Iwona GROBELNA, Michał GROBELNY
UNIWERSYTET ZIELONOGÓRSKI,
Podgórna 50, 65-246 Zielona Góra

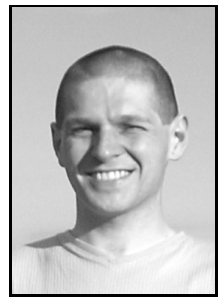
Projektowanie sterowników logicznych z wykorzystaniem łuków zezwalających i zakazujących sieci Petriego

Mgr inż. Iwona GROBELNA

Absolwentka Wydziału Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego oraz Fachhochschule Giessen-Friedberg (Niemcy). Od marca 2008 roku zatrudniona na stanowisku asystenta w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania naukowe obejmują metody weryfikacji specyfikacji systemów osadzonych. Członek Polskiego Towarzystwa Informatycznego.

e-mail: I.Grobelna@iie.uz.zgora.pl

Mgr inż. Michał GROBELNY



W roku 2007 ukończył studia na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego. Absolwent Zintegrowanych Studiów Zagranicznych Uniwersytetu Zielonogórskiego i Fachhochschule Giessen-Friedberg (Niemcy). Od października 2011 zatrudniony na stanowisku asystenta w Katedrze Mediów i Technologii Informacyjnych Uniwersytetu Zielonogórskiego. Zainteresowania naukowe obejmują metody specyfikacji osadzonych systemów sterowania. Członek Polskiego Towarzystwa Informatycznego.

e-mail: M.Grobelny@kmti.uz.zgora.pl

Streszczenie

Artykuł dotyczy zagadnień związanych z projektowaniem sterowników logicznych z wykorzystaniem łuków zezwalających i zakazujących sieci Petriego. Zaproponowano nowatorskie podejście do regułowej specyfikacji sterownika logicznego, przygotowanej w postaci abstrakcyjnego autorskiego modelu logicznego dogodnego do formalnej weryfikacji modelowej oraz syntezy logicznej. Szczególną uwagę zwrócono tutaj na łuki zakazujące i zezwalające interpretowanych sieci Petriego, ich realizację w abstrakcyjnym modelu logicznym i interpretację w innej postaci specyfikacji zachowania sterownika logicznego – diagramach aktywności języka UML.

Słowa kluczowe: specyfikacja behawioralna, rekonfigurowalny sterownik logiczny, UML, diagramy aktywności.

Logic controller design using enabling and inhibitor arcs of Petri nets

Abstract

The paper focuses on logic controller design using enabling and inhibitor arcs of Petri nets. There is proposed a novel original approach to rule-based specification of logic controller behaviour prepared as an abstract logical model suitable for formal verification and logic synthesis. Special interest is put on enabling and inhibitor (disabling) arcs of interpreted Petri nets, their realization in an abstract logical model and interpretation in other specification form – namely UML activity diagrams (in version 2.x). These arcs can be used for flow synchronization or controlled usage of shared resources. After a short introduction (Section 1), some basic concepts on logic controller specification are presented (Section 2), in particular considering (interpreted) Petri nets and UML (activity) diagrams. Usage of enabling and inhibitor arcs is shown on an example of the interpreted Petri net in Fig. 1 (transitions firing sequence in Fig. 2), followed by their representation in the proposed abstract rule-based logical model, its formal verification (using model checking technique) and synthesis (Section 3). The paper also proposes enabling and inhibitor arcs interpretation in UML activity diagrams (Section 4). Although direct representation of these arcs is not possible, the authors try to achieve an alternative solution which corresponds semantically to appropriate Petri net elements. Tab. 1 presents graphic representation of the considered arcs in interpreted Petri nets as well as in UML activity diagrams. The paper ends with a short summary (Section 5).

Keywords: behavioural specification, reconfigurable logic controller, UML, activity diagrams.

1. Wstęp

Specyfikacja procesu sterowania stanowi fundament projektu urządzenia. Dostępne są formalne techniki pozwalające na określenie zachowania projektowanego sterownika logicznego [1]. Jedną z nich są sieci Petriego [2, 3], które ze względu na swoją ugruntowaną pozycję posiadają wiele formalnych metod weryfikacji. Możliwa jest także synteza logiczna tak przygotowanej specyfikacji do rekonfigurowalnego sterownika logicznego lub sterownika PLC. Do potrzeb projektowania sterowników logicznych nadają się szczególnie interpretowane sieci Petriego [3], które uwzględniają właściwości kontrolowanych obiektów. Oprócz podstawowych elementów sieci, takich jak miejsca, tranzycje i łuki, dostępne są także łuki zezwalające i zakazujące [4], które mają wpływ na realizację odpowiednich tranzycji.

W artykule zaprezentowano sposób interpretacji łuków zezwalających i zakazujących interpretowanych sieci Petriego w autorskim regułowym abstrakcyjnym modelu logicznym. Wskazano sposób weryfikacji i syntezy regułowej formy specyfikacji uwzględniającej tego typu łuki. Zaproponowano także metodę interpretacji łuków w innej formalnej technice specyfikacji zachowania sterowników logicznych, a mianowicie diagramach aktywności języka UML (w wersji 2.x).

Artykuł podzielony jest następująco. Rozdział drugi omawia krótko aspekty związane ze specyfikacją zachowania sterownika logicznego, skupiając się na dwóch wybranych formalnych technikach – interpretowanych sieciach Petriego oraz diagramach aktywności języka UML. Następnie (w rozdziale trzecim) wprowadzono autorski regułowy model zachowania sterownika logicznego uwzględniającego łuki zezwalające i zakazujące, wskazując jednocześnie na możliwości jego formalnej weryfikacji oraz syntezy logicznej. Rozdział czwarty proponuje interpretację wspomnianych łuków sieci Petriego w alternatywnej formie specyfikacji – diagramach aktywności. Artykuł kończy się wnioskami.

2. Specyfikacja zachowania sterownika logicznego

Zachowanie projektowanego sterownika logicznego może zostać formalnie zapisane za pomocą (interpretowanej) sieci Petriego [3], sieci SFC czy też diagramów języka UML [5, 6], w szczególności diagramów aktywności [7]. Każda z technik specyfikacji posiada swoje zalety i wady, a zastosowanie w jednym projekcie dwóch form pozwala na wykorzystanie zalet obu.

(Interpretowane) sieci Petriego

Sieci Petriego [2,3] są modelem matematycznym ogólnego zastosowania opisującym relacje pomiędzy warunkami i zdarzeniami. Obecnie są wykorzystywane w wielu gałęziach przemysłu, m.in. do planowania i kontrolowania przepływu produkcji, projektowania i programowania sterowników logicznych oraz syntezy oprogramowania systemowego.

Interpretowane sieci Petriego [3, 4] jako formalny zapis specyfikacji działania sterownika logicznego modelują zachowanie współbieżnych procesów sterowania uwzględniając właściwości kontrolowanych obiektów. Istnieją dedykowane narzędzia do projektowania sieci Petriego, które ułatwiają pracę inżynierom. Część z nich wspiera dodatkowo weryfikację utworzonych sieci

pod kątem właściwości strukturalnych. Dostępne są także metody weryfikacji właściwości behawioralnych [8].

Łuki zezwalające i zakazujące pozwalają na obsługę współdzielonych zasobów, synchronizację przepływu danych czy też nadawanie priorytetów tranzycjom [4], co może być wykorzystane zwłaszcza przy procesach współbieżnych. Oznaczane są tak jak zwykłe łuki, z tą różnicą że rysowane są linią przerywaną, a łuk zezwalający kończy się okręgiem (graficzne przedstawienie łuków zestawiono w tab. 1). Dodatkowo, łuk zezwalający na realizację danej tranzycji nie zabiera markera z jego miejsca wejściowego. Przykład sieci z zastosowaniem łuków zamieszczono w rozdziale trzecim.

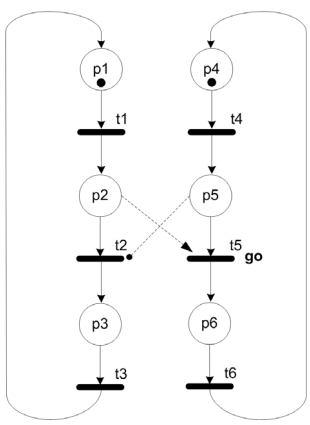
Diagramy aktywności języka UML

Notacja UML [5] (ang. *Unified Modelling Language*) w wersji 2.x upraszcza przepływ informacji pomiędzy członkami zespołu projektowego i umożliwia łatwe zrozumienie zachowania systemu nawet osobom bez doświadczenia. Początkowo UML został wprowadzony jako narzędzie do specyfikacji, wizualizacji i dokumentacji oprogramowania, jednakże bardzo szybko zyskał sobie zwolenników w innych dziedzinach. Systemy osadzone mogą być również z wyśmienitym skutkiem opisywane przy użyciu diagramów aktywności, maszyny stanów czy sekwencji [6, 7].

Diagramy aktywności języka UML (zwane także *diagramami czynności*) mogą być alternatywną lub uzupełniającą specyfikacją do sieci Petriego. Technika ta doskonale nadaje się do reprezentacji zachowania procesu sterowania, szczególnie uwzględniając jej role w trakcie konsultacji z osobą spoza świata techniki. Dużą zaletę odgrywa tutaj przejrzystość specyfikacji otrzymanej z wykorzystaniem diagramów aktywności. Stąd, diagramy aktywności mogą być elementem uzupełniającym projekt wykonany z wykorzystaniem sieci Petriego. Zważywszy na ten fakt zasadnym jest stosowanie technik transformacji [7] pomiędzy obiema wskazanymi technikami w celu uzyskania spójnych odpowiadających sobie specyfikacji.

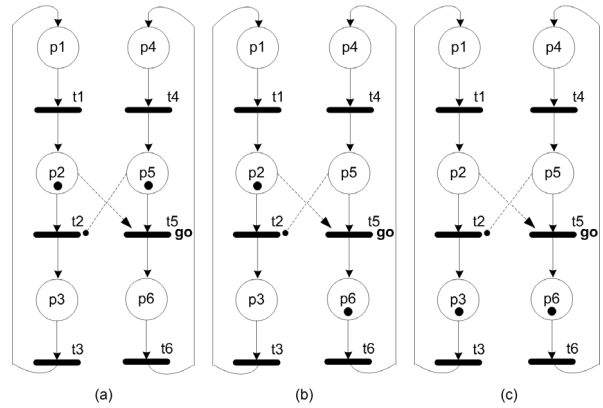
3. Autorski regułowy model zachowania sterownika logicznego

Zachowanie projektowanego sterownika logicznego formalnie przedstawione zostało za pomocą interpretowanej sieci Petriego (rys. 1, drobna modyfikacja przykładu z [4]).



Rys. 1. Interpretowana sieć Petriego – synchronizacja dwóch procesów
Fig. 1. Interpreted Petri Net – synchronization of two processes

Sieć obrazuje działanie dwóch procesów współbieżnych (miejsca *p1 – p3* oraz *p4 - p6*), które dzięki łukom zezwalającym i zakazującym zostały zsynchronizowane. Łuk zezwalający łączy miejsce *p2* z tranzycją *t5*, zaś łuk zakazujący – miejsce *p5* z tranzycją *t2*. W rezultacie, w przypadku aktywnego znakowania miejsc *p2* i *p5* (rys. 2a), najpierw wykonana zostanie tranzycja *t5* nieusuwająca markera z miejsca *p2* (rys. 2b), a następnie tranzycja *t2* (rys. 2c) i osiągnięte zostanie globalne znakowanie miejsc *p3* i *p6.*



Rys. 2. Interpretowana sieć Petriego – realizacja tranzycji
Fig. 2. Interpreted Petri Net – transitions firings

Model logiczny

Proponowany autorski model logiczny wykorzystywany jest zarówno do celów syntezy, jak i do weryfikacji modelowej. Stanowi format pośredni opisujący zachowanie projektowanego sterownika logicznego. Model [8, 9, 10] zawiera definicje zmiennych wraz z wartościami jakie mogą przyjmować. Miejsca, sygnały wejściowe oraz sygnały wyjściowe sieci Petriego są bezpośrednio odwzorowane w modelu logicznym jako zmienne. Model logiczny zawiera także zbiór reguł, które opisują zachowanie projektowanego systemu i określają zmiany wartości zmiennych z upływem czasu. Tranzycje interpretowane są jako wspomniane zbiory reguł. Przykładowo, dla sieci z rys. 1 reguły dotyczące tranzycji *t2* i *t5* wyglądają następująco (obie tranzycje uwzględniają znakowanie miejsc *p2* i *p5*):

```
t2: p2 & !p5 -> X (!p2 & p3);
t5: p2 & p5 & go -> X (!p5 & p6);
```

Sygnały wyjściowe w abstrakcyjnym modelu logicznym są aktywne przy aktywnym znakowaniu odpowiednich miejsc. Definiowane są również zmiany wartości sygnałów wejściowych (wykorzystywane tylko przy weryfikacji modelowej z uwagi na ograniczenie przestrzeni stanów).

Weryfikacja modelowa

Technika weryfikacji modelowej [11] (ang. *model checking*) pozwala na weryfikację właściwości opisujących zachowanie projektowanego systemu. Na etapie projektowania można zatem sprawdzić, czy zdefiniowany model spełnia stawiane mu wymagania.

Autorski model logiczny utworzony na podstawie interpretowanej sieci Petriego sterowania przekształcany jest do formatu wejściowego narzędzia weryfikującego NuSMV według ściśle określonych reguł [8, 9, 10].

Właściwości, jakie projektowany system ma spełniać, definiowane są przy wykorzystaniu logiki temporalnej. Należy mieć na uwadze fakt, że tylko wyspecyfikowane właściwości zostaną sprawdzone.

Synteza logiczna

Po pomyślnej weryfikacji modelowej model logiczny może być następnie przekształcony do modelu syntezowalnego, przykładowo w języku VHDL.

Model logiczny utworzony na podstawie interpretowanej sieci Petriego przekształcany jest do formatu wejściowego narzędzia weryfikującego NuSMV według ściśle określonych reguł [9, 10].

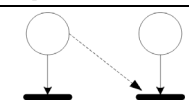
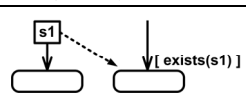
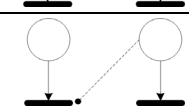
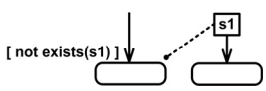
Utworzony model w języku VHDL może zostać wstępnie przesymulowany w środowisku *Active HDL* oraz przesyntezowany w środowisku *XILINX ISE*.

4. Interpretacja w diagramach aktywności języka UML

Pomimo wielu podobieństw pomiędzy diagramami aktywności UML i sieciami Petriego, nie jest niestety możliwym uzyskanie wprost odpowiadających sobie konstrukcji dla wszystkich elementów obu technik. Przykładem mogą być tutaj omawiane w artykule łuki zezwalające i zakazujące występujące w sieciach Petriego, jednakże niemożliwe do uzyskania wprost w diagramach aktywności UML (w wersji 2.x). Fakt ten wynika z bezstanowej natury diagramów aktywności, przez co nie ma możliwości połączenia stanu systemu z akcją (odpowiadającą tranzycji sieci Petriego), co byłoby stuprocentowym odpowiednikiem omawianych łuków.

Istnieją jednak sposoby zminimalizowania wpływu braku stanów i uzyskanie odpowiedników łuków zakazujących i zezwalających. Diagramy aktywności w ramach swoich elementów posiadają element o nazwie *obiekt*. W transformacji procesów sterowania może on posłużyć do odwzorowania specyficznego oczekiwanego stanu systemu poprzez jego utworzenie pomiędzy następującymi po sobie akcjami. W przypadku omawianych łuków element *obiekt* posłuży za sygnalizację stanu systemu, który zabrania lub zezwala na wykonanie powiązanej łukiem akcji. Dodatkowym elementem, który należy wprowadzić w tym miejscu w celu zachowania kompatybilności z diagramami aktywności określonymi przez konsorcjum OMG, jest wprowadzenie do warunku uruchomienia kolejnej tranzycji synchronizowanego procesu sprawdzenia istnienia obiektu odpowiadającego za odwzorowanie stanu systemu.

Tab. 1. Graficzne przedstawienie łuków zezwalających i zakazujących
Tab. 1. Graphic representation of enabling and inhibitor arcs

	Interpretowana sieć Petriego	Diagram aktywności UML
łuk zezwalający		
łuk zakazujący		

Odpowiadające sobie konstrukcje w sieciach Petriego i diagramach aktywności zaprezentowane zostały w tab. 1. W diagramach aktywności wykorzystywane są obiekt *s1* oraz odpowiedni warunek wykonania akcji, gdzie w łuku zezwalającym warunek przy synchronizowanej akcji będzie miał postać *exists(s1)*, gdzie *s1* jest omawianym obiektem odpowiadającym miejscu sieci Petriego reprezentującym stan synchronizowanego procesu. Alternatywnie dla odpowiedniku łuku zakazującego omawiany warunek będzie miał postać *not exists(s1)*, gdzie *s1* odpowiada omawianemu obiektowi. Dodatkowo należy tutaj podkreślić, że elementy zaproponowane w tab. 1 dla diagramu aktywności i zrealizowane przy pomocy przerywanych linii (strzałka i linia z kropką na końcu) nie są elementem specyfikacji diagramów aktywności UML określonej przez konsorcjum OMG. Wprowadzono te dodatkowe elementy w celu poprawienia czytelności diagramu. Ich wygląd bezpośrednio nawiązuje do strzałek zezwalających i zakazujących sieci Petriego. Należy jednakże pamiętać, że są to tylko elementy dodatkowe i nie sią one niezbędne, wystarczające w tym przypadku jest rozszerzenie warunków uruchomienia akcji o sprawdzenie istnienia obiektu (sygnalizacji pożądanego stanu systemu).

5. Wnioski

Proponowany autorski regułowy sposób reprezentacji nadaje się zarówno do formalnej weryfikacji, jak i do syntezy logicznej jako rekonfigurowalny sterownik logiczny. Dużą zaletą rozwiązania jest fakt, że model syntezowalny opisany jest w podobny sposób co model weryfikowalny. Po pomyślnej weryfikacji modelowej osiągamy zatem zweryfikowaną specyfikację behawioralną w języku logiki temporalnej, która będzie abstrakcyjnym programem funkcjonowania matrycowego rekonfigurowalnego sterownika logicznego. Program sterownika logicznego (implementacja) będzie więc poprawny względem jego pierwotnej specyfikacji.

Zastosowanie łuków zezwalających i zabraniających w interpretowanych sieciach Petriego pozwala na synchronizację przepływu oraz kontrolę współdzielonych zasobów, co ma szczególne znaczenie zwłaszcza przy procesach współbieżnych. Alternatywnie (lub łącznie) stosowana inna forma specyfikacji zachowania sterownika logicznego, mianowicie diagramy aktywności języka UML (w wersji 2.x), z uwagi na swoją bezstanową naturę nie posiadają bezpośrednich odpowiedników omawianych łuków. W artykule zaproponowano rozwiązanie opierające się wykorzystaniu obiektów do sygnalizacji stanu projektowanego systemu, w celu utrzymania kompatybilności z elementami określonymi przez konsorcjum OMG.

Tym samym możliwe jest zastosowanie dwóch technik specyfikacji sterownika logicznego w jednym projekcie – interpretowanych sieci Petriego oraz diagramów aktywności języka UML w wersji 2.x (uwzględniając możliwość dwukierunkowej transformacji pomiędzy nimi [7]), co pozwala na wykorzystanie zalet ich obu.

Autorzy są stypendystami w ramach Poddziałania 8.2.2 „Regionalne Strategie Innowacji", Działania 8.2 „Transfer wiedzy", Priorytetu VIII „Regionalne Kadry Gospodarki" Programu Operacyjnego Kapitał Ludzki współfinansowanego ze środków Europejskiego Funduszu Społecznego Unii Europejskiej i z budżetu państwa.

6. Literatura

[1] Gomes L., Barros J.P., Costa A.: Modeling formalisms for embedded system design. Embedded Systems Handbook, Taylor & Francis Group, 2006.
[2] Girault C., Valk R.: Petri Nets for Systems Engineering, A Guide to Modelling, Verification and Applications. Berlin Heidelberg: Springer-Verlag, 2003.
[3] David R., Alla H.: Discrete, Continuous, and Hybrid Petri Nets. Berlin Heidelberg: Springer-Verlag, 2010.
[4] Minns P., Elliott I.: FSM based Digital Design using Verilog HDL. Wiley 2008.
[5] Strona domowa Unified Modelling Language: www.uml.org
[6] Łabiak G., Adamski M., Tkacz J., Doligalski M., Bukowiec A.: Role of UML modelling in discrete controller design. ICSEng 2011, Proceedings of 21st International Conference on Systems Engineering, 2011 ss. 480-481.
[7] Grobelna I., Grobelny M., Adamski M.: Petri Nets and activity diagrams in logic controller specification - transformation and verification. Mixed Design of Integrated Circuits and Systems - MIXDES 2010: proceedings of the 17th international conference, 2010, ss. 607-612.
[8] Grobelna I., Adamski M.: Model checking of control interpreted Petri nets. Mixed Design of Integrated Circuits and Systems - MIXDES 2011: proceedings of the 18th international conference, 2011, ss. 621-626.
[9] Grobelna I.: Formal verification of embedded logic controller specification with computer deduction in temporal logic. Electrical Review, 2011, nr 12a, ss. 47-50.
[10] Grobelna I.: Regułowa reprezentacja interpretowanych sieci Petriego sterowania dla potrzeb syntezy i weryfikacji. Pomiary Automatyka Kontrola, nr 8, Vol. 57, 2011, ss. 942-944.
[11] Clarke E. M., Grumberg O., Peled D. A.: Model checking. Cambridge, The MIT Press, 1999.

otrzymano / received: 02.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*