku. Dla dobrze znanych rozkładów prawdopodobieństwa można tu podać nazwę rozkładu wraz z parametrami. Gdy F jest dystrybuantą rozkładu jednopunktowego o wartości 0, co oznaczamy jako ∂_0, wtedy nie ma żadnego opóźnienia, w przeciwnym razie, opóźnienie $\tau>0$ jest realizacją zmiennej losowej o dystrybuancie F, co zapisujemy $\tau \in \mathrm{RE}(F)$. Skojarzona zmienna losowa będzie oznaczana *rand*(F).

— $p \in$ Nat jest *priorytetem* łuku.

— $E \subseteq EVENTS$ jest zbiorem *zdarzeń wyzwalających* dany łuk.

— $\{e'_1, \ldots, e'_m\}$, gdzie $e'_i \in EVENTS$ (m ≥ 0), jest skończonym zbiorem *natychmiastowych zdarzeń wewnętrznych*, które są generowane podczas wykonywania danego łuku.

— $<<e_1,F_1>, \ldots, <e_n,F_n>>$ gdzie $e_i \in EVENTS$ dla i = 1, ..., n (n≥0), jest skończoną listą *odroczonych zdarzeń wewnętrznych*, które są inicjowane podczas wykonywania danego łuku. F_i jest dystrybuantą zmiennej losowej wyrażającej opóźnienie generacji zdarzenia e_i po wykonaniu łuku.

Pozostaje w mocy nakaz, aby łuki opuszczające daną komórkę miały różny priorytet.

Pojęcie konfiguracji strukturalnej Wydajnościowych Map Stanów jest identyczne z wyżej definiowanym dla Markowowskich Map Stanów. Taka sama jest także definicja składni algebry procesowej więc nie będzie tu powtarzana. Inna jest natomiast definicja konfiguracji czasowej. Ponieważ mamy teraz do czynienia z dowolnymi rozkładami zmiennych losowych

opisujących czasy otwarcia łuków, więc musimy zapamiętywać w konfiguracji ich realizacje.

Definicja 6.2

Konfiguracja czasowa T Wydajnościowej Mapy Stanów w chwili τ dla konfiguracji strukturalnej *P* jest zdefiniowana jako czwórka:

$T(P) = <AFO, OA, IE, DE>$

gdzie:

— $AFO \subseteq ArcN \times R_+$ jest zbiorem par składających się z *nazw zamkniętych łuków*, które są relewantne do strukturalnej konfiguracji *P*, oraz czasów ich otwarcia będących realizacjami odpowiednich zmiennych losowych,

— $OA \subseteq ArcN$ jest zbiorem *nazw otwartych łuków*, które są relewantne do strukturalnej konfiguracji *P*,

— $IE \subseteq EVENTS$ jest zbiorem *dostępnych zdarzeń wewnętrznych*,

— $DE = <<e_1,\tau_1>,\ldots, <e_n,\tau_n>>$ jest listą *odroczonych zdarzeń wewnętrznych*; τ_i jest czasem wystąpienia zdarzenia e_i (i = 1,..., n, i n ≥ 0).

Startowa konfiguracja czasowa dla startowej konfiguracji strukturalnej P_{init} w chwili τ=0 jest zdefiniowana następująco:

$T(P_{init}) = < AFO_{init}, OA_{init}, IE_{init}, DE_{init} >$

gdzie:

— $AFO_{init} = \{<a, \tau> \mid a \in relevant(P_{init}), L(a).delay \neq \partial_0, \tau = RE(L(a).delay) \}$

— $OA_{init} = \{a \mid a \in relevant(P_{init}), L(a).delay = \partial_0\}$

— $IE_{init} = \varnothing$

— $DE_{init} = < >$, gdzie $< >$ oznacza pustą listę.

Konfiguracją Wydajnościowej Mapy Stanów nazywamy parę składającą się z konfiguracji strukturalnej i czasowej.

Przykład (cd.)

Konfiguracją Wydajnościowej Mapy Stanów S z rysunku 6.1 w stanie startowym i chwili $\tau = 0$ jest:

$<\phi(S), <\varnothing,\{a1,a3,a5\},\varnothing,<>>>$

Mapa Stanów zanurzona jest w *środowisku* określonym tak jak poprzednio, z tym że nie zakładamy wykładniczych rozkładów zmiennych losowych opisujących czasy pojawiania się zdarzeń zewnętrznych.

6.4. Przejścia między konfiguracjami

Niech $<P, T(P)>$, gdzie $T(P) = <$AFO, OA, *IE*, *DE* $>$, będzie konfiguracją Wydajnościowej Mapy Stanów w chwili τ. Załóżmy, że żadne przejście strukturalne nie jest możliwe w chwili τ. W tej sytuacji może zajść przejście czasowe albo może pojawić się zdarzenie zewnętrzne.

Są dwie przyczyny przejść czasowych:

1. pojawienie się zdarzenia wewnętrznego jako wynik upływu czasu opóźnienia dla któregoś z listy zdarzeń odroczonych,
2. upłynięcie czasu opóźnienia pewnego łuku o niezerowym czasie co czyni ten łuk otwartym.

Niech $<e_i,\tau_i>$ (dla pewnego i= 1, ..., n) będzie elementem z listy zdarzeń odroczonych:

$DE = <<e_1,\tau_1>, \ldots, <e_n\tau_n>>$,

takim, że $\tau_i = \tau$.

Pojawienie się wewnętrznego zdarzenia odroczonego $e_i \in EVENTS$ wywołuje następujące mikroprzejście czasowe zapisywane:

$T(P) = \mu internal(e_i) \Rightarrow T_1(P)$

gdzie

$T_1(P) = <AFO_1, OA_1, IE_1, DE_1>$,

oraz

$AFO_1 = AFO$, $OA_1 = OA$,

$IE_1 = IE \cup \{e_i\}$,

$DE_1 = <<e_1,\tau_1>, \ldots, <e_{i-1},\tau_{i-1}>,<e_{i+1},\tau_{i+1}>, \ldots, <e_n,\tau_n>>$.

Dla każdej pary $<e_j,\tau_j> \in DE$, spełniającej powyższy warunek wykonujemy odpowiednie mikroprzejście czasowe. Ich sekwencja tworzy *przejście czasowe* zapisywane:

$T(P) = internal(E_\tau) \Rightarrow T(P)'$

Tak więc po tym przejściu czasowym $EO = \{e_i \mid <e_i,\tau_i> \in DE, \tau_i = \tau\}$.

Upłynięcie czasu opóźnienia otwierające pewien łuk o nazwie $a \in ArcN$ i czasie otwarcia równym τ jest przyczyną mikroprzejścia czasowego zapisywanego następująco:

$T(P) = \mu open(a) \Rightarrow T_1(P)'$

gdzie

$T_1(P)' = <AFO_1', OA_1', IE_1', DE_1'>$,

natomiast

$AFO_1' = AFO \setminus \{<a,\tau>\}$,

$OA_1' = OA \cup \{a\}$,

$IE_1' = IE$, $DE_1' = DE$.

Sekwencja mikroprzejść dla wszystkich łuków o czasie otwarcia równym τ tworzy przejście czasowe:

$T(P) = open(A_\tau) \Rightarrow T(P)'$

Uwaga:

Przejście *internal* modyfikuje jedynie składowe *IE* i *DE*, natomiast przejście *open* — *AFO* i *OA*, konfiguracji czasowej. Jeśli więc zachodzą warunki wykonania obu typów przejść, ich kolejność jest nieistotna.

Wystąpienie zdarzenia zewnętrznego nie zmienia konfiguracji, a tylko ofertę zdarzeń. Generalnie, w danej chwili mogą wystąpić zdarzenia zewnętrzne

(teraz nie zakładamy, że w danym momencie może być tylko jedno) i zdarzenia wewnętrzne, tak więc $EO = IE \cup EE$.

Opóźnienie θ wystąpienia pierwszego zdarzenia jest wartością minimalną realizacji następujących zmiennych losowych:

1. zmiennych reprezentujących czas wystąpienia zdarzenia spośród zdarzeń znajdujących się na liście $DE = <<e_1, \tau_1>, \ldots, <e_n, \tau_n>>$ zdarzeń odroczonych,
2. zmiennych reprezentujących czas kiedy jeden z jeszcze nieotwartych łuków relewantnych do konfiguracji strukturalnej *P*, czyli łuków ze zbioru *AFO*, zostanie otwarty,
3. zmiennej reprezentującej czas pojawienia się zdarzeń zewnętrznych.

W tym modelu Map Stanów także przyjęte są następujące założenia.

Założenia 6.1

1. Czas trwania zdarzenia jest równy zero,
2. Zdarzenie oferowane w danej chwili jest natychmiast dostępne dla wszystkich łuków,
3. Czas trwania kroku, jak również dowolnego przejścia czasowego jest zerowy.

Algorytm zdefiniowany poniżej opisuje generację sekwencji mikrokroków tworzącej krok.

Algorytm 6.1

Algorytm ten jest niemal identyczny jak 5.1. Różni się w następujących punktach.

Punkt czwarty zmienia się na:

4. Modyfikuj zbiór otwartych łuków:

$$OA' = OA \setminus remove(P, OA, EO) \cup \{a \in relevant(P') \mid L(a).delay = \partial_0\}$$

Po tym punkcie należy dodać:

4a. Modyfikuj zbiór łuków przygotowanych do otwarcia:

$AFO_{tmp} = AFO \setminus remove(P,OA,EO)$,

$AFO' = AFO_{tmp} \cup$

$\cup \{<a, \tau> \mid a \in (relevant(P) \setminus AFO_{tmp}), L(a).delay \neq \partial_0,$

$\tau = \tau + \mathrm{RE}(L(a).delay)\}$,

AFO_{tmp} jest tymczasowym zbiorem par <łuk, czas> zdefiniowanym dla potrzeb tego kroku algorytmu.

Wreszcie zmodyfikować musimy punkt siódmy.

7. Zdefiniujmy nowy zbiór odroczonych zdarzeń wewnętrznych:

$$DE' = DE \,\hat{}\, \mathrm{RE}(L(a_1).deferred) \,\hat{}\, \ldots \,\hat{}\, \mathrm{RE}(L(a_n).deferred)$$

gdzie

^ oznacza konkatenację list,

przeciążona funkcja $\mathrm{RE}(<<e_1,F_1>,\ldots,<e_n,F_n>>)=<<e_1,\tau_1>,\ldots,<e_n,\tau_n>>$,

gdzie $\tau_i = \tau+\mathrm{RE}(F_i)$ dla i=1,...,n,

$mprior(P, OA, EO) = \{a_1,\ldots, a_n\}$.

Warunek zakończenia kroku jest taki sam jak dla Markowowskich Map Stanów.

Jeśli w danej chwili τ występują warunki dla zajścia przejścia czasowego i przejścia strukturalnego, najpierw wykonywane jest przejście czasowe. Kolejność natomiast przejść czasowych polegających na otwieraniu łuków oraz generacji zdarzeń wewnętrznych (z listy zdarzeń odroczonych) jest nieistotna i zawsze prowadzi do tego samego rezultatu.

Twierdzenie 5.1 trzeba nieco osłabić i nadać mu następującą formę.

Twierdzenie 6.1

Dla danej konfiguracji startowej $<P, T(P)>$ i oferty zdarzeń EO, krok

$<P, T(P)> = step(EO) \Rightarrow <P', T(P')>$

wyznacza konfigurację kończącą $<P', T(P')>$ w sposób jednoznaczny z dokładnością do realizacji zmiennych losowych określających czasy otwarcia łuków oraz czasy zajścia zdarzeń odroczonych.

Dowód jest analogiczny do dowodu twierdzenia 5.1.

Przykład (cd.)

Niech Wydajnościowa Mapa Stanów z rys. 6.1, otrzyma w chwili τ, zdarzenie "send". Zastosujmy wyżej opisany algorytm. Najpierw definiujemy zbiór $mprior(P,OA,EO) = \{a3\}$. Ponieważ nie jest on pusty, może zajść przejście strukturalne:

$\phi(S) \text{—}\{a3\}\rightarrow P'$,

gdzie

P' = RT [R [NET [WAIT] || COMP1 [XMIT] || COMP2 [WT]]].

Elementy konfiguracji czasowej

$T(P') = <AFO', OA', IE', DE'>$ będą następujące:

$AFO' = \emptyset$, $OA' = \{a1, a4, a5\}$, $IE' = \{srq\}$, $DE' = < >$.

Ponadto $EE = \emptyset$. Teraz, ponieważ $IE' \neq \emptyset$, wykonany zostanie następny mikrokrok dając:

$P' \text{—}\{a1\}\rightarrow P''$,

gdzie

P'' = RT [R [NET [TRANS] || COMP1 [XMIT] || COMP2 [WT]]],

oraz

$T(P'') = <AFO'', OA'', IE'', DE''> = <\{<a2, RE(\mathbf{U_{1,3}})>\}, \{a4, a5\}, \emptyset, <>>$.

Teraz, ponieważ zbiory EE i IE'' są puste, krok się kończy. Następne przejście będzie czasowe i może być tylko następujące:

— upłynięcie czasu opóźnienia a2 wywołujące przejście czasowe:

$$T(P'') = open(\text{a2}) \Rightarrow T(P''')$$

6.5. System przejść

Etykietowany System Przejść $LTS(WMS)$ dla Wydajnościowych Map Stanów WMS różni się od $LTS(MMS)$ inną postacią relacji przejść. Argumentami przejść czasowych są tu zbiory łuków bądź zdarzeń, a nie pojedyncze elementy.

$$LTS(WMS) = <CONF, =LAB\Rightarrow, C_{init}>$$

gdzie:

$CONF$ jest zbiorem konfiguracji $\{<P, T(P)> \mid P \in PROC\}$

$=LAB\Rightarrow \subseteq CONF \times CONF$, jest relacją przejść, gdzie $=LAB\Rightarrow$ jest zdefiniowane jako suma:

$$\{=internal(E_\tau)\Rightarrow \mid E_\tau \subseteq EVENTS\} \cup \{=open(A_\tau)\Rightarrow \mid A_\tau \subseteq ArcN\} \cup$$

$$\cup \{=micro\text{-}step(EO)\Rightarrow \mid EO \subseteq EVENTS\}$$

$C_{init} = <P_{init}, T(P_{init})>$

Zdefiniowanie Etykietowanego Systemu Przejść daje możliwość określenia procesu stochastycznego opisującego zachowanie wyjściowej Mapy Stanów. Proces ten można dalej badać metodami analitycznymi bądź symulacyjnymi. Podejście analityczne pokazane było w poprzednim rozdziale, teraz będzie zaprezentowane podejście symulacyjne.

6.6. Badania symulacyjne

6.6.1. Wstęp

Badania symulacyjne przeprowadzać można w bardziej ogólnych przypadkach niż wyliczenia analityczne. Opisywany tu komputerowy symulator ma być pomocą przy badaniu zachowania się systemów specyfikowanych za pomocą Map Stanów. Ze względu na aktualny kierunek badań prowa-

dzonych przez autora zdecydowano się zastosować w symulatorze model Wydajnościowych Map Stanów. Model ten umożliwia zadawanie opóźnień otwarcia łuków oraz generacji zdarzeń wewnętrznych za pomocą zmiennych losowych o dowolnych rozkładach.

6.6.2. Symulator

Symulator umożliwia wprowadzenie opisu analizowanego systemu w formie tekstowej o składni opisanej w rozdziale D1.2. Po wprowadzeniu opisu systemu możliwa jest symulacja jego działania poprzez wykonywanie *kroków* zgodnie z algorytmem opisanym w rozdziale 5.2.6 wraz z modyfikacjami wprowadzonymi w 6.4. Zdarzenia zewnętrzne są generowane automatycznie zgodnie ze specyfikacją podaną przez użytkownika. W każdym momencie pracy możliwe jest prześledzenie stanu systemu określonego jego *konfiguracją strukturalną* oraz *czasową*. Ponadto program umożliwia wyprowadzenie opisu stanu systemu w formacie *Algebry Procesowej* szczegółowo omówionej w punkcie 5.2.4. Dokładniejszy opis działania symulatora mieści się w Dodatku (rozdz. D1.1).

6.6.3. Przykład symulacji

Weźmy znów pod uwagę przykład rozważany w poprzednim i obecnym rozdziale. Aby możliwe było porównanie wyników symulacyjnych z analitycznymi wprowadźmy takie same parametry systemu. Tak samo jak przy poprzednich wyliczeniach usuńmy komórkę FAILURE wraz z łukiem a7.
Po przeprowadzeniu pewnej liczby kroków symulacji uzyskano następującą listę konfiguracji, przez które system przechodził.

1. RT[R[NET[WAIT]||COMP1[LOCAL]||COMP2[WT]]]<<>,{a1,a3,a5},{},<>>
2. RT[R[NET[WAIT]||COMP1[XMIT]||COMP2[WT]]]<<>,{a1,a4,a5},{},<sack>>
3. RT[R[NET[WAIT]||COMP1[XMIT]||COMP2[RCVE]]]<<a6>,{a1,a4},{},<sack>>
4. RT[R[NET[TRANS]||COMP1[XMIT]||COMP2[RCVE]]]<<a2,a6>,{a4},{},<>>

5. RT[R[NET[WAIT]||COMP1[LOCAL]||COMP2[RCVE]]]<<a6>,{a1,a3},{},<>>

6. RT[R[NET[TRANS]||COMP1[XMIT]||COMP2[WT]]]<<a2>,{a4,a5},{},<>>

Odpowiadają one konfiguracjom **C1**,**C4**,**C3**,**C6**,**C5** oraz **C2** z przykładu na stronie 85. Oczywiście tamte dotyczyły Markowowskich, a te Wydajnościowych Map Stanów, więc zamieszczone tutaj mają dodatkowy element konfiguracji czasowej — listę zamkniętych łuków AFO .

Znajdźmy teraz ułamki czasu, przez który system przebywa w poszczególnych konfiguracjach przy wartościach parametru λ_s takich jak przyjęto w tabeli 5.3 ze strony 90. Znajdźmy też czas cyklu systemu względem konfiguracji **C3** oznaczając go jak poprzednio γ_3. Wyniki przedstawione są w poniższej tabeli dla dwóch symulacji po 10000 kroków każda.

Tabela 6.1

λ_s	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	γ_3
1	0.421	0.469	0.0336	0.0178	0.0500	0.00860	2.07
	0.437	0.453	0.0334	0.0189	0.0490	0.00772	2.08
0.1	0.890	0.088	0.0064	0.00345	0.0112	0.00028	11.09
	0.886	0.091	0.0063	0.00353	0.0121	0.00017	10.90
0.01	0.988	0.0098	0.00069	0.00038	0.0013	1.2e-6	101.2
	0.988	0.0099	0.00068	0.00039	0.0013	1.7e-6	100.9

Widać dobrą zgodność powyższych danych z wynikami zawartymi w tabeli 5.3. Jedyną większą różnicę obserwujemy dla p(**C6**) przy λ_s=0.01. Wynika ona ze zbyt małej liczby kroków dla poprawnego zmierzenia tej wartości. System przebywał w konfiguracji **C6** tylko 2 lub 3 razy. Dla dłuższych symulacji (np. 250tys. kroków) wartość ta zbliża się do poprzednio wyliczonej. W ten sposób dokonaliśmy wzajemnej weryfikacji obu metod. Porównajmy teraz uzyskane wyniki z rezultatem symulacji badanego systemu przy innych rozkładach zmiennych losowych. Dla łuku *a6* oraz zdarzenia odroczonego "*sack*" wprowadźmy jednostajne rozkłady czasów otwarcia lub pojawienia się tak, żeby nie zmieniły się wartości oczekiwane

tych czasów. Dla zdarzenia "*sack*" będzie to rozkład jednostajny na przedziale (0.1,0.12), dla łuku *a6* — na przedziale (0.17,0.23). Poniższa tabela przedstawia wyniki tych badań.

Tabela 6.2

λ_s	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	γ_3
1	0.437	0.469	0.051	0.0001	0.04	0.0016	2.14
0.1	0.89	0.099	0.0097	0	0.0079	0.00003	11.38
0.01	0.988	0.01	0.0011	0	0.0009	8e-8	100.9

Widać niewielkie różnice w czasie cyklu. Natomiast występują istotniejsze różnice w ułamkach czasu przebywania w poszczególnych konfiguracjach. Tłumaczyć można to ograniczonością stosowanych rozkładów co powoduje, że niektóre kombinacje realizacji zmiennych losowych nie występują i system nie osiąga pewnych konfiguracji jak np. **C4** dla $\lambda_s<1$.

6.7. Podsumowanie

Przedstawione Wydajnościowe Mapy Stanów są połączeniem formalizmu Map Stanów z ideą Stochastycznej Algebry Procesowej. Dla omawianego rozszerzenia zdefiniowana została składnia i semantyka operacyjna w formie algebraicznej. Zastosowanie do tego celu algebry procesowej pozwoli na dalsze badania modelu z wykorzystaniem osiągnięć teorii Stochastycznych Algebr Procesowych.

Dzięki dokładnemu zdefiniowaniu poszczególnych rodzajów przejść możliwe było utworzenie symulatora Wydajnościowych Map Stanów. Dla zastosowanych tu ogólnych rozkładów prawdopodobieństw zmiennych losowych nie zawsze jest możliwe znalezienie matematycznego modelu badanego systemu i udzielenie odpowiedzi na pytanie o jego wydajność. Wtedy pomocne są badania symulacyjne. Przykład takiego podejścia do zagadnienia przedstawiony został w rozdziale 6.6.

7. Porównanie przedstawionych propozycji

W rozdziałach poprzednich przedstawione są propozycje wprowadzenia do Map Stanów rozszerzeń pozwalających na ocenę wydajności opisywanych nimi systemów. Pierwsza z nich zakłada wykorzystanie teorii sieci Petriego, a zwłaszcza ich wersji stochastycznych. Zastosowanie jej jednak wymaga dokonania żmudnej konwersji Mapy Stanów w Sieć Petriego. Prawdopodobnie możliwa jest automatyzacja tego kroku. W jego wyniku otrzymuje się Kolorowaną Hierarchiczną Sieć Petriego (*HCPN*), którą można dalej badać metodami analitycznymi bądź symulacyjnymi. Sieć ta jednak ma bardziej skomplikowaną postać niż wyjściowa Mapa Stanów, a ponadto częściowo gubi się w niej pierwotna struktura hierarchii. Wynika to z odmiennego sposobu określania zależności hierarchicznych w Kolorowanych Hierarchicznych Sieciach Petriego i w Mapach Stanów. Nie jest mianowicie określona tam semantyka zabierania bądź wkładania znacznika z/do miejsca zastępczego. Istnieje inny model rozszerzonych Sieci Petriego zwany Place Chart Nets opisany w pozycji [39] dopuszczający stosowanie podobnej jak w Mapach Stanów hierarchii. Autorowi nie jest jednak znany dowód równoważności tego modelu z Kolorowanymi Sieciami Petriego ani żadne jego stochastyczne rozszerzenie, które umożliwiałoby badanie wydajności systemów. Propozycja zamiany opisu systemu Mapami Stanów na równoważną Sieć Petriego wydaje się więc atrakcyjna pod warunkiem skonstruowania zestawu narzędzi komputerowych wspomagających jej stosowanie. Jednocześnie ta propozycja wprowadza najmniej ograniczeń do podstawowych Map Stanów. Jako stochastyczne rozszerzenie sieci Petriego zaproponowano użycie Regularnych Stochastycznych Sieci Petriego (*RSPN*). Daje to możliwość modelowania opóźnień wykonania przejść. Opóźnienia te opisywane są zmiennymi losowymi o rozkładach wykładniczych. Możliwe jest też wprowadzenie opóźnień generacji zdarzeń lub czasów życia zdarzeń również opisanych zmiennymi losowymi o rozkładach wykładniczych. Kosz-

tem komplikacji opisu można też zamodelować czas otwarcia przejścia.

Kolejna propozycja wprowadza elementy czasu i losowości bezpośrednio do formalizmu Map Stanów. Podstawowy model rozszerzony jest o losowe opóźnienia wykonania przejścia. Dodane są także czasy życia zdarzeń wewnętrznych oraz czasy przeterminowania przejść. Zdefiniowana jest w sposób formalny składnia i semantyka rozszerzenia. Pewne zadawane w modelu czasy są deterministyczne. Ponadto nie jest rozwiązany konflikt pomiędzy mogącymi zajść jednocześnie przejściami. Oba te powody sprawiają, że trudne jest analityczne badanie wydajności systemów opisywanych tym formalizmem. Przy badaniach symulacyjnych także konieczne jest zdefiniowanie sposobu postępowania na wypadek konfliktu między przejściami.

W trzeciej i czwartej propozycji zrezygnowano z czasów przeterminowania przejść oraz czasów życia zdarzeń. Tu każde zdarzenie ma zerowy czas życia i jeśli nie zostanie wykorzystane czyli nie wywoła przejścia, wtedy znika. Zamiast wspomnianych parametrów wprowadzono losowe opóźnienia generacji zdarzeń wewnętrznych. Opóźnienia te nie są równoważne czasowi życia zdarzeń z poprzedniego modelu. Tam zdarzenie mogło wywołać przejście przez cały czas swojego życia, po czym znikało, natomiast tu zdarzenie może wywołać przejście tylko w momencie pojawienia. Dla uniknięcia niedeterminizmu przy wyborze przejść wprowadzono priorytety łuków. Parametrem modelu jest tu też losowy czas otwarcia przejścia. Czas ten znowu nie jest równoważny czasowi opóźnienia wykonania przejścia z poprzedniego modelu. Określenie czasu wykonania powoduje, że dany relewantny do konfiguracji łuk może być wyzwolony w dowolnym momencie. Wykonanie go nastąpi po upływie czasu opóźnienia. Czas otwarcia łuku powoduje natomiast niemożność wyzwolenia łuku przed upływem tego czasu. Zastosowany w dwu ostatnich propozycjach zestaw rozszerzeń, jak również algebraiczne zdefiniowanie składni i semantyki modelu umożliwia zarówno symulacyjne

jak i analityczne badanie wydajności projektowanych systemów. Rozwiązanie analityczne wymaga sprecyzowania rozkładów zmiennych losowych oraz poczynienia pewnych założeń co do środowiska. Jako model matematyczny wybrano teorię procesów Markowa. W związku z tym określono rozkłady zmiennych losowych jako wykładnicze oraz założono, że zdarzenia zewnętrzne tworzą potok Poissona. Powstały w ten sposób Markowowskie Mapy Stanów. Określono dla nich analityczną metodę oceny wydajności.

Wydajnościowe Mapy Stanów są rozszerzeniem poprzedniego modelu o dowolne rozkłady zmiennych losowych. Utrudnia to analityczne badanie systemów lecz pozwala na lepsze opisanie parametrów czasowych. Rozkład wykładniczy nie zawsze jest dobrym przybliżeniem rzeczywistego rozkładu opisującego dane zjawisko. Dla tego rozszerzenia Map Stanów możliwe są badania symulacyjne.

Zbierzmy teraz właściwości poszczególnych proponowanych modeli porównując je ze sobą wzajemnie oraz z modelem zastosowanym w komercyjnym narzędziu STATEMATE.

Wprowadźmy skrótowe oznaczenia modeli:

P - Mapy Stanów poddawane konwersji do Sieci Petriego,

S - Stochastyczne Mapy Stanów,

M - Markowowskie Mapy Stanów,

W - Wydajnościowe Mapy Stanów,

H - Mapy Stanów stosowane w STATEMATE.

Tabela 7.1 zawiera zestawienie rozszerzeń czasowych stosowanych w powyższych modelach. Wystąpienie danego skrótu w pierwszej kolumnie oznacza, że w odpowiadającym mu modelu wprowadzono odpowiednie rozszerzenie. Druga kolumna zawiera skróty modeli, w których możliwe jest symulowanie danego rozszerzenia. Symulacja ta jest czasem okupiona znaczną komplikacją specyfikacji.

Tabela 7.1

Nazwa rozszerzenia	Model	Symulacja
opóźnienie wykonania przejścia	P, S, H	
opóźnienie otwarcia przejścia	M, W	P, H
przeterminowanie przejścia	S	H
czas życia zdarzeń	P, S	H
opóźnienie generacji zdarzeń	P, M, W, H	

Każde z proponowanych w pracy rozszerzeń można zasymulować w narzędziu STATEMATE jednak wymagane jest do tego użycie zmiennych oraz budowanych z nich warunków, które nie są w pracy rozważane.

Porównajmy jeszcze takie cechy modeli jak metody badania specyfikowanych systemów oraz stosowane rozkłady zmiennych losowych.

Tabela 7.2

Model	Metoda badania	Rozkłady zm. losowych
P	symulacyjna i analityczna	wykładnicze
S	symulacyjna	dowolne
M	analityczna	wykładnicze
W	symulacyjna	dowolne
H	symulacyjna[1]	dowolne (wybór)

[1] Symulator STATEMATE nie umożliwia zebrania danych statystycznych. Możliwe jest tylko prześledzenie poprawności działania specyfikowanego systemu.

8. Podsumowanie

W pracy przedstawione są propozycje wprowadzenia do Map Stanów rozszerzeń umożliwiających badanie zagadnień wydajnościowych dla specyfikowanych systemów. Badania takie możliwe są na każdym etapie pracy nad projektem od wstępnej specyfikacji począwszy. Dzięki temu już we wczesnych fazach projektów możliwa jest korekta założeń powodujących ewentualne niezgodności z wymaganiami na wydajność i wybór odpowiednich alternatywnych rozwiązań projektowych.

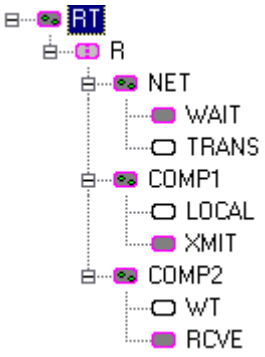
W rozprawie zaproponowano metodę oceny wydajności systemów modelowanych Mapami Stanów opartą na Kolorowanych Sieciach Petriego. Wprowadzono Stochastyczne Mapy Stanów i zdefiniowano formalnie ich składnię oraz semantykę. Dla jednej z propozycji skonstruowano analityczną metodę oceny wydajności bazującą na teorii łańcuchów Markowa. Przedstawiono sposób wyliczania średniego czasu przetwarzania dla pracy acyklicznej systemu oraz średniego czasu cyklu dla systemu reaktywnego. Dla ogólniejszego modelu Wydajnościowych Map Stanów stworzono symulator umożliwiający badania symulacyjne systemów specyfikowanych za pomocą tego formalizmu. Udowodniono w ten sposób podstawową tezę pracy stanowiącą, że jest możliwe wprowadzenie do podstawowego modelu Map Stanów Harela takich rozszerzeń aby dało się badać aspekty wydajnościowe projektowanych z ich użyciem systemów. W pracy zaprezentowano wyniki analitycznego badania prostego systemu komunikacyjnego opisanego Markowowską Mapą Stanów, symulator Wydajnościowych Map Stanów oraz rezultaty otrzymane z jego zastosowania do badania wspomnianego systemu komunikacyjnego.

Dodatek 1. Program *SimChart*

D1.1. Działanie

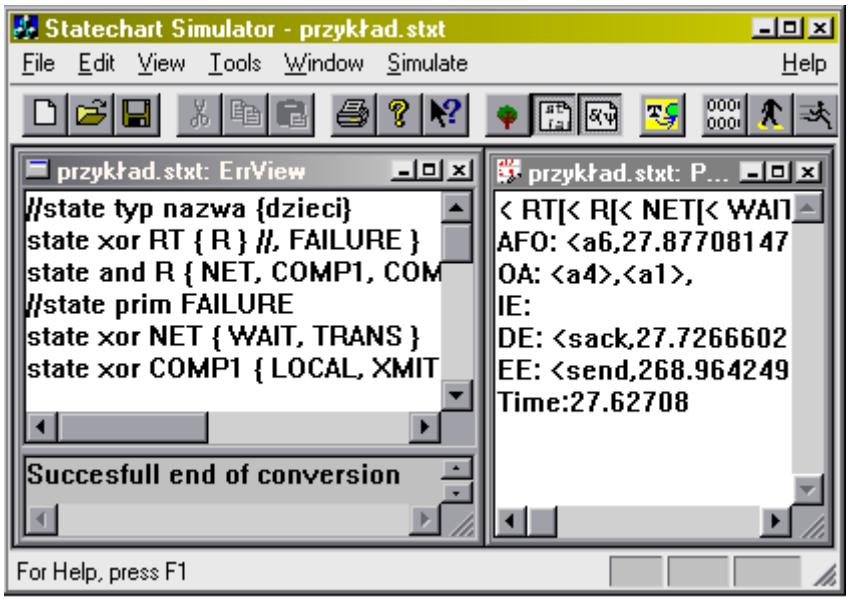
Opisywany program jest 32-bitową aplikacją działającą w systemie Windows95 symulującą działanie systemów opisywanych Wydajnościowymi Mapami Stanów. Badany system zapisuje się w formacie tekstowym opisanym w następnym punkcie. Obecna wersja programu nie umożliwia wprowadzania Map Stanów w postaci graficznej. W trakcie działania symulatora możliwe jest wprowadzanie zmian i poprawek do specyfikacji badanego systemu. Aby jednak zostały one uwzględnione należy specyfikację skompilować. Czynność ta jest jednym z poleceń programu (**Tools/Convert**). Kompilacja przerywa rozpoczętą symulację, zeruje symulowany czas oraz ustawia badany system w konfiguracji startowej. Symulacja polega na wykonywaniu pojedynczych kroków (poleceniem **Simulate/Step**) w celu stwierdzenia poprawnego zachowania się systemu bądź znalezienia błędów specyfikacji. Możliwe jest także wykonywanie serii kroków (**Simulate/Run**), podczas których zbierane są informacje statystyczne o symulowanym systemie. W obecnej wersji symulatora dopuszczalne są tylko cztery rozkłady zmiennych losowych. Są to rozkłady jednopunktowy, jednostajny na odcinku, trójkątny oraz wykładniczy.

W czasie symulacji użytkownik ma dostęp do informacji o bieżącej konfiguracji systemu. Jest ona przedstawiona w dwóch formach graficznej (tylko konfiguracja strukturalna) i tekstowej. Pierwsza forma pokazuje drzewo komórek z zaznaczeniem kolorem, które z nich należą do bieżącej konfiguracji. Użytkownik może na tym widoku zwijać lub rozwijać wybrane gałęzie. Domyślnie rozwijane są tylko poddrzewa komórek należących do bieżącej konfiguracji. Przykład prezentacji znajduje się na rysunku D.8.1.



Rys. D.8.1 Drzewo hierarchii stanów

Druga, tekstowa forma prezentacji przedstawia konfigurację strukturalną w postaci wyrażenia procesowego należącego do wyżej zdefiniowanej algebry StPA. Na tym widoku konfiguracji pokazana jest też składowa czasowa czyli łuki przygotowane do otwarcia, otwarte łuki, zdarzenia wewnętrzne natychmiastowe i odroczone. Dodatkowo widok ten pokazuje zdarzenia zewnętrzne wraz z czasami ich pojawienia oraz symulowany czas. Przykład tej formy prezentacji znajduje się w prawym oknie na rys. D.8.2.



Rys. D.8.2 Przykład sesji symulacji

Dostępne są także informacje statystyczne o systemie poddanym symulacji. Po wykonaniu odpowiednio dużej liczby kroków wyświetlane są informacje

o wszystkich konfiguracjach, przez które system przechodził. Dla każdej z nich podane mamy następujące informacje:

- łączny czas przebywania systemu w danej konfiguracji,
- ułamek czasu spędzony w konfiguracji,
- średni czas pobytu w konfiguracji,
- średni czas powrotu do danej konfiguracji,
- średni czas cyklu systemu względem danej konfiguracji,
- łączną ilość wejść do konfiguracji.

Poniżej pokazany jest fragment wyników pewnej symulacji.

```
RT[R[NET[WAIT]||COMP1[LOCAL]||COMP2[WT]]]<<>,{a1,a3,a5},{},<>>
Total 'in' time:268.33300     Part of time 'in':0.99524     Average 'in' time:134.16650
Average return time:0.63124                                 Average cycle time:27.87708
Number of entries:2.00000
RT[R[NET[WAIT]||COMP1[XMIT]||COMP2[RCVE]]]<<a6>,{a1,a4},{},<sack>>
Total 'in' time:0.09958       Part of time 'in':0.00037     Average 'in' time:0.09958
Average return time:241.89095                               Average cycle time:241.99053
Number of entries:2.00000
```

D1.2. Język opisu Wydajnościowych Map Stanów

Poniżej przedstawiony jest, w notacji EBNF, język opisu Map Stanów użyty w obecnej wersji symulatora. Symbole pisane z małej litery są symbolami nieterminalnymi gramatyki, podczas gdy symbole pisane z dużej litery są leksemami. Wyrażenia w nawiasach klamrowych są opcjonalne, gwiazdka oznacza element mogący się powtarzać zero lub więcej razy, a znak plusa — element mogący wystąpić więcej niż jeden raz. Po dwóch ukośnych kreskach występuje komentarz zarówno w omawianym języku, jak i w opisującym go metajęzyku. Ponadto, co nie jest tu uwidocznione, ignorowane są znaki spacji, tabulacji oraz nowej linii co pozwala na czytelniejszy dla człowieka zapis specyfikacji. Na początek zdefiniujmy symbole terminalne za pomocą wyrażeń regularnych.

Stan	"[Ss]tate"
Kraw	"[Aa]rc"
ExEvent	"[Ee]xternal"
AND	"[Aa][Nn][Dd]"
XOR	"[Xx][Oo][Rr]"
PRIM	"[Pp][Rr][Ii][Mm]"
ChildsBeg	"{"
ChildsEnd	"}"
LabelBeg	"<"
LabelEnd	">"
ListBeg	"<"
ListEnd	">"
ParmBeg	"("
ParmEnd	")"
Colon	":"
Comma	","
Name	"[a-zA-Z_][a-zA-Z_0-9]*"
Number	"[.0-9]+{[dDfFlLeE]{[\+\-]}[0-9]+}"
WS	"[\ \t]" <<skip();>>
NL	"\r|\n|\r\n" <<skip();>>
Comment	"//~[\r\n]*" <<skip();>>

Występujące za definiującym wyrażeniem polecenie <<skip();>> oznacza, że dany leksem ma być rozpoznany w strumieniu wejściowym i pominięty.

Poniżej przedstawione są produkcje gramatyczne.

```
file :    (chartEl )* Eof ;

chartEl : Stan                  // początek opisu stanu
            (AND|XOR|PRIM)      // typ stanu
            Name                // unikalna nazwa
            {childs}            // lista nazw stanów potomnych
          |
```

```
            Kraw              // początek opisu łuku
            Name              // nazwa stanu startowego,
            Comma
            Name              // nazwa stanu końcowego
            Comma
            {aname}           // opcjonalna nazwa łuku
            arclabel          // etykieta
          |
            External          // opis zdarzenia zewnętrznego
            Name              // nazwa zdarzenia
            Comma
            rnd_distr         // nazwa rozkładu zm. losowej
          ;

childs :  ChildsBeg Name (Comma Name)* ChildsEnd;

aname :   Name Colon;         // nazwa łuku,

arclabel: LabelBeg
          rnd_distr           // rozkład zm. losowej (otwarcia łuku),
          Comma
          Number              // priorytet,
          Comma
          ev_list             // lista zdarzeń wyzwalających,
          Comma
          ev_list             // lista akcji natychmiastowych,
          Comma
          def_list            // lista zdarzeń odroczonych
          LabelEnd;
```

```
ev_list : ListBeg
          { nm:Name
            ( Comma
            nm1:Name)*
          }                    // lista nazw zdarzeń
          ListEnd;

def_list: ListBeg
          { def_ev
            ( Comma def_ev
            )*
          }                    // lista par <ev,F>
          ListEnd;

def_ev :  ListBeg
          Name                 // nazwa zdarzenia
          Comma
          rnd_distr            // nazwa rozkładu zm. losowej
          ListEnd;

rnd_distr : Number  //rozkład jednopunktowy zapis uproszczony
          |
            Name               //nazwa rozkładu
            ParmBeg
            { Number
              ( Comma Number)*
            }                  // lista parametrów
            ParmEnd;
```

Przykładowa Mapa Stanów z rys. 6.1 na str. 95 może zostać zapisana w powyższym języku następująco.

```
//state typ nazwa {dzieci}
state xor RT { R, FAILURE }
state and R { NET, COMP1, COMP2 }
state prim FAILURE
```

```
state xor NET { WAIT, TRANS }
state xor COMP1 { LOCAL, XMIT }
state xor COMP2 { WT, RCVE }
//arc ze_stanu, do_stanu, nazwa:<otw, prior, <trig>, <akcje>, <odrocz>>
arc WAIT, TRANS,     a1:< 0, 1, <srq>, <>, <> >
arc TRANS, WAIT,     a2:< range(1,3), 1, <>, <rrq>,<<sack,exp(9.3)>> >
arc LOCAL, XMIT,     a3:< 0, 1, <send>, <srq>, <> >
arc XMIT, LOCAL,     a4:< 0, 1, <sack>, <>, <> >
arc WT, RCVE,        a5:< 0, 1, <rrq>, <>, <> >
arc RCVE, WT,        a6:< range(0,1), 1, <>, <>, <> >
arc R, FAILURE,      a7:< 0, 1, <fail>, <>, <> >
//zdarzenia zewnętrzne: nazwa, rnd
external send exp(0.1)
external fail exp(1e-6)
```

Literatura

[1] Ajmone Marsan M., Balbo G., Conte G., A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems, ACM Transactions Computer Systems, Vol. 2, May 1984, 93-122.

[2] Ajmone Marsan M., Balbo G., Conte G., Performance Models of Multiprocessor Systems, MIT Press, 1989.

[3] Armstrong J., Industrial Integration of Graphical and Formal Specifications, Journal of Systems Software, 40, 211-225, Elsevier Science Inc., 1998.

[4] Atamna Y., Definition of the model "Stochastic Timed Well Formed Coloured Nets", In: Proc. of 5th International Workshop on Petri Nets and Performance Models, IEEE Computer Society Press, 1993, 24-33.

[5] Babczyński T., Huzar Z., Magott J., Algebraic semantics for Markovian Statecharts, In Bradley J.T., Davis N.J. (eds) Proceedings of the XV UKPEW, 105-120, Bristol 1999.

[6] Babczyński T., Huzar Z., Magott J., Derivation of Markovian processes from the UML Statecharts, Archiwum Informatyki Teoretycznej i Stosowanej (przyjęta do druku),

[7] Babczyński T., Konwersja Diagramów Harela w Sieci Petriego, In Proc. of the 4th Conf. on Real-Time Systems 1997, Oficyna Wydawnicza Politechniki Wroclawskiej, 217-226, Szklarska Poręba Sept. 1997.

[8] Babczyński T., Semantyka Stochastycznych Map Stanów, In Proc. of the 5th Conf. on Real-Time Systems 1998, Oficyna Wydawnicza Politechniki Wroclawskiej, 7-16, Szklarska Poręba Sept. 1998.

[9] Bause F., Bucholtz P., Protocol analysis using a timed version of SDL, In Quemada J., Manas J., Vasquez E., eds., Proc. of the 3rd International Conference Formal Description Techniques, 269-285, Madrid 1990.

[10] Bernardo M., Gorrieri R., A Tutorial of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time, Theoretical Computer Science, 201, 1-54, 1998.

[11] Booch G., Object-Oriented Analysis and Design with Applications, Redwood City, CA, Benjamin/Cummings, 1994.

[12] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide, Addison-Wesley, 1998.

[13] Brinksma E., Katoen J.-P., Kangerak R., Latella D., A Stochastic Causality-Based Process Algebra, The Computer Journal, 38, 552-565, 1995.

[14] Davis, A. M., "A Comparison of Techniques for the Specification of External System Behavior", Communications of ACM - 31:9 - 1098-1115 - 1988

[15] Denning P. J., Buzen J. P., The Operational Analysis of Queueing Network Models, ACM Computing Surveys, 10/3, 225-261, 1978.

[16] Douglass B. P., Doing Hard Real. Developing Real-Time Systems with UML, Addison-Wesley, 1999.

[17] Douglass B. P., Real-Time UML, Addison-Wesley, 1998.

[18] Dutheillet C., Haddad S., Regular Stochastic Petri Nets, In: G. Rozenberg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science vol. 483, Springer-Verlag 1991, 186-210. Also in K. Jensen and G. Rozenberg (eds): High-level Petri Nets. Theory and Application, 470-493.

[19] Fricke O., Die Strukturierte Analyse auf der Basis höherer Petrinetze, Diplomarbeit, Universität Hamburg, Fachbereit Informatik, Hamburg 1995.

[20] Fugetta, A., "A Classification of CASE Technology", Computer - 12 (1993) - 25-38 - IEEE, 1993

[21] Górski J. i inni, Inżynieria oprogramowania w projekcie informatycznym, ZNI MIKOM, 1999.

[22] Groote J. F., Vaandrager F., Structured Operational Semantics and Bisimulation as a Congruence, Information and Computation, 100, 202-260, Academic Press 1992.

[23] Harel D., Lachover H., Naamad A., Pnueli A., Politi M., Sherman R., Shtull-Trauring, A., Trakhtenbrot, M., STATEMATE: A Working Environment for the Development of Complex Reactive Systems, IEEE Trans. Soft. Eng. 16 (1990), 403-414.

[24] Harel D., Naamad A., The STATEMATE Semantics of Statecharts, ACM TOSEM – 1996.

[25] Harel D., Pnueli A., Schmidt J., Sherman R., On The Formal Semantics of Statecharts, 2nd IEEE Symposium on Logic in Computer Science, 54-64, Ithaca, 1987.

[26] Harel D., Politi M., Modelling Reactive Systems with Statecharts: The STATEMATE Approach , Pat No. D-1100-43, i-Logix Inc., June 1996.

[27] Harel, D., On Visual Formalisms, Communications of ACM - 31:5 - 514-530 - 1988.

[28] Harel, D., Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programing, 8, 231-274, 1987.

[29] Hermanns H., Rettelbach M., Weiss T., Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching , The Computer Journal, 38/7, 530-541, 1995.

[30] Hillston J., Compositional approach to performance modelling, Ph.D. Thesis, University of Edinburgh, 1994.

[31] Hoare C. A. R., Communicating Sequential Processes, Communications ACM, 21/8, 666-677, 1978.

[32] Huber P., Jensen K., Shapiro R.M., "Hierarchies in Coloured Petri Nets" In: G. Rozenberg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science Vol. 483, Springer-Verlag 1991, 313-341. Also in K. Jensen and G. Rozenberg (eds.): High-level Petri Nets. Theory and Application, 215-243.

[33] Huzar Z., Magott J., Syntax and semantics of a real-time and performance evaluation extension of LOTOS, Fundamenta Informaticae, 29, 77-96, 1997.

[34] ISO/IEC 8807, Information Processing Systems — Open System Interconnections — LOTOS, a formal description technique based on temporal ordering of observational behaviour, 1989.

[35] ISO-DIS9074, ISO/TC97/SC21/WG1-FDT/S.C.-B, Estelle, a formal description technique based on extended state transition model, 1987.

[36] Jensen K., Coloured Petri Nets: A High-level Language for System Design and Analysis, In: G. Rozenberg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science vol. 483, Springer-Verlag 1991, 342-416. Also in K. Jensen and G. Rozenberg (eds.): High-level Petri Nets. Theory and Application, 44-122.

[37] Juanole G., Atamna Y., Dealing with arbitrary time distributions with the Stochastic Timed Petri Net model - Aplication to queueing systems ,In: Proc. of 4th International Workshop on Petri Nets and Performance Models, IEEE Computer Society Press, 1991, 32-41.

[38] Kesten Y., Pnueli A., Timed and Hybrid Statecharts and their Textual Representation, Proceedings of Real-Time and Fault Tolerant Systems, Lecture Notes in Computer Science 571, 591-619, Springer-Verlag, 1992.

[39] Kishinevsky M., Cortadella J., Kondratyev A., Lavagno L., Taubin A., Yakovlev A., Place Chart Nets and their synthesis, In: Proc. of International Symposium on Advanced Research in Synchronous Circuits and Systems, IEEE Computer Society Press, April 1997.

[40] Léonard L., Leduc G., A Formal Definition of Time in LOTOS, in Quemada J. (ed.), Revised Draft on Enhancements to LOTOS, ISO/IEC JTC1/SC21/WG1 N1349, 1994.

[41] Lilja D. J., Measuring computer performance. A practitioner's guide, Cambridge University Press, 2000.

[42] Maggiolo-Schettini A., Merro M., Priorities in Statecharts, Proceedings of LOMAPS'96, Lecture Notes in Computer Science 1192, 404-429, Springer-Verlag, 1997.

[43] Maggiolo-Schettini A., Peron A., Retiming Techniques for Statecharts, Universita di Pisa - 1996.

[44] Magott J., Performance Evaluation of Systems Defined in Specification Language Estelle, Fundamenta Informaticae, 24/4, 333-357, 1995.

[45] Miguel C., Fernandez A., Vidaller L., LOTOS extended with probabilistic behaviour, Formal Aspects of Computing, **5**, 253-281, 1993.

[46] Milner R., A Calculus of Communicating Systems, Lecture Notes in Computer Science 92, 1980.

[47] Mostowski A. W., Stark M., Elementy Algebry Wyższej, PWN, 1975.

[48] Murata T., Petri Nets: Properties, Analysis and Applications, Proc. of IEEE, 77/4, 541-580, 1989.

[49] Pacut A., Prawdopodobieństwo. Teoria. Modelowanie probabilistyczne w technice, Wydawnictwa Naukowo-Techniczne, Warszawa 1985.

[50] Peron A., Maggiolo-Schettini A. , Transitions as Interrupts: A New Semantics for Timed Statecharts, Lecture Notes in Computer Science 789, 806-821, Springer-Verlag, Berlin, 1994.

[51] Peron A., Statecharts - Transition Structures and Transformations, TAPSOFT: 6th International Joint Conference on Theory and Practice of Software Development, Lecture Notes in Computer Science 915, 1995

[52] Peterson J. L., Petri Net Theory and the Modelling of Systems, Prentice-Hall, 1981.

[53] Petersohn C., Urbina L., A Timed Semantics for the STATEMATE Implementation of Statecharts, Proceedings of the Fourth International FME Symposium FME'97, TU Gratz, Austria, 1997.

[54] Pnueli A., Shalev M., What is in a step: On the Semantics of Statecharts, In: T. Ito and A.R. Meyer (Eds), Theoretical Aspects of Computer Software, Lecture Notes in Computer Science 526, 244-264, Springer-Verlag, 1991.

[55] Reisig W., Petri Nets — An Introduction, Springer-Verlag, 1985.

[56] Rockstrom A., Saracco R., SDL—CCITT Specification and Description Language, IEEE Trans. of Communication, 30/6, 1310-1318, 1982.

[57] Rozanow J. A., Wstęp do teorii procesów stochastycznych, PWN, 1974.

[58] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., Object-Oriented Modelling and Design, Englewood Cliffs, NJ, Prentice Hall, 1991.

[59] Sacha K., Bezpieczeństwo oprogramowania w normie IEC 1509, In: Preceedings on the IV Conference on Real-Time Systems 1997, Szklarska Poręba Sept. 97, Oficyna Wydawnicza Politechniki Wrocławskiej, 175-182, 1997.

[60] Sacha K., Projektowanie oprogramowania systemów wbudowanych, Prace Naukowe Politechniki Warszawskiej, z. 115, Oficyna Wydawnicza PW, 1996.

[61] Sacha K., Specyfikacja i Synteza Oprogramowania w Metodzie Transnet In: Preceedings on the III Conference on Real-Time Systems 1996, Szklarska Poręba Sept. 96, Oficyna Wydawnicza Politechniki Wrocławskiej, 49-58, 1996.

[62] Smith C.U., Performance Engineering of Software Systems, Addison-Wesley, 1990.

[63] Smith C.U., Williams L.G., Performance Engineering Evaluation of Object-Oriented Systems with SPE•ED™, Performance Engineering Services and Software Engineering Research, 1997.

[64] Stewart W.J., Atif K., Plateau B., The numerical solution of stochastic automata networks, European Journal of Operation Research, 86, 503-525, 1995.

[65] Unified Modelling Language, UML Semantics v. 1.1, Rational Software Corporation, September 1997.

[66] Uselton A.C., Smolka S.A., A Compositional Semantics for Statecharts using Labelled Transition Systems, State University of New York at Stony Brook, Proceedings of CONCUR 94, Lecture Notes in Computer Science 836, 2-17, Springer-Verlag, 1994.

[67] Uselton A.C., Smolka S.A., A Process Algebraic Semantics for Statecharts via State Refinement, State University of New York at Stony Brook, Proceedings of IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET), 262-281, June, 1994.

[68] von der Beeck, M., A Comparison of Statechart Variants, in Formal Techniques in Real-Time and Fault-Tolerant Systems (Langmaack, de Roever and Vytopil, eds.), Lecture Notes in Computer Science, vol. 863, Springer-Verlag, New York, 1994, pp. 128-148.

[69] Zamir S. (ed.), Handbook of Object Technology, CRC Press, 1999.

[70] Zieliński R., Generatory liczb losowych, WNT, Warszawa 1972.

mgr inż. Tomasz Babczyński

Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej
ul Janiszewskiego 11/17
50-372 Wrocław

Niniejszy raport otrzymują:

1. OINT	1 egz.
2. Biblioteka Główna PWr.	1 egz.
3. Z-ca Dyrektora Instytutu	1 egz.
4. Promotor	1 egz.
5. Recenzenci	2 egz.
6. Autor	2 egz.
Razem:	8 egz.

Raport wpłynął do Redakcji I-6
w czerwcu 2000 r.