II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

WYKORZYSTANIE LOGIKI SEKWENTÓW GENTZENA DO SYMBOLICZNEJ ANALIZY SIECI PETRIEGO

Jacek Tkacz

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

e-mail: J.Tkacz@iie.uz.zgora.pl

STRESZCZENIE

W referacie przedstawiono koncepcję i sposób analizy sieci Petriego z wykorzystaniem naturalnego wnioskowania Gentzena. Zastosowanie niniejszej metody pozwala na pominięcie kilku kroków w procesie analizy symbolicznej sieci, które opisywane są w literaturze i zastąpienie ich jednym bardziej abstrakcyjnym algorytmem. Istotną nowością w pracy jest wykorzystanie do tego celu autorskiego, eksperymentalnego systemu automatycznego wnioskowania metodą Gentzena. System ten został zaimplementowany w języku C i sprawdzony podczas rozwiązywaniu różnorodnych zagadnień o charakterze kombinatorycznym, obejmujących przestrzeń co najmniej kilkunastu zmiennych.

1. WPROWADZENIE

W literaturze opisywane są różne metody analizy sieci Petriego, które można podzielić na następujące cztery grupy [8]:

- metoda grafu znakowań,
- metody macierzowe,
- metody symboliczne,
- techniki redukcji.

Charakteryzują się one różną złożonością obliczeniową i efektywnością. Nadają się do badania różnego rodzaju klas sieci. Obecnie, w związku z rozwojem metod analizy symbolicznej, dostępnych jest coraz więcej algorytmów i narzędzi, możliwych do zastosowania w badaniach nad sieciami Petriego. Stosując metodę symboliczną można badać strukturalne własności sieci. Metoda symboliczna bazująca na logice Gentzena [1,7] i symbolicznym przetwarzaniu danych, polega na wyznaczeniu wszystkich blokad i pułapek w sieci z jednoczesnym zlokalizowaniem tych fragmentów, w których występują defekty.

2. SYSTEM GENTZENA

W systemie Gentzena zastosowano dziesięć reguł wnioskowania [1] dla następujących spójników logicznych: negacja „/”, dysjunkcja „+”, koniunkcja „*”, implikacja „->”

i równoważność <->. Dla każdego spójnika zostały podane dwie reguły jego eliminowania. Jedna z nich dotyczy sytuacji, gdy spójnik logiczny znajduje się po lewej stronie sekwentu, a druga, gdy spójnik umieszczony po prawej stronie sekwentu. Wyboru reguły dokonuje się poprzez lokalizację głównego spójnika względem znaku wynikania logicznego „|-”. Reguły Gentzena operują na logicznych zdaniach warunkowych, złożonych z warunku i skutku.

Reguły stosowane są zawsze do spójnika głównego formuły nieelementarnej w redukowanym sekwencie. Proces redukcji powtarzany jest tak długo, aż otrzymane zostaną same sekwenty znormalizowane, to jest sekwenty niezawierające żadnych spójników logicznych.

Dodatkowo do systemu Gentzena wprowadzono cztery reguły minimalizacji wyrażeń normalizowanych [1,7]: tautologii, sklejania sekwentów, pochłaniania oraz consensusu. Minimalizacja w tym przypadku, poza zmniejszeniem liczby wyników, ma znaczny wpływ na skrócenie przebiegu normalizacji. Odpowiednio wczesne zlokalizowanie sklejeń i tautologii pozwala na pominiecie analizy zbędnych sekwentów, które nie mają wpływu na wartość logiczną badanego wyrażenia.

Nowością w pracach nad algorytmem jest wprowadzenie dodatkowego spójnika *XOR* „<+>”, rozszerzającego system Gentzena o kolejne dwie reguły wnioskowania oraz wprowadzenie stałych, zawsze prawdziwej „1” i zawsze fałszywej „0”.

3. WYZNACZANIE BLOKAD I PUŁAPEK W SIECI PETRIEGO

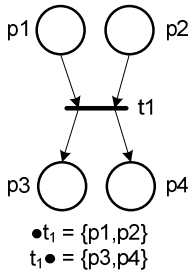
Struktura topologiczna sieci Petriego może być przedstawiona w postaci różnych wyrażeń logicznych. Jednym ze sposobów przedstawionych w literaturze jest opisanie pewnych własności strukturalnych sieci w postaci formuł Horna [2] i zastosowanie efektywnych algorytmów np. Thelena-Mathonego [8] do analizy tych własności. Możliwe jest zastąpienie algorytmu Thelena-Mathonego [4] naturalnym wnioskowaniem Gentzena. Właściwości systemu Gentzena umożliwiają analizę złożonych wyrażeń opisujących blokady i pułapki, bez konieczności transformacji ich do postaci formuł Horna.

a) Wyznaczanie blokad.

Na podstawie definicji blokady dla fragmentu sieci przedstawionego na rysunku (rys. 1) otrzymywane jest:

$$\text{jeżeli } p_3 \in S, \text{ to } p_1 \in S \text{ lub } p_2 \in S \text{ oraz} \quad (1)$$

$$\text{jeżeli } p_4 \in S, \text{ to } p_1 \in S \text{ lub } p_2 \in S \quad (2)$$



Rys. 1. Fragment sieci

Przedstawiając p_i za pomocą zmiennej logicznej x_i zależności (1) i (2) można zapisać w postaci wyrażeń:

$$x_3 \text{->} x_1+x_2 \tag{3}$$

$$x_4 \text{->} x_1+x_2. \tag{4}$$

Stosując własność:

$$a+b \text{->} c \equiv (a \text{->} c) \wedge (b \text{->} c) \tag{5}$$

otrzymuje się:

$$(x_3+x_4) \text{->} (x_1+x_2). \tag{6}$$

Po lewej stronie znaku implikacji znajduje się suma wszystkich zmiennych wyjściowych tranzycji t, natomiast po prawej stronie suma wszystkich zmiennych wejściowych:

$$\sum_{pi \in t\bullet} x_i \text{->} \sum_{pj \in \bullet t} x_j. \tag{7}$$

Dla całej sieci wyrażenie opisujące wszystkie tranzycje przyjmie postać:

$$\prod_{t \in T} \left(\sum_{pi \in t\bullet} x_i \text{->} \sum_{pj \in \bullet t} x_j\right). \tag{8}$$

Wszystkie wektory odpowiadające rozwiązaniom powyższego wyrażenia (8) odpowiadają blokadom w sieci Petriego.

a) Wyznaczanie pułapek

Na podstawie definicji pułapki dla fragmentu sieci przedstawionego na rysunku (rys. 1) otrzymywane jest:

$$\text{jeżeli } p_1 \in Q, \text{ to } p_3 \in Q \text{ lub } p_4 \in Q \text{ oraz} \tag{9}$$

$$\text{jeżeli } p_2 \in Q, \text{ to } p_3 \in Q \text{ lub } p_4 \in Q \tag{10}$$

Przedstawiając p_i za pomocą zmiennej logicznej x_i zależności (9) i (10) można zapisać w postaci wyrażeń:

$$x_1 \text{->} x_3+x_4 \tag{11}$$

$$x_2 \text{->} x_3+x_4. \tag{12}$$

Stosując własność:

$$a+b->c \equiv (a->c) \wedge (b->c) \tag{13}$$

otrzymuje się:

$$(x_1+x_2)->(x_3+x_4). \tag{14}$$

Po lewej stronie znaku implikacji znajduje się suma wszystkich zmiennych wejściowych tranzycji t, natomiast po prawej stronie suma wszystkich zmiennych wyjściowych:

$$\sum_{pi \in \bullet t} x_i -> \sum_{pj \in t \bullet} x_j \tag{15}$$

Dla całej sieci wyrażenie opisujące wszystkie tranzycje przyjmie postać:

$$\prod_{t \in T} \left(\sum_{pi \in \bullet t} x_i -> \sum_{pj \in t \bullet} x_j\right). \tag{16}$$

Wszystkie wektory odpowiadające rozwiązaniom powyższego wyrażenia (16) odpowiadają pułapkom w sieci Petriego.

Aby poddać normalizacji Gentzena wyrażenia opisujące blokady i pułapki, należy zapisać je w postaci sekwentów:

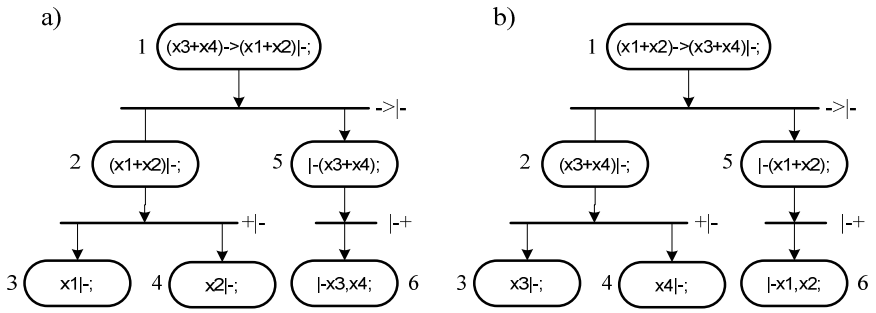
- blokady

$$(x_3+x_4)->(x_1+x_2)|-; \tag{17}$$

- pułapki

$$(x_1+x_2)->(x_3+x_4)|-; \tag{18}$$

Poniższy rysunek (rys. 2) przedstawia graficzne reprezentacje normalizacji sekwentów (17,18) według reguł Gentzena, opisujących blokady i pułapki dla fragmentu sieci (rys. 1).



Rys. 2. Normalizacja wyrażeń wg reguł Gentzena: a) blokady; b) pułapki.

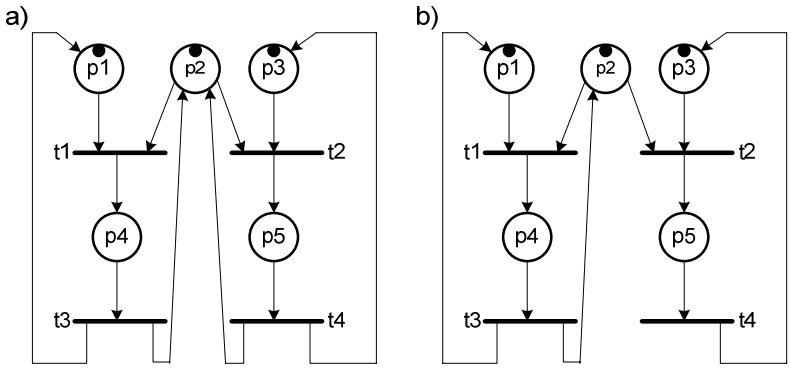
Blokady i pułapki występują po lewej stronie znaku wynikania logicznego w znormalizowanych sekwentach:

- blokady „x_1” i „x_2”,
- pułapki „x_3” i „x_4”.

4. BADANIE ŻYWOTNOŚCI SIECI Z ZASTOSOWANIEM LOGIKI GENTZENA

Algorytm wyznaczania wszystkich blokad i pułapek nadaje się do sprawdzania żywotności sieci Petriego [5,6]. Aby stwierdzić, czy dana sieć jest żywa, należy zbadać zależności między blokadami a pułapkami. Po wyznaczeniu rozwiązań dla normalizowanych sekwentów opisujących blokady i pułapki, należy sprawdzić zgodnie z własnością Commonera [3], czy każda wyznaczona blokada zawiera oznakowaną pułapkę. W przypadku pułapek należy sprawdzić, czy każda wyznaczona pułapka zawiera oznakowaną blokadę.

Zastosowanie algorytmu Gentzena automatyzuje proces badania żywotności. Możliwe jest połączenie procesu wyznaczania blokad i pułapek wraz z badaniem zależności pomiędzy nimi. Jako przykład przedstawione zostaną symboliczne procesy badania sieci żywej (rys. 3a) i nieżywej (rys. 3b). Ze względu na dużą liczbę obliczeń, dokumentacja z poszczególnych przebiegów normalizacji została pominięta.



Rys. 3. Przykład sieci a) żywej, b) nieżywej z klasy AC

Sekwenty opisujące blokady i pułapki dla sieci żywej (rys. 3a), uzyskane według algorytmu opisanego w rozdziale 3, będą miały postać:

- blokady

$$((p4->(p1+p2)) * (p5->(p2+p3)) * ((p1 + p2)->p4) * ((p2 + p3)->p5))|-; \quad (19)$$

- pułapki:

$$((p1+p2)->p4) * ((p2+p3)->p5) * (p4->(p1 + p2)) * (p5->(p2 + p3))|-; \quad (20)$$

W wyniku normalizacji sekwentów (19,20) otrzymane zostaną wszystkie wyrażenia opisujące blokady i pułapki w sieci (tab. 1). W zależności od normalizowanego wyrażenia lewa strona każdego sekwentu opisuje blokadę lub pułapkę.

Tab. 1. Wyznaczone blokady i pułapki dla sieci żywej

Sekwenty blokad	Sekwenty pułapek	Blokady	Pułapki
p1,p3,p4,p5|-;	p4,p5,p1,p3|-;	{p1,p3,p4,p5}	{p1,p3,p4,p5}
p2,p4,p5|-;	p4,p5,p2|-;	{p2,p4,p5}	{p2,p4,p5}
p1,p4|-p2,p3,p5;	p4,p1|-p5,p2,p3;	{p1,p4}	{p1,p4}
p3,p5|-p1,p2,p4;	p5,p3|-p4,p1,p2;	{p3,p5}	{p3,p5}
|-p2,p3,p1,p5,p4;	|-p5,p4,p2,p3,p1;		

Wszystkie blokady i pułapki wzajemnie się zawierają. Oznacza to, że badana sieć jest żywa.

Sekwenty opisujące blokady i pułapki dla sieci nieżywej (rys. 3b) będą miały postać:

- blokady:

$$((p4->(p1+p2)) * (p5->(p3+p2)) * ((p1 + p2)->p4) * (p3->p5))|-; \quad (21)$$

- pułapki:

$$((p1+p2)->p4) * ((p3+p2)->p5) * (p4->(p1 + p2)) * (p5->p3)|-; \quad (22)$$

W wyniku normalizacji wyrażeń (21,22) otrzymano sekwenty dla blokad i pułapek (tab. 2).

Tab. 2. Wyznaczone blokady i pułapki dla sieci nieżywej

Sekwenty blokad	Sekwenty pułapek	Blokady	Pułapki
p1,p3,p4,p5|-;	p4,p5,p1,p3|-;	{p1,p3,p4,p5}	{p1,p3,p4,p5}
p2,p4,p5|-;	p4,p5,p2,p3|-;	{p2,p4,p5}	{p2,p3,p4,p5}
p2,p4|-p3;	p4,p1|-p5,p3,p2;	{p2,p4}	{p1,p4}
p1,p4|-p3,p5;	p5,p3|-p4,p1,p2;	{p1,p4}	{p3,p5}
p3,p5|-p1,p2,p4;	|-p5,p4,p3,p2,p1;	{p3,p5}	
|-p3,p1,p2,p5,p4;			

W tabeli (tab. 2) zaznaczono blokady, które nie zawierają żadnych pułapek, co oznacza, że badana sieć nie jest żywa.

W celu całkowitego zautomatyzowania mechanizmu badania żywotności sieci Petriego w oparciu o system Gentzena, należy zbudować sekwent z wyrażeń opisujących zarówno blokady i pułapki. Ogólny zapis takiego sekwentu będzie wyglądał następująco:

$$|- \prod_{t \in T} (\sum_{pi \in t\bullet} x_i -> \sum_{pj \in \bullet t} x_j) <-> \prod_{t \in T} (\sum_{pi \in \bullet t} x_i -> \sum_{pj \in t\bullet} x_j); \quad (23)$$

Konstrukcja taka zakłada, że wszystkie blokady zawierają pułapki i jednocześnie wszystkie pułapki zawierają blokady. Wynikiem normalizacji sekwentu (23) będą pułapki lub blokady,

które się wzajemnie nie zawierają. Jeżeli w procesie normalizacji nie powstanie żaden sekwent, będzie to oznaczało, że badana sieć jest żywa.

W wyniku normalizacji sekwentu powstałego według wzoru (23), dla przykładowej sieci z rysunku 3a:

$$|\text{-}(((p4\text{->}(p1+p2)) * (p5\text{->}(p2+p3)) * ((p1 + p2)\text{->}p4) * ((p2 + p3)\text{->}p5)))$$

$$<\text{->}(((p1+p2)\text{->}p4) * ((p2+p3)\text{->}p5) * (p4\text{->}(p1 + p2)) * (p5\text{->}(p2 + p3))); \quad (24)$$

nie powstał żaden sekwent, co potwierdza fakt, że badana sieć jest żywa.

W przypadku normalizacji sekwentu dla sieci z rysunku 3b:

$$|\text{-}(((p4\text{->}(p1+p2)) * (p5\text{->}(p3+p2)) * ((p1 + p2)\text{->}p4) * (p3\text{->}p5)))$$

$$<\text{->}(((p1+p2)\text{->}p4) * ((p3+p2)\text{->}p5) * (p4\text{->}(p1 + p2)) * (p5\text{->}p3)); \quad (25)$$

powstaną sekwenty (26,27) jednoznacznie wskazujące blokady i pułapki, które nie pokrywają się wzajemnie, informując o tym, że badana sieć nie jest żywa.

$$p2,p4|\text{-}p3,p5; \quad (26)$$

$$p2,p4,p5|\text{-}p3; \quad (27)$$

Metoda oprócz zbadania żywotności sieci nie pozwala na stwierdzenie, czy dany sekwent wynikowy jest blokadą, czy pułapką. Taka identyfikacja możliwa jest, jeżeli normalizacji poddane zostaną następujące sekwenty:

- blokady |- pułapki:

$$\prod_{t\in T} (\sum_{pi\in t\bullet} x_i \text{->} \sum_{pj\in \bullet t} x_j) \text{ |- } \prod_{t\in T} (\sum_{pi\in \bullet t} x_i \text{->} \sum_{pj\in t\bullet} x_j); \quad (28)$$

- pułapki |- blokady:

$$\prod_{t\in T} (\sum_{pi\in \bullet t} x_i \text{->} \sum_{pj\in t\bullet} x_j) \text{|- } \prod_{t\in T} (\sum_{pi\in t\bullet} x_i \text{->} \sum_{pj\in \bullet t} x_j); \quad (29)$$

W przypadku badania sieci żywej obydwa wyrażenia nie będą miały rozwiązań. W przypadku sieci nieżywej wynikiem normalizacji wyrażenia (blokady |- pułapki;) będą pułapki niezawierające blokad, natomiast dla wyrażenia (pułapki |- blokady,) wynikiem będą blokady niezawierające pułapek. Może wystąpić sytuacja, kiedy tylko jedno z wyrażeń będzie posiadało rozwiązania, tak jak ma to miejsce w badanym przykładzie, gdzie dla wyrażenia (29) wynikiem będą blokady niezawierające pułapek..

5. PODSUMOWANIE, WYNIKI EKSPERYMENTÓW.

Wszystkie normalizacje wykonywane były przy pomocy eksperymentalnego oprogramowania na komputerze klasy P4 z zegarem 2Ghz. Poniższa tabela (tab. 3) przedstawia czasy normalizacji oraz liczby węzłów drzewa dowodu informujące o liczbie wykonywanych operacji. Należy zwrócić też uwagę, że zastosowany algorytm aplikacji często wykonuje więcej niż jedną operację w jednym węźle drzewa oraz pomija możliwie jak

najwcześniej obliczenia, które nie mają wpływu na wartość logiczną badanego wyrażenia. Tabela 3 zawiera wyniki testów implementacji algorytmu Gentzena dla przykładu z rozdziału 4, dotyczącego badania żywotności sieci.

Tab. 3. Wyznaczone blokady i pułapki dla sieci nieżywej

Rodzaj sekwentu	Rodzaj sieci	Liczba węzłów	Czas
|-blokady<->pułapki;	żywa	148	<1s
blokady|-pułapki;	żywa	77	<1s
pułapki|-blokady;	żywa	71	<1s
|-blokady<->pułapki;	nieżywa	153	<1s
blokady|-pułapki;	nieżywa	72	<1s
pułapki|-blokady;	nieżywa	80	<1s

Eksperymentalny program wnioskujący wykonywał wszystkie przykładowe normalizacje w czasie mniejszym od 1 sekundy. Liczba węzłów w drzewie dowodu nie przekroczyła liczby 153. W odróżnieniu od kosztownych lub nadmiarowych systemów profesjonalnych, układ wnioskujący Gentzena umożliwia klarowną, w miarę szybką analizę symboliczną sieci Petriego.

LITERATURA

[1] M. Adamski: *Projektowanie układów cyfrowych systematyczną metodą strukturalną*, Wydawnictwo Wyższej Szkoły Inżynierskiej w Zielonej Górze, Zielona Góra 1990

[2] K. Barkaoui, M. Minoux: *A polynomial-time graph algorithm to decide liveness of some basic classes of bounded Petri nets.* Lecture Notes in Computer Science, Berlin 1992:Springer Verlag, Vol.616,ss.62-75

[3] F. Commoner: *Deadlocks in Petri Nets.* - Wakefield 1972: Applied Data Res. Inc

[4] H. J. Mathony: *Uniwersal logic design algorithm and its application the synthesisof two-lewel switching circuits.* IEE Proceedings Letters, Elsevier Science Publishers (North Holland),Vol.29,1990,pp. 195-210

[5] T. Murata: *Petri nets:properties, analysis and applications.* Proceedings of the IEEE 1989, Vol.77, No4, ss. 541-580

[6] Z. Suraj, M. Szpyrka: *Sieci Petriego I PN-Tools*. Wydawnictwo Wyższej Szkoły Pedagogicznej, Rzeszów 1999

[7] Z. Pawlak: *Automatyczne dowodzenie twierdzeń.* Państwowe Zakłady Wydawnictw Szkolnych, Warszawa 1965

[8] A. Węgrzyn: *Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sterowania binarnego*. Rozprawa doktorska, Politechnika Warszawska, Warszawa, 2002