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

NOWOCZESNA IMPLEMENTACJA HEURYSTYCZNEGO ALGORYTMU KODOWANIA STRUKTURALNEGO STANÓW LOKALNYCH W AUTOMATACH WSPÓŁBIEŻNYCH

Piotr Bubacz, Bartosz Mstowski, Marian Adamski

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

e-mail: {M.Adamski, P.Bubacz}@iie.uz.zgora.pl, B.Mstowski@adbglobal.com

STRESZCZENIE

W artykule została przedstawiona nowoczesna implementacja heurystycznego algorytmu kodowania strukturalnego stanów lokalnych w automatach współbieżnych. W ramach pracy przedstawiono środowisko .NET oraz możliwość jego wykorzystania w implementacji algorytmów analizy i kodowania sieci Petriego. Wynikiem pracy jest zintegrowane środowisko umożliwiające projektowanie, analizę i kodowanie sieci Petriego – „Petri Net Workshop".

1. WPROWADZENIE

Nowoczesne technologie wspierają pracę inżynierską i naukową. Jednym z najważniejszych elementów nowoczesnego programowania jest wielokrotne wykorzystanie modułów oraz łatwość rozwijania aplikacji.

Możliwości te daje programowanie zorientowane obiektowo, gdzie klasy są łatwo wykorzystywane w innych aplikacjach. Jednak dotychczas stosowane to było tylko w jednym języku programowania. Programiści musieli zdecydować się na jeden język i najczęściej na jedno środowisko programistyczne. Podejście to zmieniła firma Microsoft, prezentując technologię .NET. Pozwala ona nie tylko wykorzystanie tych samych klas w wielu aplikacjach, ale również łączenie ze sobą wielu języków programowania, tak, że każdy programista może tworzyć elementy w dowolnym języku i środowisku. Szczególnie jest to ważne w pracy naukowej. Każda z osób prowadzących badania rozwija swoją część projektu w dowolnym języku, a dzięki właściwościom środowiska .NET umożliwia to ich wzajemne współdziałanie, skoordynowane w jednej aplikacji.

Wykorzystując zoptymalizowane struktury danych oraz możliwości platformy .NET przeprowadzana jest łatwa, a co najważniejsze prosta do rozszerzania, implementacja algorytmów projektowania, analizy i kodowania sieci Petriego.

2. HEURYSTYCZNY ALGORYTMU KODOWANIA STRUKTURALNEGO STANÓW LOKALNYCH W AUTOMATACH WSPÓŁBIEŻNYCH

Heurystyczny algorytmu kodowania strukturalnego stanów lokalnych w automatach współbieżnych został zaproponowany w [1]. Jego zastosowanie i modyfikacje zostały przedstawione w pracach [2,3,4,6].

Algorytm strukturalnego kodowania miejsc sieci składa się z dwóch głównych etapów. W etapie pierwszym następuje kodowanie makromiejsc (makrostanów lokalnych), w kolejnym etapie kodowane są miejsca wewnątrz tych makromiejsc. Makromiejsce nierównoległe stopniowo rozróżnia się wprowadzając zmienne kodujące, przyjmujące wartości 0 w makromiejscu, wartość 1 w makromiejscach do niego przyległych i wartość – w makromiejscach równoległych.

Minimalizacja liczby zmiennych kodujących związana jest z kolejnym wyborem wierzchołków grafu współbieżności o najwyższym stopniu incydencji. Zakończenie kodowania makromiejsc następuje, gdy wszystkie makromiejsca rozróżniane są co najmniej jedną zmienna kodującą. Łatwo stwierdzić, że superpozycja kodów równocześnie oznakowanych makromiejsc tworzy wtedy unikalny kod wierzchołka grafu znakowań osiągalnych makrosieci. Należy zwrócić uwagę na fakt, że makromiejsce w proponowanym sposobie kodowania reprezentuje dowolny fragment sieci otrzymanej na drodze jej hierarchicznego składania (agregacji) lub rozkładania (deagregacji).

Kodowanie miejsc w ramach makromiejsca odbywa się w sposób analogiczny do kodowania makromiejsc, jeżeli to makromiejsce reprezentuje podsieć Petriego z równoległością. W przypadku makromiejsc typu automatowego wykorzystuje się znane metody kodowania teorii automatów.

Algorytm kodowania globalnego makromiejsc:

1. Dokonać dekompozycji sieci do postaci makrosieci reprezentującej analizowaną sieć Petriego.
2. Wyznaczyć graf znakowań osiągalnych makrosieci i na jego podstawie określić relacje:
   a) współbieżności makromiejsc R,
   b) niewspółbieżności (następstw) makromiejsc N.

   Relacje należy przedstawić w postaci grafów niezorientowanych lub odpowiadających im list następstw wierzchołków. Graf N jest dopełnieniem grafu R do grafu pełnego.
3. Przyjąć $i=1$ oraz $Ni=N$
4. Dopóki graf Ni nie jest grafem pustym (macierz niewspółbieżności nie zawiera samych zer), wykonać:

4.1. Wyznaczyć stopień incydencji dla wszystkich wierzchołków grafu Ni. Wybrać wierzchołek o najwyższym stopniu incydencji. Wierzchołki reprezentujące makromiejsca równoległe, przyległe do tych samych wierzchołków można potraktować jako jedno makromiejsce o stopniu incydencji równej sumie stopni poszczególnych makromiejsc.

4.2. Wybrać wierzchołek najwyższym stopniu incydencji.

4.3. Wybranemu makromiejscu *mp* przypisać kod *Q(i)=0*, dla każdego z pozostałych miejsc wykonać

4.3.1. Jeśli makromiejsce jest współbieżne do danego i jednocześnie niewspółbieżne do tego samego zbioru miejsc, co *mp*, przypisać kod *Q(i) = 0*. Wykreślić ten wierzchołek z grafu tzn. wyzerować odpowiedni wiersz i kolumnę w macierzy *N(i)* i zaznaczyć jako usunięte.

4.3.2. W przeciwnym wypadku jeśli makromiejsce jest niewspółbieżne z *mp* przypisać *Q(i) = 1*.

4.3.3. Jeśli makromiejsce zostało usunięte z *N(i)* i jest niewspółbieżne z *mp* to przypisać kod *Q(i)* = −. Oznacza to, że zmienna może być wykorzystana w kodowaniu miejsc należących do makromiejsca.

4.3.4. W przeciwnym wypadku makromiejsce współbieżne z wybranym, które nie zostało usunięte otrzymuje kod *Q(i)* = *. Oznacza to, że zmienna ta nie może być już wykorzystana do kodowania tego makromiejsca ani żadnego z miejsc do niego należących

4.3.5. Usunąć mp z macierzy *N(i)*.

5. Dokonać korekcji kodów częściowych. Dla każdego makromiejsca *mp* wykonać:

5.1. Dla każdej zmiennej *Q(i)* kodu *mp*, jeśli *Q(i)* = −, wykonać:

5.1.1. Znaleźć makromiejsce *mp'* współbieżne do *mp*.

5.1.2. Jeśli zmienna *Q(i)* kodu *mp'* jest różna od – oraz różna od *, zmienić wartość zmiennej *Q(i)* kodu *mp* na *.

5.1.3. Powtórzyć krok 5.1.1. aż do sprawdzenia wszystkich makromiejsc równoległych do *mp*.

6. Zakodować miejsca w ramach poszczególnych makromiejsc (kodowanie lokalne)

Szczegółowy opis algorytmu i przykłady kodowania można znaleźć w [1,6].

3. WPROWADZENIE DO ŚRODOWISKA .NET

Technologia .NET jest nowatorską technologią umożliwiającą tworzenie rozwiązań niezależnych od systemu operacyjnego i platformy sprzętowej, m.in. dla komputerów stacjonarnych (np. PC), mobilnych (np. Pocket PC), usług WWW (np. ASP.NET), usług sieciowych, rozwiązań bazodanowych (ADO.NET) oraz wielu innych. Platforma .NET pozwala na tworzenie i użytkowanie bezpiecznych i stabilnych aplikacji - zarówno skupionych jak i rozproszonych.

Sercem platformy .NET jest Common Language Runtime (w skrócie CLR), czyli wspólne dla różnych języków środowisko wykonywania programów. Kod programu w platformie .NET nie jest kompilowany bezpośrednio do postaci kodu wykonywalnego, lecz do postaci języka Microsoft Intermediate Language (w skrócie MSIL), czyli pośredniego między kodem programu a instrukcjami procesora. Do głównych zadań CLR należą: przydzielanie pamięci obiektom oraz zarządzanie tą pamięcią przy pomocy bardzo efektywnych technik garbage collection, sprawdzanie poprawności typów oraz dbanie o bezpieczeństwo.

Główną cechą, która wyróżnia platformę .NET wśród innych środowisk programistycznych, jest wsparcie wielu języków programowania. Umożliwia to grupom programistów o różnych umiejętnościach tworzenie komponentów oprogramowania w wybranych językach, a następnie połączenie opracowanych komponentów w dobrze współpracującą całość. Połączenie między językami realizowane jest nawet na poziomie klasy. Programiści mogą dziedziczyć po klasie zdefiniowanej w innym języku.

Wraz z wprowadzeniem CLR programiści uzyskali możliwość wyboru spośród szerokiej palety języków, co pozwala im programować w języku najbardziej odpowiadającym ich umiejętnościom i najlepiej nadającym się do wykonania otrzymanego zadania. Współpraca fragmentów kodu napisanych w różnych językach możliwa jest ze względu na to, że każdy język zgodny z platformą .NET Framework kompilowany jest do kodu MSIL, uruchamianego następnie w środowisku CLR.

Model programowania na platformie .NET jest bardzo zwarty i uproszczony. Wszelkie usługi systemowe udostępniane są w postaci wspólnego modelu zorientowanego obiektowo. W ten sposób ukrywane są szczegóły i koncepcje użyte w bibliotekach systemu. Znajomość tych interfejsów jest wymagana tylko wtedy, gdy programista świadomie chce użyć funkcjonalności systemu nie zdefiniowanej w kodzie .NET.

Kolejną cechą technologii .NET, która miała wpływ na jej wybór jest przenoszalność pomiędzy platformami. Jak wspomniano wcześniej kompilacja kodu .NET do języka IL pozwala w momencie uruchomienia przełożyć ten kod na instrukcje niemalże dowolnego procesora 32 i 64-bitowego. Tym zadaniem zajmuje się specjalny Just-In-Time (JIT)

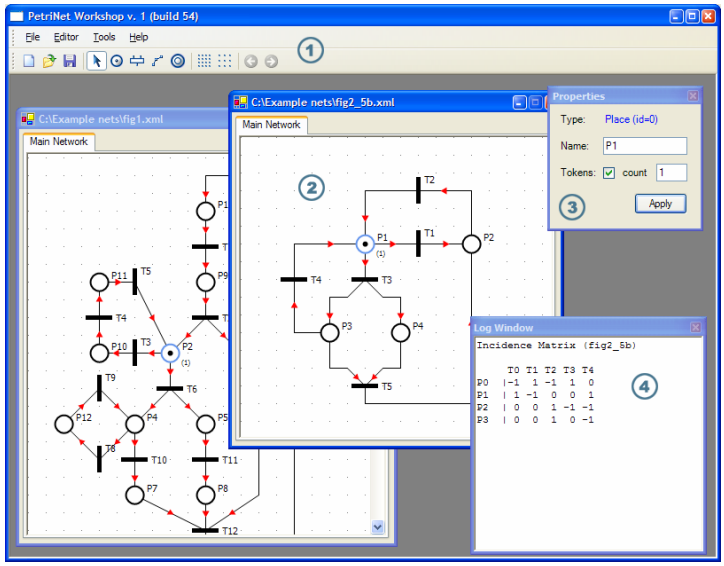
kompilator w CLR, który kompiluje kod pośredni na instrukcje natywne danej platformy. Kompilacja może mieć miejsce w trakcie wykonywania aplikacji lub przed jej uruchomieniem. Dzięki czemu programy napisane w .NET osiągają zbliżoną wydajność do programów napisanych w C++.

4. PROGRAM „*PETRI NET WORKSHOP*”

„Petri Net Workshop” jest zintegrowanym środowiskiem projektowania oraz badania sieci Petriego zaproponowanym przez mgra inż. Bartosza Mstowskiego w [6]. Środowisko zawiera graficzny edytor do projektowania sieci Petriego wraz z narzędziami do analizy własności sieci i ich wykorzystania w różnych praktycznych algorytmach. Program napisany został w zarządzanym języku C++ (ang. Managed C++ Extentions), będącym rozszerzeniem języka C++ o możliwości nowej platformy .NET [5].

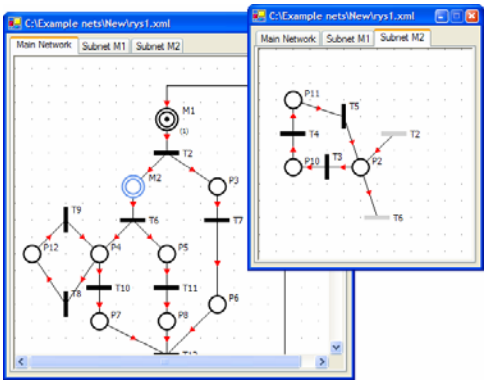
Program posiada również funkcje dydaktyczną, prezentuje kolejne kroki dekompozycji sieci. Możliwa jest dokładna obserwacja i analiza kolejnych kroków tworzenia makromiejsc i prezentowanie studentom algorytmu dekompozycji strukturalnej na przykładach.

Wygląd głównego okna programu przedstawiono na rys 1. Interfejs tworzą 4 elementy (oznaczone na rysunku kolejnymi numerami): menu główne wraz z paskiem narzędzi (1), okna dokumentów (2), okno własności (Properties) (3) oraz okno, w którym generowane są wyniki działania programu oraz różne komunikaty (Log Window) (4). Interfejs programu został napisany w języku angielskim, aby nie zawężać grona jego ewentualnych użytkowników.



Rys. 1. Wygląd graficznego interfejsu użytkownika programu „Petri Net Workshop”

Program jest aplikacją implementującą MDI (Multi-Document Iterface), co pozwala na równoczesną pracę nad różnymi sieciami przechowywanymi w oddzielnych dokumentach. Zasadniczo każdemu dokumentowi przypisana jest jedna sieć Petriego, która zawiera dowolną liczbę podsieci. Okno dokumentu ma zakładki, z których pierwsza zawiera makrosieć najwyższego poziomu hierarchii, a pozostałe zawierają podsieci należące do kolejnych makromiejsc. Na rys 2. przedstawiono dwa widoki przykładowego dokumentu, z siecią będącą w trakcie procesu dekompozycji.



Rys. 2. Okna dokumentu programu „Petri Net Workshop”

Jako format zapisu sieci został wybrany język PNML. Jest on niezależny od specyficznych cech różnych narzędzi i platform. Pozwala zapisać dowolny typ sieci Petriego, w szczególności zapewnia możliwość definiowania nowych typów, co jest ważne dla projektantów nowych systemów. PNML potrafi reprezentować każdy typ sieci wraz z jego specyficznymi właściwościami i rozszerzeniami. Dzięki czemu nie ma potrzeby wyciągać lub ignorować niektórych specyficznych informacji w czasie konwersji z modelu używanego przez określone narzędzie do formatu PNML.

5. NOWOCZESNA IMPLEMENTACJA

Wybór platformy .NET, w prezentowanej implementacji, został spowodowany zaletami CLR, jednak kluczowe znaczenie przy jej wyborze miały inne technologie dostarczane wraz z platformą .NET Framework:

- Managed C++ Extensions - będącą rozszerzeniem możliwości języka C++,
- Base Class Library - będącą biblioteką setek gotowych do użycia klas,
- Windows Forms - zbiór klas .NET opakowujących większość funkcji Windows API pozwalających szybko tworzyć aplikacje okienkowe,
- System.XML Parser – narzędzie, które umożliwiło implementację zapisu i odczytu sieci Petriego w formacie PNML.

Nowoczesność implementacji polega na wykorzystaniu programowania zorientowanego obiektowo w środowisku .NET. Środowisko to umożliwia wykorzystanie powstałych klas w innych językach wspieranych przez tą technologię. Możliwe zatem jest wykorzystanie zaprojektowanego środowiska, jak i zaimplementowanego algorytmu przez innych programistów.

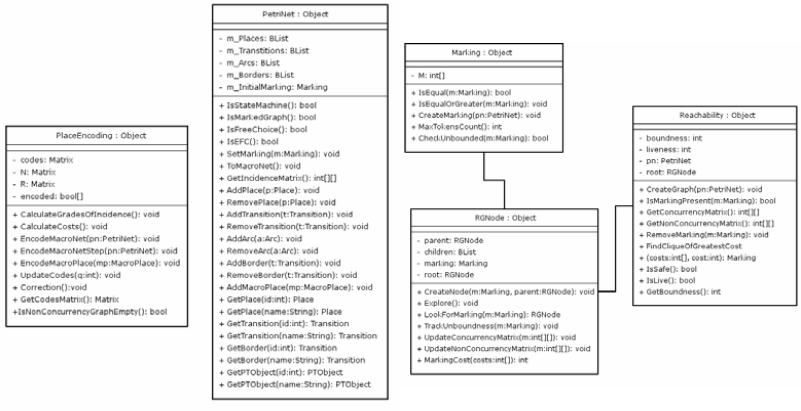
Dodatkowym atutem aplikacji jest wykorzystanie języka PNML jako formatu zapisu sieci. Format umożliwia wymianę zaprojektowanych sieci z innymi programami do analizy i symulacji sieci Petriego tj. CPN Tools i inne [7].

Struktura klas została szczegółowo przemyślana pod kątem najprostszej implementacji i późniejszego rozszerzania możliwości aplikacji. Jest ona elastyczna i otwarta. Zaprojektowane klasy umożliwiają graficzną reprezentację sieci, jak również są wykorzystywane do implementacji prezentowanego algorytmu.

Klasa *PetriNet* jest główną i najbardziej rozbudowaną klasą w programie (Rys. 3). Udostępnia ona wszelkie metody operowania na strukturze sieci: dodawanie usuwanie, pobieranie referencji do konkretnych jej elementów, a także metodę *ToMacroNet(),* która inicjuje proces dekompozycji sieci. Poza tym zawiera metody pozwalające na klasyfikację sieci, które mogą być pomocne w algorytmach, w których istotną rolę odgrywają własności sieci.

Kolejną kluczową klasą będącą wynikiem pracy autora jest *PlaceEncoding*, która zawiera zbiór funkcji związanych z kodowaniem sieci zdekomponowanej. Ponieważ głównymi strukturami, na których opiera się algorytm kodowania są macierze, wykorzystuje ona klasę *Matrix* do enkapsulacji danych macierzy i wszelkich potrzebnych na nich operacji (*rys 3*).

Klasy umożliwiające graficzną prezentacje sieci i klasy przechowujące struktury listowe z powodu ograniczonego miejsca nie zostały przedstawione w niniejszym referacie, więcej informacji znajduje się w [6].



Rys. 3. Diagram klas PetriNet, PlaceEncoding oraz klas tworzących strukturę drzewa znakowań

6. WNIOSKI I KIERUNKI DALSZYCH PRAC

Technologia .NET umożliwia znacznie efektywniejszą realizację algorytmu kodowania miejsc w porównaniu ze środkami programistycznymi dostępnymi w czasie opracowania jego pierwszej wersji [1].

Zaprezentowana aplikacja umożliwia łatwe projektowanie i kodowanie stanów lokalnych w automatach współbieżnych opisanych interpretowanymi sieci Petriego. Posiada również funkcje dydaktyczną umożliwia prezentacje kolejnych kroków dekompozycji sieci.

Struktura zaprojektowanych klas jest otwarta i elastyczna. Umożliwia to łatwe i szybkie tworzenie nowych modułów na bazie istniejącego kodu. Program stanowi bazę do rozwoju zintegrowanej aplikacji umożliwiającej analizę, kodowanie a w przyszłości syntezę sterowników cyfrowych opisanych interpretowanymi sieciami Petriego.

W ramach dalszych prac program zostanie rozbudowany o kolejne funkcje napisane w innych językach programowania tj. C i Java. W szczególności dołączone zostaną funkcje systemu PeNCAD. Dodatkowo, dzięki zastosowaniu języka PNML, istnieje możliwość wymiany zaprojektowanych sieci z innymi programami do analizy i symulacji sieci Petriego.

LITERATURA

[1] M. Adamski, *Heurystyczna metoda strukturalnego kodowania miejsc sieci Petriego*, Zeszyty Naukowe WSI, Nr 78, Zielona Góra, 1986, s. 113-125

[2] M. Adamski, J. Mirkowski, *Zbiór przykładów automatycznego kodowania miejsc sieci Petriego*, Wyższa Szkoła Inżynierska, Zielona Góra, 1988

[3] K. Biliński, *Application of Petri Nets in parallel controllers design, PhD. thesis*, University of Bristol, Bristol, 1996

[4] K. Biliński, M. Adamski, J. M. Saul, E. L. Dagless, *Petri-net-based Algorithms for Parallel-controller Synthesis*, IEE Proceedings - Computer Digital Techniques, Vol. 141, No. 6, Bristol, 1994, s. 405-412

[5] R. Grimes, *Programming with Managed Extensions for Microsoft Visual C++*, Microsoft Press, Redmond, 2003

[6] B. Mstowski, *Heurystyczny algorytm kodowania strukturalnego stanów lokalnych w automatach współbieżnych*, praca dyplomowa pod kierunkiem prof. dra hab. Mariana Adamskiego, Uniwersytet Zielonogórski, Zielona Góra, 2004

[7] *Petri Nets Tools Database*, http://www.daimi.au.dk/PetriNets/tools/quick.html