Iwona GROBELNA
UNIWERSYTET ZIELONOGÓRSKI,
ul. Podgórna 50, 65-246 Zielona Góra

Weryfikacja modelowa interpretowanych sieci Petriego sterowania

Mgr inż. Iwona GROBELNA

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

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

Streszczenie

Artykuł przedstawia oryginalne podejście do weryfikacji modelowej interpretowanych sieci Petriego sterowania. Sieci Petriego są powszechnie wykorzystywane w przemyśle. Najczęściej jednak weryfikowane są pod kątem właściwości strukturalnych, a właściwości behawioralne (mimo ich dużego znaczenia) są pomijane. Technika weryfikacji modelowej pozwala na weryfikację właściwości opisujących zachowanie projektowanego systemu. Model logiczny otrzymany na podstawie istniejącej sieci Petriego sterowania przedstawiany jest na poziomie RTL w taki sposób, że nadaje się zarówno do formalnej weryfikacji, jak i do syntezy logicznej jako rekonfigurowalny sterownik logiczny lub PLC.

Słowa kluczowe: weryfikacja modelowa, sterownik logiczny, interpretowane sieci Petriego sterowania, logika temporalna.

Model checking of control interpreted Petri Nets

Abstract

The paper introduces a novel approach to model checking with Control Interpreted Petri Nets [15]. Petri Nets [9, 11, 12, 13] are commonly used in the industry. However, they are mostly verified against structural properties, and behavioral properties are out of scope. The model checking technique [3, 7, 8, 21, 22] allows verifying properties which describe behavior of the designed system. Properties to be verified are expressed in temporal logic [16, 17, 18, 19, 20]. The logical model (Fig. 1) derived from existing Petri net is presented at RTL level (*Register Transfer Level*) in such a way, that it is easy to be formally verified as well as to logical synthesized as a reconfigurable logic controller or PLC (*Programmable Logic Controller*). It operates on variables which correspond to places, input and output signals of the Control Interpreted Petri Net (Section 3). The variables change their values according to some specified rules. The logical model is afterwards transformed into input format of the NuSMV model checker [23] and formally verified (Section 4). Control Interpreted Petri Net (Fig. 2) is divided into elementary subnets (Fig. 3). Each elementary subnet consists of a single place and its input and output transitions. Each elementary subnet is interpreted as a single segment of model description in the NuSMV tool. Each elementary subnet represents a two-states state machine which is usually realized as a single macrocell (Fig. 4) in the FPGA circuit. The properties to be verified are expressed in LTL or CTL logic. If any of them is not satisfied in the described system model, the appropriate counterexample is generated (Fig. 6). In the example in the paper the verification finds a subtle error resulting from incorrect / incomplete specification (Fig. 5) and allows the user to localize the error source.

Keywords: model checking, logic controller, Control Interpreted Petri Nets, temporal logic.

1. Wstęp

Specyfikacja procesu sterowania stanowi fundament projektu urządzenia i z tego powodu jest jednym z najbardziej kluczowych momentów cyklu projektowania. Potencjalne błędy mogą mieć wpływ na kolejne etapy tworzenia systemu i tym samym zwiększyć koszty wytworzenia produktu końcowego. Szczególnie istotne jest wczesne wykrycie błędów przy projektowaniu bezpiecznych systemów osadzonych [1]. Drobne błędy mogą wtedy wpłynąć na działanie systemu i spowodować tragiczne w skutkach następstwa [2].

Intepretowane sieci Petriego sterowania są wykorzystywane w przemyśle jako formalna specyfikacja sterowników logicznych. Zwykle są one weryfikowane jedynie pod kątem właściwości strukturalnych. Właściwości behawioralne mają jednak również duże znaczenie. Mogą one być sprawdzane przy wykorzystaniu techniki weryfikacji modelowej [3]. Weryfikacja modelowa pozwala na wykrycie błędów wynikających z nieprawidłowej interpretacji specyfikacji [4]. Jest jedną z formalnych metod weryfikacji, oprócz innych jak np. automatycznego dowodzenia twierdzeń (ang. *theorem proving*) [5, 6], i jest obecnie wykorzystywana w przemyśle zarówno przy tworzeniu sprzętu [7], jak również oprogramowania [8].

W prezentowanym w artykule nowatorskim podejściu model logiczny utworzony na podstawie interpretowanej sieci Petriego sterowania przedstawiony jest na poziomie RTL (ang. *Register Transfer Level*). Tak przygotowany model logiczny nadaje się zarówno do formalnej weryfikacji, jak i do syntezy jako rekonfigurowalny sterownik logiczny (ang. *Reconfigurable Logic Controller*) lub PLC (ang. *Programmable Logic Controller*).

Artykuł podzielony jest następująco. Rozdział drugi przedstawia istniejący stan wiedzy dotyczący metod specyfikacji sterowników logicznych, sieci Petriego, interpretowanych sieci Petriego sterowania, logiki temporalnej jako matematycznego aparatu oraz techniki weryfikacji modelowej. Rozdział trzeci prezentuje nowatorską metodę tworzenia modelu logicznego na podstawie istniejącej interpretowanej sieci Petriego sterowania. Rozdział czwarty omawia weryfikację tak przygotowanego modelu logicznego w środowisku NuSMV. Rozdział piąty podsumowuje artykuł i przedstawia kierunki dalszych badań.

2. Istniejący stan wiedzy

Specyfikacja sterownika logicznego jest pierwszym etapem procesu projektowania, powinna więc dokładnie opisywać działanie tworzonego właśnie systemu. Ważne jest, aby określone zostały wymagania, jakie projektowany system ma spełniać. Przygotowanie dobrej specyfikacji, uwzględniającej również wpływ otoczenia, jest trudnym zadaniem [7]. Specyfikacja może być formalnie zapisana w różnych postaciach [9], m.in. przy wykorzystaniu sieci Petriego, czy też diagramów czynności języka UML 2.x. Diagramy czynności mogą zostać przekształcone do sieci Petriego, a następnie formalnie zweryfikowane w celu sprawdzenia spójności pomiędzy opisem modelu, a wymaganiami dotyczącymi jego zachowania [10].

2.1. Sieci Petriego

Sieci Petriego [9, 11, 12, 13] 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. Dostępne są narzędzia pozwalające na automatyczne generowaniu kodu na podstawie przygotowanej sieci Petriego [14]. Graficzna reprezentacja sieci Petriego pozwala na określanie takiego zachowania jak równoległość i współbieżność, wybór, synchronizacja czy też współdzielenie zasobów [13].

Sieć Petriego może być formalnie zapisana [11] jako uporządkowana trójka $PN = (P, T, F)$, gdzie:

(a) P oznacza miejsca sieci
(b) T oznacza tranzycje sieci
(c) F oznacza połączenia miejsca z tranzycją ($P \rightarrow T$) lub tranzycji z miejscem ($T \rightarrow P$)

2.2. Interpretowane sieci Petriego sterowania

Interpretowane sieci Petriego sterowania [15] modelują zachowanie współbieżnych sterowników logicznych uwzględniając właściwości kontrolowanych obiektów. Warunki tranzycji powiązane są z sygnałami wejściowymi, a miejsca powiązane są z sygnałami wyjściowymi sterownika.

Interpretowana sieć Petriego może być formalnie przedstawiona [11] jako uporządkowana szóstka $PN_{IO} = (PN, X, Y, \rho, \lambda, \gamma)$, gdzie:
(a) PN jest żywą i bezpieczną siecią Petriego
(b) X jest zbiorem sygnałów wejściowych
(c) Y jest zbiorem sygnałów wyjściowych
(d) $\rho: T \rightarrow 2^X$ jest funkcją, która każdej tranzycji przyporządkowuje jednoznacznie podzbiór sygnałów wejściowych $X(T)$; 2^X oznacza zbiór wszystkich możliwych podzbiorów X
(e) $\lambda: M \rightarrow Y$ jest funkcją wyjść typu Moore'a, która każdemu znakowaniu sieci M przyporządkowuje jednoznacznie pewien podzbiór sygnałów wyjściowych $Y(M)$
(f) $\gamma: (M \times X) \rightarrow Y$ jest funkcją wyjść typu Mealy'ego, która znakowaniu sieci M oraz stanowi wejść X przyporządkowuje jednoznacznie podzbiór stanów wyjść Y

2.3. Logika temporalna

Logika temporalna [16, 17, 18] wywodzi się z logiki modalnej. Obecnie wykorzystywana jest ona zarówno do specyfikacji programów, jak i ich późniejszej weryfikacji, syntezy, czy też przy programowaniu logicznym.

Klasyczną logiką temporalną jest liniowa logika temporalna LTL (ang. *Linear Time Logic*). Opisuje ona zależności w systemie przedstawiając sekwencję stanów. Logiką temporalną z podziałem czasu jest logika rozgałęziona CTL (ang. *Computation Tree Logic*). Czas przedstawiany jest tutaj jako drzewo rozszerzające się w przyszłość, gdzie głównym korzeniem jest aktualna chwila. Charakterystyczne dla logiki CTL są kwantyfikatory ścieżkowe oraz stanowe (ang. *path / state quantifiers*). Logika rozgałęziona CTL jest częściej wykorzystywana w przemyśle, niż logika liniowa LTL [19], a także nadaje się lepiej do weryfikacji niedeterministycznych programów [20].

2.4. Weryfikacja modelowa interpretowanych sieci Petriego sterowania

Technika weryfikacji modelowej (ang. *model checking*) [3, 21] [22] pozwala na automatyczną weryfikację behawioralnej specyfikacji systemu przy wykorzystaniu narzędzi wnioskowania komputerowego. Danymi wejściowymi do narzędzia weryfikującego (w pracy wykorzystywane jest narzędzie NuSMV w aktualnej wersji 2.5.2 [23]) jest opis modelu oraz lista wymagań stawianych projektowanemu systemowi. Wymagania zdefiniowane są przy pomocy logiki temporalnej. Należy mieć na uwadze fakt, że tylko wyspecyfikowane właściwości zostaną sprawdzone. Narzędzie weryfikujące zwraca odpowiedź czy dany model spełnia stawiane mu wymagania, a w przypadku gdy tak nie jest – dodatkowo wygenerowany kontrprzykład.

Intepretowana sieć Petriego sterowania może być wykorzystana do przygotowania modelu logicznego. W prezentowanym w artykule nowatorskim rozwiązaniu opis logiczny przedstawiony jest na poziomie RTL. Jest to forma dogodna zarówno do formalnej weryfikacji, jak i syntezy logicznej.

Technika weryfikacji modelowej może być wykorzystana do weryfikacji całego systemu, bądź też jego części. Częściowa weryfikacja jest szczególnie przydatna w przypadku dużych systemów, gdzie proces projektowania jest złożony i może być wykonany w kilku etapach.

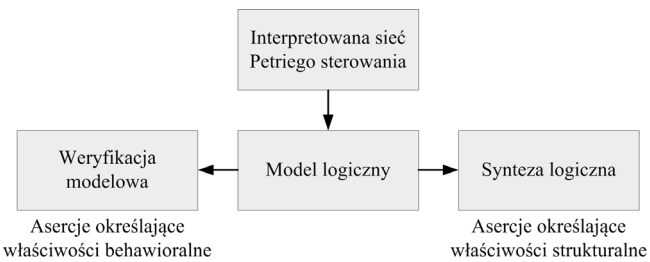
W literaturze podjęte zostały próby weryfikacji sieci Petriego. Jednakże, nie skupiają się one na interpretowanych sieciach Petriego sterowania przedstawionych na poziomie RTL. Podejmowane były również próby weryfikacji diagramów UML, jak np. w [24].

W pracy [25] prezentowane są czasowe sieci Petriego (ang. *Timed Petri Nets*) z czasami realizacji przypisanymi do tranzycji lub miejsc. W pracy [26] omówione zostały sieci Petriego ze stemplami czasowymi dla systemów osadzonych. Tokeny przechowują tam wartość oraz stempel czasowy, co różni je od podejścia w klasycznych sieciach Petriego. Ponadto, każda tranzycja interpretowana jest jako osobny proces, który zmienia markowanie miejsc. W pracy [27] rozważane są synchroniczne interpretowane sieci Petriego. Autorzy podejmują próbę ich weryfikacji z wykorzystaniem języka PROMELA i narzędzia weryfikującego SPIN. W pierwszym podejściu tylko jeden sygnał wejściowy (warunek tranzycji) może być aktywny w danym czasie, co utrudnia rzeczywistą analizę działania systemu. W drugim podejściu autorzy skupiają się raczej na właściwościach strukturalnych. Sztucznie wprowadzone zostały priorytety dla tranzycji, co deformuje funkcjonowanie opisywanego systemu.

3. Model logiczny interpretowanej sieci Petriego sterowania

Opracowany został model logiczny, który wykorzystywany jest zarówno do celów syntezy, jak i do weryfikacji modelowej. Stanowi on format pośredni opisujący zachowanie projektowanego sterownika logicznego.

Model logiczny otrzymany na podstawie interpretowanej sieci Petriego sterowania przedstawiony jest na poziomie RTL w taki sposób, że jest łatwo syntezowalny jako rekonfigurowalny sterownik logiczny lub PLC bez dodatkowych zmian.



Rys. 1. Model logiczny interpretowanej sieci Petriego sterowania
Fig. 1. Logical model of Control Interpreted Petri Net

Model logiczny zawiera definicje zmiennych wraz z wartościami jakie mogą przyjmować. Elementy sieci Petriego, które są bezpośrednio odwzorowane w modelu logicznym jako zmienne to:
(a) miejsca (P)
Każde miejsce traktowane jest jako osobna zmienna typu logicznego. Zmienna ta przyjmuje wartość *TRUE*, jeżeli dane miejsce zawiera token. W danej chwili wiele miejsc może jednocześnie zawierać token (procesy współbieżne), wiele zmiennych może więc mieć przypisaną wartość *TRUE*.
(b) sygnały wejściowe (X)
Każdy sygnał wejściowy traktowany jest jako osobna zmienna typu logicznego. Zmienna ta przyjmuje wartość *TRUE*, jeżeli dany sygnał jest aktywny. W danej chwili wiele sygnałów wejściowych może być aktywnych jednocześnie, wiele zmiennych może więc mieć przypisaną wartość *TRUE*.
(c) sygnały wyjściowe (Y)
Każdy sygnał wyjściowy traktowany jest jako osobna zmienna typu logicznego. Zmienna ta przyjmuje wartość *TRUE*, jeżeli dany sygnał jest aktywny. W danej chwili wiele sygnałów wyjściowych może być aktywnych jednocześnie, wiele zmiennych może więc mieć przypisaną wartość *TRUE*.

Alternatywnym podejściem jest wykorzystanie logiki wielowartościowej do reprezentacji zmiennych i wartości, jakie mogą one przyjmować. Zmienna reprezentująca miejsca sieci Petriego mogłaby wtedy przyjmować jedną z wartości ze zbioru stanów globalnych. Zmienne reprezentujące sygnały wejściowe oraz wyjściowe należałoby jednak nadal traktować jako osobne zmienne przyjmujące wartości *TRUE / FALSE*. Wiele sygnałów wejściowych lub wyjściowych może być bowiem aktywne jednocześnie. W przypadku małej liczby takich sygnałów można by się pokusić na użycie logiki wielowartościowej, tak jak w przypadku definicji miejsc. Zmienna reprezentująca sygnały przyjmowałaby wtedy jedną z wartości ze zbioru sygnałów (sygnały występujące pojedynczo lub w grupach). Tak sporządzony model logiczny także nadaje się do weryfikacji modelowej, gdyż narzędzie NuSMV wspiera enumeracje. W prezentowanym rozwiązaniu wybrana została logika binarna, ponieważ lepiej nadaje się ona jednocześnie do logicznej syntezy oraz weryfikacji modelowej.

Model logiczny poza definicją zmiennych zawiera także zbiór reguł, które opisują zachowanie projektowanego systemu i określają zmiany wartości zmiennych z upływem czasu. Tranzycje T oraz funkcja $\rho: T \rightarrow 2^X$ interpretowane są jako wspomniane zbiory reguł.

Funkcja $\lambda: M \rightarrow Y$ interpretowana jest jako przypisania sygnałów wyjściowych do odpowiednich miejsc, zgodnie z zasadą:

dla każdego miejsca $p \in P$ przypisz
(jeżeli zdefiniowano) sygnały wyjściowe $y \in Y$

Model logiczny może zostać poddany syntezie jako rekonfigurowalny sterownik logiczny lub PLC. Najważniejsze stają się wtedy sygnały wyjściowe sterownika, które kontrolują zachowanie systemu i sterują procesami.

4. Weryfikacja modelu logicznego

Model logiczny utworzony na podstawie interpretowanej sieci Petriego sterowania przekształcany jest do formatu wejściowego narzędzia weryfikującego NuSMV. Inne formy specyfikacji sterowników logicznych także mogą być poddane weryfikacji modelowej, np. algorytmiczne maszyny stanów [28].

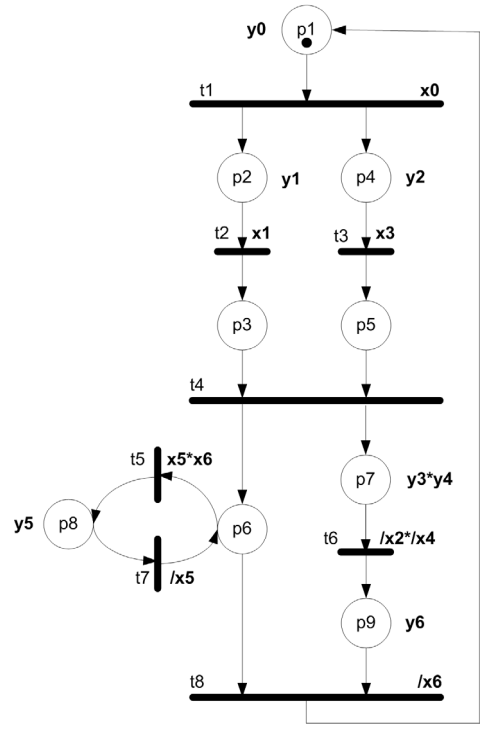
Model logiczny utworzony na podstawie sieci z rys. 2 zawiera definicje miejsc (*p1, ..., p9*), sygnałów wejściowych (*x0, ..., x6*) oraz sygnałów wyjściowych (*y0, ..., y6*). Formuły logiczne opisują zachowanie systemu oraz zmiany wartości zmiennych. Model logiczny przekształcany jest do formatu wejściowego narzędzia NuSMV [23] (wersja 2.5.2).

Przekształcenie modelu logicznego do formatu narzędzia NuSMV odbywa się według następujących reguł:

(a) Każde miejsce $p \in P$ jest zmienną typu logicznego (*Boolean*)
(b) Każdy sygnał wejściowy $x \in X$ jest zmienną typu logicznego (*Boolean*)
(c) Każdy sygnał wyjściowy $y \in Y$ jest zmienną typu logicznego (*Boolean*)
(d) Każde miejsce zmienia swoje znakowanie zgodnie z regułami określonymi przez tranzycje T oraz funkcję $\rho: T \rightarrow 2^X$; warunki zmian pomiędzy miejscami (przepływ tokenów) występują parami – w poprzednim stanie i w następnym stanie
(e) Każdy sygnał wejściowy zmienia losowo swoją wartość. Dostępne są jednak tylko wartości oczekiwane związane z konkretnymi miejscami sieci Petriego lub adekwatne do sytuacji
(f) Każdy sygnał wyjściowy zmienia swoją wartość zgodnie z regułami określonymi za pomocą funkcji $\lambda: M \rightarrow Y$

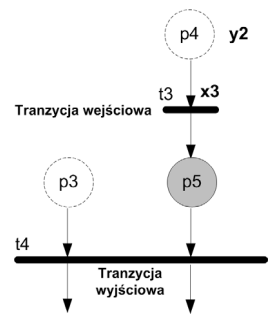
Opisana metoda nadaje się także do modelowania nieinterpretowanych sieci Petriego.

Proponowana oryginalna metoda transformacji obejmuje kilka podstawowych kroków. Interpretowana sieć Petriego sterowania jest dzielona na elementarne podsieci.



Rys. 2. Przykładowa interpretowana sieć Petriego sterowania [11]
Fig. 2. Sample Control Interpreted Petri Net [11]

Elementarna podsieć (rys. 3) zawiera jedno miejsce oraz wszystkie z nim związane tranzycje wejściowe i wyjściowe. Każda elementarna podsieć interpretowana jest jako osobny segment w opisie modelu w narzędziu NuSMV. Tak więc dla sieci o n miejscach tworzone są n segmenty dla następnych wartości zmiennych reprezentujących miejsca (9 miejsc w sieci z rys. 2 oraz 9 opowiadających im segmentów w opisie modelu w NuSMV).

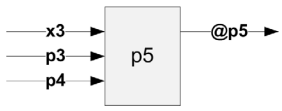


Rys. 3. Elementarna podsieć
Fig. 3. Elementary subnet

Elementarna podsieć z rys. 3 zawiera miejsce *p5*, jego tranzycję wejściową *t3* z warunkiem odpalenia *x3* (pośrednio z miejscem *p4*) oraz tranzycję wyjściową *t4* (pośrednio z miejscem *p3*). W opisie modelu w narzędziu NuSMV dana podsieć elementarna reprezentowana będzie jako jeden segment dla zmian znakowania miejsca *p5*.

Każda elementarna podsieć reprezentuje cyfrowy automat dwustanowy (maszynę stanów), który jest zazwyczaj realizowany w formie pojedynczej makrokomórki w rekonfigurowalnym układzie FPGA.

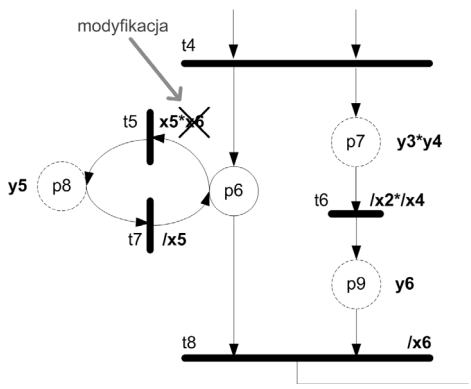
Dla elementarnej podsieci z rys. 3 utworzona zostanie pojedyncza makrokomórka przedstawiona na rys. 4. Sygnałami wejściowymi są tutaj miejsca *p3* i *p4* oraz sygnał *x3*. Sygnałem wyjściowym jest następny stan miejsca *p5*.



Rys. 4. Makrokomórka
Fig. 4. Macrocell

Właściwości, jakie projektowany system ma spełniać, definiowane są przy wykorzystaniu logiki LTL lub CTL. Właściwości strukturalne mogą zostać sprawdzone na poziomie sieci Petriego i nie wymagają użycia technika weryfikacji modelowej. Właściwości takie jak żywotność czy brak zakleszczeń mogą być zweryfikowane przez narzędzia służące do tworzenia sieci Petriego, jak np. WoPeD [29] czy PIPE2 [30]. Właściwości behawioralne nie są wspierane przez tego typu narzędzia, lecz są także istotne z punktu widzenia działania system. Pomagają one wykryć nawet drobne błędy na wczesnym etapie tworzenia systemu. Właściwości behawioralne określają wymagania dotyczące bezpieczeństwa (sytuacje, które nie mogą mieć miejsca) oraz wymagania dotyczące żywotności (sytuacje, które muszą się zdarzyć).

Przedstawiona wcześniej metoda weryfikacji modelu logicznego pozwala na wykrycie błędów wynikających także z nieprawidłowego zaprojektowania interpretowanej sieci Petriego sterowania. Wprowadźmy drobną modyfikację do sieci Petriego z rys. 2. Modyfikacja ta zmienia warunki odpalenia tranzycji *t5* (rys. 5). Miejsce *p6* posiada wtedy dwie tranzycje wyjściowe: tranzycję *t5* z warunkiem odpalenia *x5* (jeżeli miejsce *p6* zawiera token) oraz tranzycję *t8* z warunkiem odpalenia */x6* (jeżeli miejsca *p6* i *p9* zawierają token). Taka drobna modyfikacja zmienia zasadniczo działanie sieci. Można postawić sobie pytanie, co dzieje się w sytuacji, gdy miejsca *p6* i *p9* zawierają token, sygnał wejściowy *x5* jest aktywny, a sygnał wyjściowy *x6* jest nieaktywny. Przy wykorzystaniu logiki temporalnej można sprawdzić, czy możliwe jest jednoczesne odpalenie tranzycji *t5* i *t8*, a w wyniku tego jednoczesne markowanie miejsc *p1* i *p8*. Zdefiniowana właściwość `AG !(p1&p8)` nie jest spełniona w opisanym modelu systemu. W trakcie przeprowadzania weryfikacji modelowej generowany jest kontrprzykład (rys. 6), który obrazuje niepożądane zachowanie systemu.



Rys. 5. Modyfikacja interpretowanej sieci Petriego sterowania z rys. 2
Fig. 5. Modification of Control Interpreted Petri Net of Fig. 2

Kontrprzykład (rys. 6) demonstruje ścieżkę zdarzeń, które doprowadzają do opisanej sytuacji. Przedstawione są kolejne stany, które umożliwiają przeanalizowanie kontrprzykładu i odnalezienie źródła błędu. W omawianym przypadku z początkowego znakowania sieci Petriego w miejscu *p1* (*State: 1.1*) przechodzimy przez kolejne znakowania doprowadzające nas do aktywnych miejsc *p6* i *p9* (*State: 1.8*). W tymże stanie (*State: 1.8*) sygnał wejściowy *x5* jest aktywny (co umożliwia odpalenie tranzycji *t5*), a sygnał wejściowy *x6* jest nieaktywny (co umożliwia odpalenie tranzycji *t8*). Jako rezultat obie tranzycje odpalane są jednocześnie, a w następnym stanie (*State: 1.9*) system przechodzi do znakowania sieci Petriego w miejscach *p1* i *p8*, co jest niewątpliwie sytuacją niepożądaną.

```
-- specification AG !(p1&p8)  is false
-- as demonstrated by the following execution
sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  p1 = TRUE
  p2 = FALSE
  ...                          -> State: 1.6 <-
  p9 = FALSE                     p3 = FALSE
  x0 = FALSE                     p5 = FALSE
  ...                            p6 = TRUE
  x6 = FALSE                     p7 = TRUE
  y0 = FALSE                     x2 = TRUE
  ...                            x4 = TRUE
  y6 = FALSE                     y1 = FALSE
-> State: 1.2 <-                 y2 = FALSE
  x0 = TRUE                    -> State: 1.7 <-
  y0 = TRUE                      x2 = FALSE
-> State: 1.3 <-                 x4 = FALSE
  p1 = FALSE                     y3 = TRUE
  p2 = TRUE                      y4 = TRUE
  p4 = TRUE                    -> State: 1.8 <-
  x0 = FALSE                     p7 = FALSE
-> State: 1.4 <-                 p9 = TRUE
  x1 = TRUE                      x5 = TRUE
  x3 = TRUE                    -> State: 1.9 <-
  y0 = FALSE                     p1 = TRUE
  y1 = TRUE                      p6 = FALSE
  y2 = TRUE                      p8 = TRUE
-> State: 1.5 <-                 p9 = FALSE
  p2 = FALSE                     x5 = FALSE
  p3 = TRUE                      y3 = FALSE
  p4 = FALSE                     y4 = FALSE
  p5 = TRUE                      y6 = TRUE
```

Rys. 6. Wygenerowany kontrprzykład
Fig. 6. Generated counterexample

5. Podsumowanie

Interpretowane sieci Petriego sterowania są powszechnie wykorzystywane w przemyśle. Jednakże, zwykle są weryfikowane tylko pod kątem właściwości strukturalnych. Dedykowane narzędzia do projektowania sieci Petriego wspierają analizę strukturalną. Właściwości behawioralne są najczęściej pomijane, mimo ich dużego znaczenia praktycznego. Weryfikacja zachowania projektowanego systemu pozwala na wczesne wykrycie ewentualnych błędów powstałych na etapie specyfikacji systemu.

Artykuł prezentuje nowatorskie podejście do weryfikacji modelowej interpretowanych sieci Petriego sterowania. Na podstawie istniejącej sieci Petriego tworzony jest model logiczny na poziomie RTL. Model taki nadaje się zarówno do weryfikacji modelowej pod kątem behawioralnych właściwości jakie projektowany system ma spełniać, jak również do syntezy w postaci rekonfigurowalnego sterownika logicznego lub PLC. Otrzymany program sterownika logicznego (jego implementacja) będzie zatem poprawny względem jego pierwotnej (formalnie zweryfikowanej) specyfikacji.

Weryfikacja modelowa jest wartościową techniką sprawdzania poprawności przygotowanej specyfikacji. Mimo iż nigdy nie potwierdzi, że system jest wolny od błędów, może wykazać, że pewne określone przez użytkownika właściwości są spełnione. Dodatkową zaletą jest również możliwość częściowej weryfikacji, co może być wykorzystane przy projektowaniu złożonych systemów.

Dalsze plany badań obejmują m.in. zagadnienia automatycznej transformacji interpretowanych sieci Petriego sterowania do opisu modelu w narzędziu weryfikującym NuSMV.

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
Lubuskie Warte zachodu
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Autor jest stypendystą 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] Avizienis A., Laprie J., Randell B.: Fundamental concepts of computer system dependability, IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments, Seoul, Korea, May 2001.

[2] Khalgui M., Hanisch H.-M.: Reconfiguration of industrial embedded control systems, Tenth International Conference on Application of Concurrency to System Design, ACSD 2010.

[3] Emerson E.A.: The beginning of model checking: a personal perspective, Lecture Notes in Computer Science, 25 Years of Model Checking: History, Achievements, Perspectives, 2008, str. 27 – 45.

[4] Kropf T.: Introduction to Formal Hardware Verification, ISBN 3-540-65445-3, Springer-Verlag Berlin Heidelberg New York, 1999.

[5] Clarke E.M., Wing J.M. et al.: Formal methods: state of the art and future directions, ACM Computing Surveys, Vol. 28, No. 4, December 1996.

[6] Kern C., Greenstreet M.R.: Formal Verification in Hardware Design: A Survey, ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 4, Issue 2, 1999, str. 123 – 193.

[7] Fix L.: Fifteen years of formal property verification in Intel, Lecture Notes in Computer Science (LNCS) 2008, Volume 5000/2008, O. Grumberg, H. Veith (Eds.), str. 139 – 144, 2008.

[8] Holzmann G.J., Joshi R., Groce A.: Model driven code checking, Automated Software Engineering, Vol. 15, Issue 3-4, December 2008, str. 283 – 297.

[9] Gomes L., Barros J.P., Costa A.: Modeling formalisms for embedded system design, Embedded Systems Handbook, Taylor & Francis Group, LLC, 2006.

[10] 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, str. 607 – 612.

[11] Adamski M., Chodań M.: Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC, Wydawnictwo Politechniki Zielonogórskiej 2000.

[12] Adamski M., Karatkevich A., Węgrzyn M. (ed.): Design of embedded control systems, Springer 2005.

[13] David R., Alla H.: Petri Nets & Grafcet. Tools for modelling discrete event systems, Prentice Hall 1992.

[14] Frey G., Litz L: Verification and validation of control algorithms by coupling of Interpreted Petri Nets, Proceedings of the IEEE SMC'98, 1998, Vol. 1, str. 7–12.

[15] Adamski M.: A rigorous design methodology for reprogrammable logic controllers, The International Workshop on Discrete-Event System Design, DESDes'01, June 2001, Przytok.

[16] Ben-Ari M.: Logika matematyczna w informatyce. Klasyka informatyki. Wydawnictwa Naukowo-Techniczne, Warszawa 2005.

[17] Huth M., Ryan M.: Logic in Computer Science. Modelling and Reasoning about Systems, Cambridge University Press 2004.

[18] Klimek R.: Wprowadzenie do logiki temporalnej, AGH Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Kraków 1999.

[19] Rice M.V., Vardi M.Y.: Branching vs. Linear Time: Final Showdown, Proceedings of the 2001 Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001, LNCS Volume 2031, Springer-Verlag, str. 1 – 22.

[20] Lamport L.: "Sometime" is sometimes "not never", On the Temporal Logic of Programs, Proceedings of the Seventh ACM Symposium on Principles of Programming Languages, ACM SIGACT-SIGPLAN 1980, str. 174 – 185.

[21] Clarke E.M., Emerson E.A., Sistla A.P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications, ACM Transactions on Programming Languages and Systems, Vol. 8, No. 2, April 1986, str. 244 – 263.

[22] Clarke E.M., et al.: Automatic verification of sequential circuit designs, Phil. Trans. R. Soc. Lond. A (1992) 339, str. 105 – 120.

[23] Cavada R. et al.:, NuSMV 2.5 user manual, dostępne na http://nusmv.fbk.eu/

[24] Niewiadomski A., Penczek W., Szreter M.: Towards checking parametric reachability for UML state machines, Novosibirsk University 2009, Proc of. 7th International Andrei Ershov Memorial Conference Perspectives of System Informatics, str. 229-240.

[25] Penczek W., Półrola A.: Advances in verification of Time Petri Nets and timed automata, Springer 2006.

[26] Cortés L. A., Eles P., Peng Z.: Formal coverification of embedded systems using model checking, Proceedings of the 26th EUROMICRO Conference (EUROMICRO'00), 2000 IEEE.

[27] Ribeiro Ó. R., Fernades J.M.: Translating synchronous Petri nets into PROMELA for verifying behavioural properties, IEEE 2007.

[28] Grobelna I.: Formalna analiza interpretowanych algorytmicznych maszyn stanów ASM z wykorzystaniem narzędzia model checker, Metody Informatyki Stosowanej, nr 3/2008 Tom 16, str. 107 – 124.

[29] WoPeD homepage: www.woped.org

[30] PIPE homepage: http://pipe2.sourceforge.net/

otrzymano / received: 13.03.2011
przyjęto do druku / accepted: 04.05.2011 *artykuł recenzowany*

INFORMACJE

Nowy dział „Niepewność wyników pomiarów" na stronie internetowej Wydawnictwa PAK

Uprzejmie informuję, że na stronie internetowej Wydawnictwa PAK (WWW.pak.info.pl) został utworzony dział „Niepewność wyników pomiarów". Na p.o. redaktora działu został powołany dr inż. Paweł Fotowicz.

Dr P. Fotowicz jest ekspertem w zakresie problematyki niepewności, autorem szeregu wartościowych publikacji w czasopismach krajowych i zagranicznych. Prezentował swoje prace na licznych konferencjach i warsztatach szkoleniowych.

W dziale „ Niepewność wyników pomiarów", obok dostępu do aktualnych wybranych opracowań dotyczących niepewności jest możliwość zadawania „Pytań do eksperta". Pytania powinny być konkretne i szczegółowo sprecyzowane.

Pytania i odpowiedzi o istotnym znaczeniu dla szerszego grona metrologów będą archiwizowane i dostępne dla użytkowników strony internetowej Wydawnictwa PAK.

Zapraszam do odwiedzania działu „Niepewność wyników pomiarów" i do udziału w jego rozwoju.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK