Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

DYNAMICZNA ANALIZA SYSTEMÓW WSPÓŁBIEŻNYCH

Andrzej Karatkiewicz

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: a.karatkiewicz@iie.uz.zgora.pl

STRESZCZENIE

Analiza systemów współbieżnych jest skomplikowanym problemem, co spowodowane jest dużą ilością stanów osiągalnych w takim systemie. W niniejszym artykule podano zarys wybranych metod analizy systemów współbieżnych, których struktura opisana jest z wykorzystaniem sieci Petriego.

1. WPROWADZENIE

System sekwencyjny (np. algorytm albo program komputerowy) może być analizowany i weryfikowany na dwóch głównych poziomach: syntaktycznym, (który łatwo da się sformalizować; takiej analizy dokonuje każdy kompilator poprzez rozbiór gramatyczny) oraz semantycznym, (który nie jest w całości formalizowalny; do takiej weryfikacji potrzebny jest człowiek).

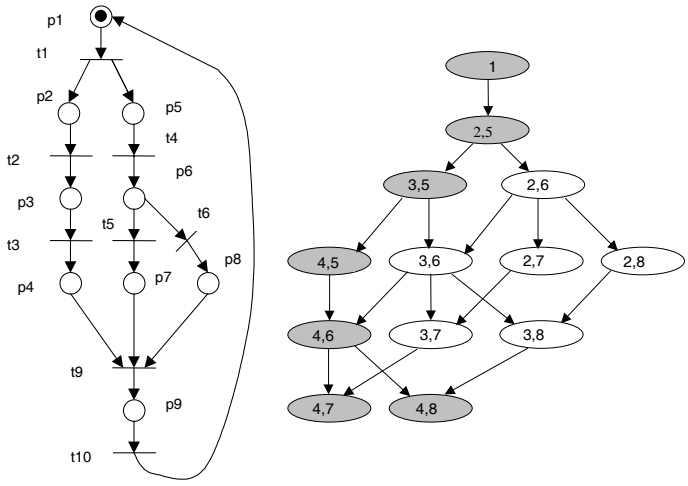
Przy analizie systemu współbieżnego (takiego jak sieć Petriego, diagram Statechart albo sieć automatowa) niezbędne jest wprowadzenie dodatkowego poziomu, znajdującego się pomiędzy wyżej wymienionymi. Dotyczy on wzajemnego oddziaływania współbieżnie aktywnych części systemu. Mogą one, na przykład, wzajemnie się zablokować albo w inny sposób zakłócić swoje działanie. Można to formalnie określić i formalnie sprawdzić, ale rozbiór gramatyczny wyżej wzmiankowanych problemów na takim poziomie nie wykryje. Stosuje się do tego inne metody. Niektóre z nich są tematem tegoż artykułu. Metody opisane są na poziomie analizy sieci Petriego, będących jednym z podstawowych modeli systemów współbieżnych.

2. SIECI PETRIEGO I ICH ANALIZA

Sieci Petriego [9] są popularnym modelem matematycznym i graficznym, służącym do opisu systemów współbieżnych. Sieć Petriego może być przedstawiona jako dwudzielny graf skierowany z dwoma rodzajami wierzchołków, które nazywane są *miejscami* i *tranzycjami*. Stan sieci Petriego, nazywany *znakowaniem*, zadaje się poprzez *znaczniki*, które mogą się znajdować w miejscach sieci. Jeśli znaczniki znajdują się we wszystkich wejściowych

miejscach tranzycji, to taka tranzycja jest *aktywna* i może zostać *zrealizowana*. Realizacja tranzycji zabiera jeden znacznik z każdego z miejsc wejściowych i dodaje jeden znacznik do każdego z miejsc wyjściowych tranzycji. *Blokadą* (ang. *deadlock*) nazywa się znakowanie, w którym żadna tranzycja nie jest aktywna. Sieć jest *żywą*, jeśli dla każdej tranzycji t z każdego znakowania M jest osiągalne znakowanie M', w którym tranzycja t jest aktywna. Sieć jest *bezpieczna*, jeśli w każdym z osiągalnych znakowań żadne miejsce nie zawiera więcej niż jeden znacznik. Sieć żywa i bezpieczna często nazywa się siecią *poprawnie zbudowaną* (ang. *well-formed*); takie sieci mają szczególne znaczenie praktyczne.

Na rys. 1 pokazany jest przykład sieci Petriego oraz jej przestrzeni stanów (grafu osiągalności).



Rys. 1. Przykład sieci Petriego i jej grafu osiągalności

Z przykładu tego widać, że współbieżność może powodować nadmierne zwiększenie ilości stanów osiągalnych w systemie. Ilość osiągalnych stanów wykładniczo zależy od rozmiaru systemu współbieżnego (jest to znane jako „problem eksplozji stanów", *state explosion problem*). Dlatego, chociaż graf osiągalności pozwala na łatwe znalezienie odpowiedzi na większość pytań, związanych z analizą sieci, praktycznie w taki sposób można badać tylko niewielkie sieci. Powstało szereg metod, pozwalających badać sieć w bardziej oszczędny sposób, na przykład, ograniczając się do konstruowania tylko częściowych grafów osiągalności [1].

3. ANALIZA POPRZEZ CZĘŚCIOWE BADANIE PRZESTRZENI STANÓW

Jedną z najbardziej znanych metod badania sieci poprzez „optymalną symulację", jest tak zwana „metoda upartych zbiorów" (*stubborn set method*) [10].

Niech sieć ma dwie współbieżne gałęzie i żadnej synchronizacji między nimi; niech każda gałąź ma n stanów. Łącznie osiągalnych stanów jest n^2, ale wynik końcowy ewolucji sieci byłby ten sam, gdyby każdą gałąź symulować osobno, a w trakcie takiej symulacji starczyłoby zbadać tylko 2n stanów. Metodę tę można rozszerzyć na dużo ogólniejsze przypadki.

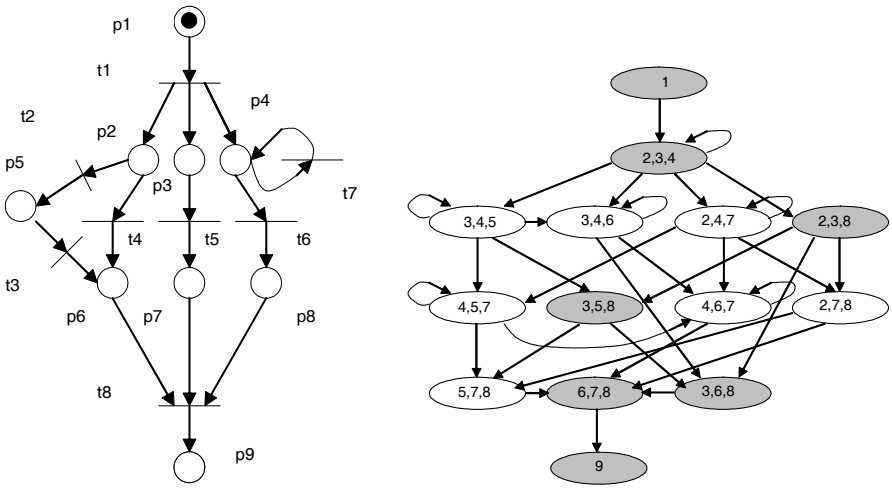
„Upartym zbiorem" dla znakowania M nazywa się zbiór tranzycji T_S, który spełnia 3 warunki: (1) każda nieaktywna tranzycja w T_S ma puste miejsce wejściowe p takie, że wszystkie tranzycje, dla których p jest miejscem wyjściowym, należą do T_S; (2) żadna aktywna tranzycja w T_S nie ma wspólnego miejsca wejściowego z tranzycją poza T_S; (3) T_S zawiera aktywną tranzycję. Metoda polega na tym, że w każdym danym znakowaniu symuluje się odpalenie tylko aktywnych tranzycji, należących do T_S. W taki sposób buduje się częściowy (zredukowany) graf osiągalności, pozwalający zbadać niektóre własności sieci. W swojej klasycznej wersji dla dowolnej sieci Petriego graf zawiera wszystkie blokady sieci. Na rys. 1 wierzchołki, zaznaczone na szaro, należą do zredukowanego grafu osiągalności.

Wykrywanie blokad ma szczególne znaczenie dla analizy systemów współbieżnych, bo z reguły prawidłowo działający system albo nie powinien mieć takich stanów (czyli działać w sposób cykliczny, jak typowy układ sterujący), albo powinien kończyć pracę w konkretnym określonym stanie (albo jednym z kilku określonych stanów). Metoda upartych zbiorów dobrze nadaje się do analizy w takim zakresie; w [4] pokazano, że poza wykrywaniem blokad pozwala ona również wykrywać znakowania, z których żadna blokada nie jest osiągalna.

Sprawdzanie takich własności sieci, jak żywotność i bezpieczeństwo, jest możliwe przy wykorzystaniu niektórych modyfikacji metody [10,11]. Ciekawe wyniki otrzymano, stosując metodę do niektórych podklas sieci. Na przykład, w modelowaniu, bazującym na sieciach Petriego, często stosuje się rozszerzone sieci swobodnego wyboru (EFC-sieci) z jednoelementowym znakowaniem początkowym. Dla takich sieci obliczenie „upartego zbioru" dla danego znakowania jest dużo prostsze niż w przypadku ogólnym. Stosowanie metody „upartych zbiorów" w jej klasycznej wersji pozwala uzyskać w takim przypadku więcej informacji, niż w przypadku ogólnym. W [7] zostało pokazane, że analizując zredukowany graf osiągalności dla takiej sieci, można sprawdzić, czy jest ona poprawnie zbudowana. Trwają prace nad zastosowaniem metody „upartych zbiorów" do sprawdzania poprawności zbudowania sieci w przypadku ogólnym. Na podstawie wymienionych metod zostały opracowane algorytmy analizy systemów, opisanych za pomocą diagramów SFC [7], diagramów Statechart [5], interpretowanych sieci Petriego z łukami zezwalającymi i zabraniającymi [8].

Inna metoda analizy poprzez częściowe badanie przestrzeni stanów bazuje na dekompozycji sieci na bloki i analizie tych bloków pojedynczo, w określonej kolejności [12]. Sumaryczna ilość osiągalnych znakowań każdego pojedynczego bloku może być wielokrotnie mniejsza,

niż ilość wszystkich osiągalnych znakowań sieci. Metoda ta pozwala obliczać blokady sieci oraz analizować żywotność i bezpieczeństwo sieci z jednoelementowym znakowaniem początkowym. Na rys. 2 pokazany jest prosty przykład jej zastosowania. Na pełnym grafie osiągalności sieci na szaro zaznaczone są znakowania rozpatrzone w metodzie blokowej.



Rys. 2. Przykład zastosowania blokowej metody analizy sieci

4. CZY MOŻNA JESZCZE BARDZIEJ OSZCZĘDZIĆ PAMIĘĆ?

Analiza sieci za pomocą metod, omówionych w poprzednim rozdziale, wymaga mniej czasu i pamięci, niż konstruowanie i badanie całej przestrzeni stanów. Jednak wymagany czas i pamięć nadal wykładniczo zależą od rozmiaru sieci, więc wydaje się, iż istotne są badania nad możliwością dalszego zmniejszenia złożoności obliczeniowej metod badania. Takie możliwości istnieją, chociaż, niestety, znane metody pozwalają oszczędzić pamięć kosztem czasu wykonywania algorytmu [11]. Jednakże w sytuacjach, w których pamięć jest parametrem bardziej krytycznym, takie oszczędzanie miałoby sens.

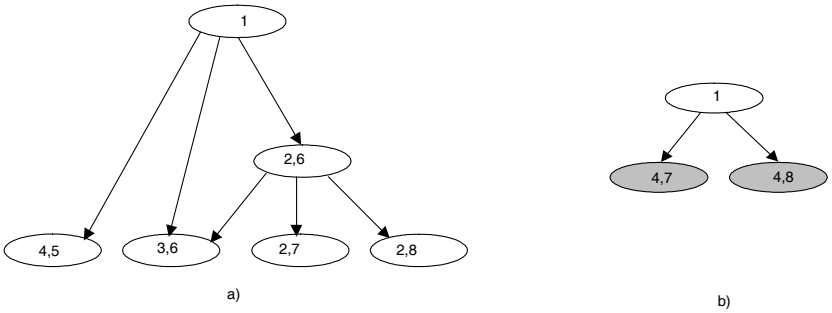
Zwrócimy jeszcze raz uwagę na problem wykrywania blokad. Podczas badania przestrzeni stanów w takim celu, znakowania nie będące blokadami, mają znaczenie tylko pod względem tego, które znakowania są z nich osiągalne. Więc czy koniecznie trzeba takie „przejściowe" znakowania przechowywać w pamięci po ich zbadaniu?

Gdyby algorytm badał przestrzeń stanów systemu, w którym są wzajemnie osiągalne stany i wyrzucałby z pamięci każdy stan po obliczeniu stanów bezpośrednio z niego osiągalnych, to by się zapętlił. Co najmniej jedno znakowanie z każdego cyklu w grafie osiągalności musi zostać w pamięci. W [3] udowodniono twierdzenie, na którego podstawie można tego dokonać.

Twierdzenie 1. Dla każdego cyklu grafu osiągalności sieci Petriego istnieje cykl grafu sieci taki, że każdej tranzycji, przez którą on przechodzi, odpowiada łuk cyklu grafu osiągalności.

To znaczy, że wystarczy znaleźć taki podzbiór tranzycji sieci, w którym znalazła by się co najmniej jedna tranzycja dla każdego cyklu sieci (jest to problemem „decyklizacji" grafu skierowanego [2]), po czym przechowywać w pamięci tylko znakowania, otrzymane przy realizacji tranzycji, należących do tego zbioru. Metoda obliczania w taki sposób blokad sieci (*dynamiczna redukcja grafu osiągalności*) opisana została w [6]; analizy niektórych innych własności sieci – w [3].

Na rys. 3 pokazany został dynamicznie redukowany graf osiągalności dla sieci z rys. 1 – dla jednego z kroków konstruowania oraz w postaci końcowej.



Rys. 3. Graf osiągalności dla sieci z rys. 1, budowany z zastosowaniem dynamicznej redukcji: a) w trakcie konstruowania (w momencie, kiedy zawiera maksymalną ilość wierzchołków); b) w postaci końcowej

Omówione podejście pozwala znacznie zmniejszyć ilość pamięci, potrzebnej do analizy sieci. Jednak czas analizy może okazać się znacznie większy, bo przy zastosowaniu dynamicznej redukcji może się okazać, że takie same znakowania sieci będą badane wielokrotnie. Pod tym względem skuteczne okazuje się połączenie metody dynamicznej redukcji z metodą „upartych zbiorów", bo wtedy taki niepożądany efekt występuje w mniejszym stopniu.

Jeśli trzeba tylko wykryć osiągalne blokady, ilość pamięci można jeszcze zmniejszyć, przechowując w pamięci nie graf, a tylko zbiór znakowań. Bardziej oszczędnym wobec tego dla pamięci byłby algorytm wykrywania blokad, łączący metodę „upartych zbiorów" z metodą dynamicznej redukcji, przechowując przy tym w pamięci tylko podzbiór osiągalnych znakowań.

5. PODSUMOWANIE

Chociaż analiza systemów współbieżnych poprzez badanie ich przestrzeni stanów jest problemem, wymagającym dużego nakładu czasu i pamięci, opracowane zostały metody, znacznie upraszczające taką analizę, zwłaszcza, jeśli dotyczy to niektórych podklas sieci,

mających szczególne znaczenie praktyczne. Z tego wynika, że formalna weryfikacja systemów współbieżnych w rozsądnym czasie i z wykorzystaniem ograniczonej pamięci jest możliwa.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2003-2006 jako projekt badawczy nr 4 T11C 006 24.

LITERATURA

[1] M. Heiner: *Petri net based system analysis without state explosion*, Proc. High Perfomance Computing '98, Boston, April 1998, session "Petri net Applications and HPC", pp. 394-403

[2] A. Karatkevich: *On algorithms for decyclisation of oriented graphs*, Proc. Discrete-Event System Design '01, Przytok, June 2001, session "Petri-Net Based Digital Design", pp. 35-40.

[3] А.Г. Короткевич: *Динамическая редукция графов достижимости сетей Петри*, Радиоэлектроника и информатика, No 1, 2002, С. 76-82.

[4] A. Karatkevich: *To behavior analysis of a class of Petri nets*, Proc. Workshop on Real-Time Programming '03, Łagów, May 2003, session "Formal methods in discrete control", pp. 33-38.

[5] A. Karatkevich: *Deadlock analysis in Statecharts*, Proc. Forum on Specification & Desogn Languages '03, Frankfurt, September 2003, session "Property Specification and Analysis", pp. 414-424.

[6] A. Karatkevich, M. Adamski: *Deadlock analysis of Petri nets: Minimization of memory amount*, Proc. Electronic Circuits and Systems '01, Bratislava, September 2001, pp. 69-72.

[7] A. Karatkevich, M. Adamski, M. Węgrzyn: *Rapid correctness analysis for Sequential Function Chart*, 45. Internationales Wissenschaftliches Kolloquium, Ilmenau, November 2000, Tagung "Eingebettete Systeme", pp. 679-684.

[8] A. Karatkevich, G. Andrzejewski: *Analiza wybranych własności interpretowanej sieci Petriego metodą optymalnej symulacji*, Mat. Krajowej Konferencji Elektroniki '02, Kołobrzeg – Dźwirzyno, czerwiec 2002, t. II, str. 685-690.

[9] J.L. Peterson: *Petri net theory and the modeling of systems*, Prentice-Hall, Inc., 1981

[10] A. Valmari: *State of the art report: Stubborn sets*, Petri Nets Newsletter, 46, 1994, pp. 6-14.

[11] A. Valmari: *The state explosion problem*, Lectures on Petri Nets I: Basic Models, LNCS Tutorials, LNCS 1491, Springer-Verlag, 1998, pp. 429-528.

[12] A. Zakrevskij, A. Karatkevich, M. Adamski: *A Method of analysis of operational Petri nets*, Proc. Advanced Computer Systems '01, Mielno, October 2001, session "Logic Synthesis and Simulation", Kluwer Academic Publishers, 2002, pp. 449-460.