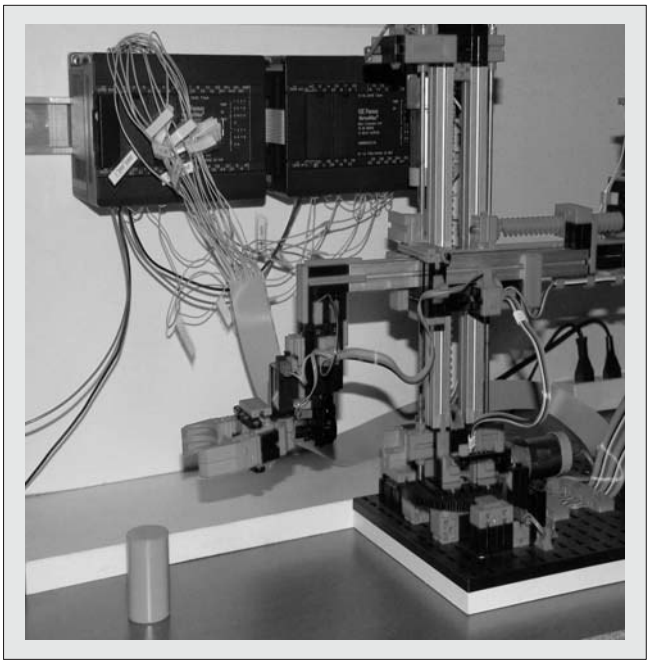
Grzegorz ANDRZEJEWSKI*

Redukcja liczby zmiennych kodujących w interpretowanych sieciach Petriego

Realizacja systemów sterowania binarnego jest istotną dziedziną badań naukowych. Popularnym modelem specyfikacji formalnej takich systemów są m. in. interpretowane sieci, nazywane dalej sieciami Petriego. Sposoby ich realizacji z wykorzystaniem różnych platform implementacyjnych przedstawiono m. in. w pracach [1, 2, 3, 4, 5, 6]. Dla realizacji sprzętowych jest istotna nie tylko sama topologia sieci, ale również sposoby przechowywania w sterowniku informacji o stanie bieżącym tej sieci. Od właściwego sposobu realizacji wymienionego problemu zależy w głównej mierze jakość implementacji zarówno w sensie zajętości zasobów systemu, jak i czasów jego reakcji.

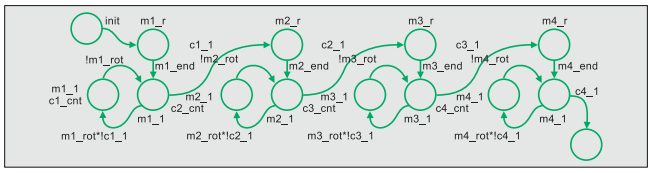
Ze względu na istotę działania sieci, jej miejsca mogą być oznakowane lub nie . Wobec tego naturalne jest przyporządkowanie do miejsc sieci zmiennych logicznych, mogących reprezentować ich stan. Istnieją różne metody przyporządkowywania takich zmiennych do miejsc. Zarówno przyporządkowanie, jak i proces przyporządkowywania, nosi nazwę kodowania sieci. Najprostszym kodowaniem jest przyporządkowanie osobnej zmiennej logicznej do każdego miejsca sieci (*one-to-one*). Jednak takie rozwiązanie może być nieprzydatne dla platform implementacyjnych, w których występuje ograniczona liczba dostępnych zmiennych logicznych, np. w sterownikach **PLC** serii nano lub mikro. W literaturze przedstawiono sposoby redukcji zmiennych kodujących [1, 7, 8]. Algorytmy te nie są jednak proste i nie zawsze dają dobre efekty dla sieci o prostych topologiach.



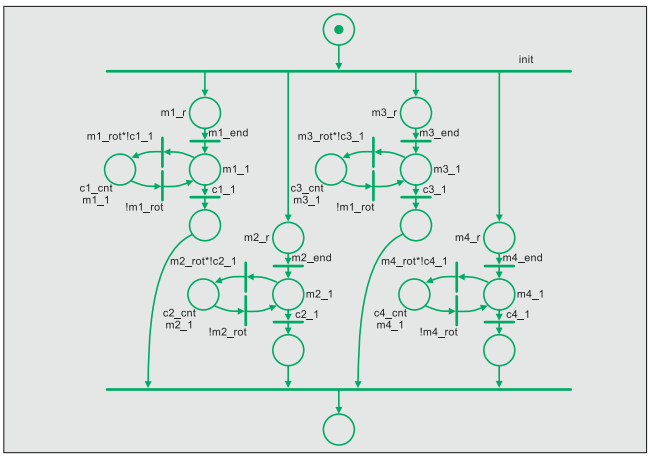
Rys. 1. Robot 3D firmy Fischerwerke Artur Fischer Gmbh & Co. KG

Bardzo często zdarza się w praktyce, że pewien proces można opisać w sposób sekwencyjny, ale dla skrócenia czasu jego wykonywania wyszukuje się w nim pewne fragmenty, które mogą zostać wykonane niezależnie (do pewnego stopnia) od innych. Fragmenty takie, nazywane podprocesami, same w sobie mają strukturę sekwencyjną i mogą być względem siebie wykonywane współbieżnie [4, 9]. Prostym przykładem takiego procesu jest kalibracja robota 3D (Fischerwerke Artur Fischer Gmbh & Co. KG), rys. 1 [10].

* Uniwersytet Zielonogórski, Instytut Informatyki i Elektroniki, e-mail: g.andrzejewski@iie.uz.zgora.pl



Rys. 2. Fragment opisu sekwencyjnego procesu kalibracji robota 3D



Rys. 3. Fragment opisu współbieżnego procesu kalibracji robota 3D

Na rys. 2 i 3 przedstawiono proces kalibracji robota, opisany sekwencyjnie z wykorzystaniem **FSM** (*Finite State Machine*) oraz współbieżnie z wykorzystaniem sieci Petriego.

Dla sieci o takiej topologii można wskazać prosty sposób kodowania, oparty na wyodrębnianiu sekwencyjnych fragmentów sieci, będących względem siebie w stosunku współbieżności.

ALGORYTM KODOWANIA

Dla pełniejszego zrozumienia zasad kodowania poniżej zamieszczono kilka definicji porządkujących.

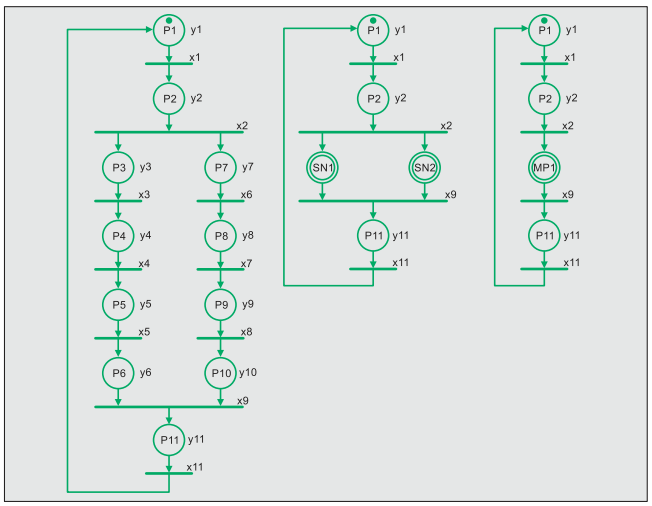
Definicja 1. Sieć (podsieć) nazywana jest automatową, jeśli każda tranzycja tej sieci (podsieci) ma jedno miejsce wejściowe i jedno miejsce wyjściowe. Przykładem podsieci automatowej jest podsieć złożona z miejsc {P1, P2} z rys. 4.

Definicja 2. Maksymalną podsiecią automatową nazywa się taką podsieć, do której nie można już dołączyć innych miejsc sieci, z której tworzy się podsieć, tak aby nie naruszyć jej własności wynikających z definicji 1. Przykładem maksymalnej podsieci automatowej jest podsieć złożona z miejsc {P3, P4, P5, P6} z rys. 4.

Definicja 3. Makromiejscem jest nazywane miejsce wyróżnione zastępujące pewną podsieć. Przykładem makromiejsca jest miej-

sce SN1 oznaczone podwójnym okręgiem na rys. 4, zastępujące podsieć {P3, P4, P5, P6}.

Definicja 4. Dwie podsieci są nazywane współbieżnymi względem siebie, jeśli można je zawrzeć w makromiejscach mających wspólną tranzycję wejściową i wspólną tranzycję wyjściową. Przykładem podsieci współbieżnych względem siebie są podsieci



■ **Rys. 4. Przykład testowy xmp2 sieci Petriego wraz z transformacjami**

{P3, P4, P5, P6} oraz {P7, P8, P9, P10} z rys. 4. Mogą być one zastąpione odpowiednio przez makromiejsca SN1 i SN2, które mają jedną wspólną tranzycję wejściową i jedną wspólną tranzycję wyjściową.

Definicja 5. Dwie maksymalne podsieci automatowe są nazywane sekwencyjnymi względem siebie, jeśli nie są względem siebie współbieżne. Przykładem maksymalnych podsieci automatowych sekwencyjnych względem siebie są podsieci {P3, P4, P5, P6} oraz {P12, P13, P14, P15} z rys. 6.

Definicja 6. Siecią końcową nazywa się taką sieć, w której nie ma podsieci współbieżnych względem siebie. Przykładem sieci końcowej jest sieć złożona z miejsc {P1, P2, MP1, P11} z rys. 4.

Proponowany algorytm zakłada, że sieć spełnia wymagania topologiczne przedstawione wcześniej.

Krok 1. Wyszukanie maksymalnych podsieci automatowych współbieżnych względem siebie.

Jeśli odnaleziono współbieżne podsieci automatowe, to należy przejść do kroku 2, a jeśli nie, to do kroku 4.

Krok 2. Przydzielenie zmiennych kodujących dla każdej z podsieci.

Liczbę zmiennych kodujących dla podsieci automatowej określa się z zależności:

$$\log_2 |SN| \le n < \log_2 |SN| + 1 \qquad (1)$$

gdzie: $n \in N$ jest liczbą zmiennych kodujących, $|SN|$ jest liczbą miejsc w podsieci.

Krok 3. Zastąpienie podsieci wyszukanych w kroku 1 makromiejscem. Przejście do kroku 1.

Krok 4. Przydzielenie zmiennych kodujących do sieci końcowej.

Liczbę zmiennych wyznacza się jak w kroku 2.

Przykład nieco prostszej sieci i jej kolejnych transformacji przedstawiono na rys. 4.

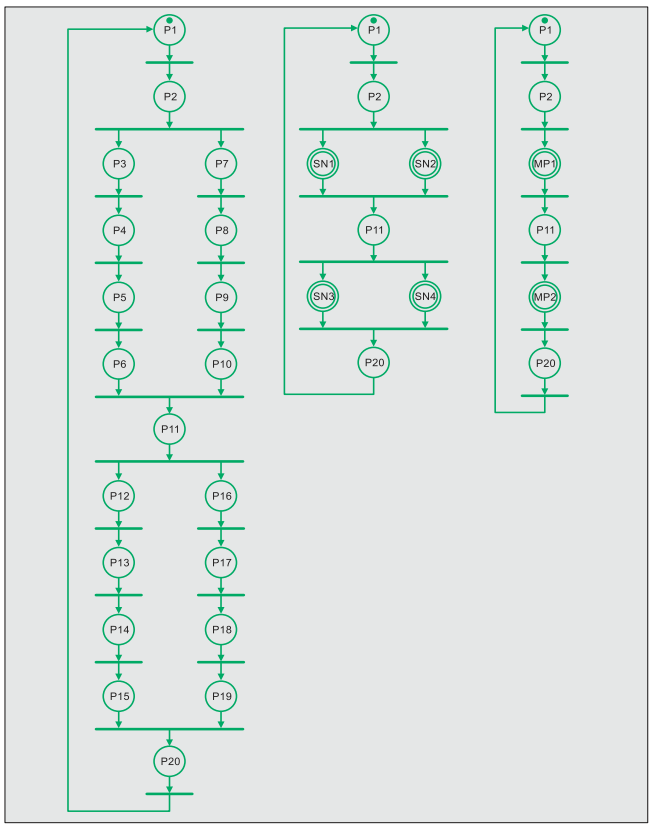
Dla prezentowanej sieci w pierwszym kroku są wyznaczone dwie maksymalne podsieci automatowe SN1 = {P3, P4, P5, P6} i SN2 = {P7, P8, P9, P10}. Pozostają one w stosunku współbieżności względem siebie. W kroku drugim przydziela się dla nich zmienne kodujące: {Q2, Q3} dla SN1 oraz {Q4, Q5} dla SN2. W następnym kroku są one zastępowane przez jedno makromiejsce MP1 = {SN1, SN2}. Ponieważ nie ma już żadnych podsieci współbieżnych, wobec tego w ostatnim kroku jest wyznaczana główna sieć automatowa SN3 = {P1, P2, MP1, P11} oraz są do niej przydzielone zmienne kodujące {Q0, Q1}. Przykład kodowania na wyznaczonej liczbie zmiennych kodujących przedstawiono na rys. 5.

Wynikiem działania algorytmu jest zakodowanie powyższej sieci z wykorzystaniem sześciu zmiennych kodujących, a nie jedenastu w przypadku kodowania metodą *one-to-one*. Można jeszcze ograniczyć liczbę zmiennych kodujących wprowadzając pewną modyfikację algorytmu, polegającą na wielokrotnym wykorzystaniu zmiennych kodujących podsieci, które są względem siebie sekwencyjne. Przykładem niech będzie sieć z rys. 6.

	Q0	Q1	Q2	Q3	Q4	Q5
P1	0	0	–	–	–	–
P2	0	1	–	–	–	–
P3	1	1	0	0	–	–
P4	1	1	0	1	–	–
P5	1	1	1	1	–	–
P6	1	1	1	0	–	–
P7	1	1	–	–	0	0
P8	1	1	–	–	0	1
P9	1	1	–	–	1	1
P10	1	1	–	–	1	0
P11	1	0	–	–	–	–

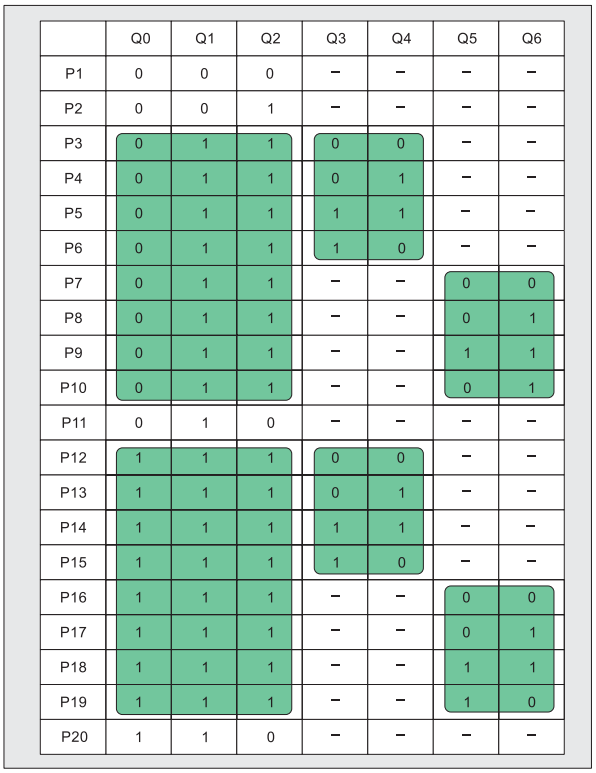
■ **Rys. 5. Tablica kodowania dla sieci z rys. 4**

Dla przykładu z rys. 6 algorytm wyszukuje dwie współbieżne podsieci automatowe SN1 = {P3, P4, P5, P6} i SN2 = {P7, P8, P9, P10}. Następnie przydziela dla nich zmienne kodujące {Q3,



■ **Rys. 6. Przykład testowy xmp2 sieci Petriego wraz z transformacjami**

Q4} dla SN1 oraz {Q5, Q6} dla SN2 i zastępuje je makromiejscem MP1 = {SN1, SN2}. W kolejnym kroku są odnajdywane współbieżne podsieci automatowe SN3 = {P12, P13, P14, P15} i SN4 = {P16, P17, P18, P19} i zamiast przydzielać do nich cztery no-



	Q0	Q1	Q2	Q3	Q4	Q5	Q6
P1	0	0	0	–	–	–	–
P2	0	0	1	–	–	–	–
P3	0	1	1	0	0	–	–
P4	0	1	1	0	1	–	–
P5	0	1	1	1	1	–	–
P6	0	1	1	1	0	–	–
P7	0	1	1	–	–	0	0
P8	0	1	1	–	–	0	1
P9	0	1	1	–	–	1	1
P10	0	1	1	–	–	0	1
P11	0	1	0	–	–	–	–
P12	1	1	1	0	0	–	–
P13	1	1	1	0	1	–	–
P14	1	1	1	1	1	–	–
P15	1	1	1	1	0	–	–
P16	1	1	1	–	–	0	0
P17	1	1	1	–	–	0	1
P18	1	1	1	–	–	1	1
P19	1	1	1	–	–	1	0
P20	1	1	0	–	–	–	–

Rys. 7. Tablica kodowania dla sieci z rys. 6

we zmienne kodujące można wykorzystać poprzednie, tj. {Q3, Q4} dla SN3 oraz {Q5, Q6} dla SN4. Następnie są one zastępowane makromiejscem MP2 = {SN3, SN4}. Powtórny przydział jest możliwy, ponieważ podsieci zawarte w makromiejscach MP1 i MP2 są względem siebie sekwencyjne. W ostatnim kroku jest wyznaczana końcowa sieć automatowa SN5 = {P1, P2, MP1, P11, MP2, P20} oraz są do niej przydzielone zmienne kodujące {Q0, Q1, Q2}. Przykład kodowania dla wyznaczonej liczby zmiennych przedstawiono na rys. 7.

* * *

Przedstawione sposoby kodowania interpretowanej sieci Petriego wymagają mniej zmiennych kodujących, co ma niebagatelne znaczenie przy realizacji systemów sterowania na platformach z ograniczoną liczbą wewnętrznych zmiennych logicznych. Zestawienie tabelaryczne liczby zmiennych kodujących w zależności od sposobu kodowania, przedstawiono w tabeli 1.

Tabela. 1. Wyniki różnych wariantów kodowania dla prezentowanych przykładów

Test\kodowanie	Kodowanie *one-to-one*	Algorytm podstawowy	Algorytm zmodyfikowany
xmp1	18	10	10
xmp2	11	6	6
xmp3	20	11	7

Dalsze badania zostaną ukierunkowane na określenie wpływu metody kodowania na złożoność realizacyjną systemów opisanych interpretowanymi sieciami Petriego.

LITERATURA

[1] Adamski M.: *Projektowanie układów cyfrowych systematyczną metodą strukturalną,* Wydawnictwo WSI w Zielonej Górze, 1992

[2] Andrzejewski G.: *Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych*, Zielona Góra, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, 2003

[3] Andrzejewski G.: *Hierarchical Petri nets for digital controller design, in Design of embedded control systems*, New York, Springer, 2005

[4] Andrzejewski G., Skowroński Z.: *Zrównoleglanie algorytmów sterowania w systemach klasy PLC*, Pomiary Automatyka Kontrola, 2006, nr 6, wyd. spec.

[5] Andrzejewski G., Mróz P.: *Realizacja hierarchicznych sieci Petriego z wykorzystaniem sterowników klasy PLC*, Pomiary Automatyka Kontrola, 2007, nr 5

[6] Biliński K.: *Application of Petri nets in parallel controler design*, PhD thesis, University of Bristol, UK, Jan 1996

[7] Biliński K., Dagless E. L., Saul J. M., Adamski M.: *Parallel controller synthesis from a Petri net specification*, Proc. of the Conference on European Design Automation, Grenoble, France, 1994

[8] Bubacz P., Mstowski B., Adamski M.: *Nowoczesna implementacja heurystycznego algorytmu kodowania strukturalnego stanów lokalnych w automatach współbieżnych*, materiały II konferencji naukowej KNWS'05, Złotniki Lubańskie, Polska, 2005

[9] Skowroński Z.: *Translacja specyfikacji funkcjonalnej układów cyfrowych na sieć Petriego dla potrzeb syntezy systemowej*, Rozprawa doktorska, Szczecin 2000

[10] http: //www. fischertechnik. com

Artykuł recenzowany

*Aneta BAL**

Tworzenie obrazu gradientowego na potrzeby segmentacji wododziałowej barwnych obrazów cyfrowych[1)]

Obraz jest istotnym, a czasami jedynym, źródłem informacji o badanych obiektach lub systemach. Dlatego obraz, techniki obrazowania oraz wizja komputerowa nabierają coraz większego znaczenia w wielu dziedzinach życia. Przykładem zastosowań szeroko rozumianego obrazowania są m. in. takie dziedziny, jak: diagnostyka medyczna, wojskowość, monitoring środowiska czy kontrola jakości w przemyśle.

[1)] Praca finansowana ze środków na działalność statutową Instytutu Automatyki Politechniki Śląskiej w roku 2008 BK-209/RAu1/2008 t.4.

*** Politechnika Śląska, Instytut Automatyki, e-mail: aneta.bal@polsl.pl**

W celu pozyskania informacji zawartych w obrazie jest konieczne przeprowadzenie jego analizy. Jednak aby było to możliwe niezbędne jest wydzielenie w obrazie jego fragmentów, tzw. obszarów, mających znaczenie z punktu widzenia celu analizy, a drogą do tego celu jest proces *segmentacji obrazów*. Ma on istotne znaczenie dla ostatecznego wyniku analizy. Związek ten jest konsekwencją tego, że w wyniku segmentacji trudna do analizy reprezentacja pikselowa obrazu jest zastępowana reprezentacją obszarową, która umożliwia opisanie wyodrębnionych w procesie segmentacji obszarów przez ich cechy i atrybuty, które są wymagane na etapie analizy obrazów. Niewłaściwie przeprowadzo-