Monika WIŚNIEWSKA
UNIWERSYTET ZIELONOGÓRSKI,
ul. Licealna 9, 65-417 Zielona Góra

Automatyczny system wspomagający proces projektowania systemów dyskretnych z wykorzystaniem hipergrafów

Mgr inż. Monika WIŚNIEWSKA

Ukończyła studia na Wydziale Elektrycznym Uniwersytetu Zielonogórskiego, o specjalności Inżynieria Komputerowa. Obroniła pracę magisterską w 2003 r. W roku 2001 odbyła przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Od 2004 r. jest słuchaczem studiów doktoranckich, specjalność informatyka. Jej zainteresowania naukowe to dekompozycja systemów dyskretnych z wykorzystaniem hipergrafów.

e-mail: M.Wisniewska@weit.uz.zgora.pl

Streszczenie

W artykule zaprezentowany został autorski system wspomagający proces projektowania systemów dyskretnych z wykorzystaniem hipergrafów. Narzędzie Hippo składa się ze zbioru bibliotek, realizujących najważniejsze operacje z zakresu teorii grafów i hipergrafów (m. in. kolorowanie, pokrycie, dopełnienie, dualizm, itd.), które zostały zrealizowane pod kątem ich zastosowania w dekompozycji systemów dyskretnych. Głównym zadaniem systemu jest usprawnienie oraz automatyzacja procesu dekompozycji systemów dyskretnych stosowanych m.in. w projektowaniu zaawansowanych układów cyfrowych (redukcja rozmiaru pamięci, selekcja klas kompatybilności, minimalizacja funkcji logicznych, dekompozycja automatów cyfrowych). Opracowany system Hippo umożliwia przeprowadzenie automatycznego procesu dekompozycji z zastosowaniem różnych algorytmów (zarówno grafowych, jak i hipergrafowych), w efekcie pozwalając wybrać użytkownikowi najkorzystniejsze rozwiązanie.

Słowa kluczowe: graf, hipergraf, dekompozycja systemów dyskretnych, autorski system wspomagający proces dekompozycji systemów dyskretnych z wykorzystaniem hipergrafów (system Hippo).

CAD system for automatic decomposition of discrete systems based on hypergraphs

Abstract

In the paper a dedicated CAD system Hippo for automatic decomposition of discrete systems is presented. The tool consists of a set of libraries. Each library was designed as a separate module to solve the particular problem from the field of the graph and hypergraph theories (among others: vertex coloring, vertex covering, transversal computation, dualism, computation of the graph and hypergraph complement). The main task of the system is to improve the process of decomposition of discrete systems (for example: reduction of the microinstruction length, selection of the compatibility classes, decomposition of concurrent automata). The Hippo system consists of eight main modules:
- complement –calculation of graph/hypergraph complement;
- coloring – five methods of coloring of graph and hypergraph (four greedy and one backtracking);
- transversal – four methods of transversals computation (fast reduction algorithm, greedy, backtracking, mixed fast reduction and greedy);
- exact transversals – the calculation of the exact transversals is based on the Knuth DLX algorithm, the main advantage of such a solution is polynomial computational time in case of exact hypergraphs;
- dualism (only for hypergraphs) – calculates the dual hypergraph;
- converting graph to hypergraph;
- converting hypergraph to graph;
- conversion of the graph/hypergraph description to the TeX format.

In the paper particular libraries are described in detail. Moreover, the stand-alone application (Hippo) is shown. Finally, an example of automatic decomposition of the discrete system is presented. All steps and required operations are described.

Keywords: graph, hypergraph, decomposition of discrete system, CAD system Hippo for automatic decomposition of Petri Nets based on hypergraphs.

1. Wprowadzenie

W ostatnich latach można zauważyć coraz większy wpływ teorii grafów na rozwój techniki cyfrowej [1, 2]. Wiąże się to ze wzrostem rozmiaru układów logicznych, a co za tym idzie, pojawieniem się nowatorskich metod projektowania systemów cyfrowych [3]. O ile jeszcze kilkanaście lat temu projektant mógł w łatwy i szybki sposób opracować funkcjonalność swojego układu, o tyle dziś jest to niemożliwe bez zastosowania narzędzi wspomagających projektowanie opierających się przede wszystkim na algorytmach wykorzystujących teorię grafów nieskierowanych. Wiele obecnych metod dekompozycji systemów dyskretnych, stosowanych przez producentów wiodących firm przygotowujących narzędzia do syntezy układów cyfrowych bazuje na wykorzystaniu klasycznych grafów nieskierowanych [1]. Takie podejście wymusza stosowanie coraz bardziej skomplikowanych oraz modyfikację już istniejących algorytmów. Powodowane jest to ciągłym wzrostem rozmiaru dostępnych systemów cyfrowych, co się wiąże ze zmianą sposobu implementacji opracowanego modelu w fizycznym układzie logicznym. Rozwiązaniem problemu może być zastosowanie teorii hipergrafów [4]. Wykorzystanie hipergrafów do realizacji operacji stosowanych w technice cyfrowej, takich jak minimalizacja funkcji logicznych, zmniejszenie rozmiaru pamięci czy partycjonowanie systemu dyskretnego, wydaje się intuicyjne, bardziej przejrzyste i efektywniejsze niż w przypadku klasycznych grafów nieskierowanych [5, 6, 7].

W artykule zaprezentowano autorski program Hippo wspomagający proces dekompozycji systemów dyskretnych z wykorzystaniem hipergrafów. Narzędzie składa się ze zbioru bibliotek, w których zaimplementowano najważniejsze operacje z zakresu teorii grafów oraz hipergrafów. Celem opracowanego programu była automatyzacja procesów zachodzących podczas dekompozycji systemów dyskretnych.

2. Motywacja realizacji bibliotek oraz systemu Hippo

Pomimo bardzo szczegółowej analizy i wnikliwych internetowych poszukiwań, nie natrafiono na istniejący już system wspomagający proces obliczeń z wykorzystaniem teorii hipergrafów. Wprawdzie pojawiają się gotowe aplikacje (zazwyczaj komercyjne) wspomagające proces operacji grafowych, jednakże nie natrafiono na programy operujące na teorii hipergrafów. Dlatego też zdecydowano się na autorską realizację systemu, który w zamierzeniu miał służyć weryfikacji i automatyzacji metod dekompozycji systemów dyskretnych, zaproponowanych przez Autorkę w [5, 6, 7]. Ostatecznie projekt został znacznie rozbudowany, zaimplementowano w nim szereg dodatkowych algorytmów i metod (np. konwersje pomiędzy różnymi formatami, reprezentacja graficzna, możliwość eksportu grafiki wektorowej, itd.). W tym przypadku motywacją było praktyczne wykorzystanie systemu w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego (zarówno w dydaktyce, jak i w badaniach naukowych). Warto podkreślić, że program znalazł swoje zastosowanie podczas realizacji projektu pokazywanego podczas Festiwalu Nauki 2010, a także podczas Dnia Województwa Lubuskiego 2010 (projekt „Inteligentny dom widzi i słyszy”).

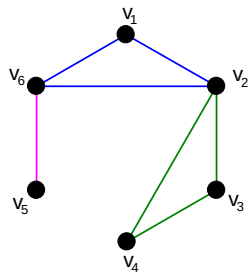
3. Dane wejściowe w systemie Hippo

System Hippo operuje na strukturach macierzowych. Standardowym wejściem jest opis grafu nieskierowanego lub hipergrafu przedstawiony w formie tekstowej. Dane wejściowe mogą zostać przekazane poprzez plik tekstowy lub bezpośrednio w środowisku graficznym. Struktura domyślnych danych wejściowych jest bardzo zbliżona do macierzy incydencji grafu/hipergrafu i ma następującą postać:

- Pierwszy wiersz definiuje liczbę wierzchołków grafu/hipergrafu.
- Drugi wiersz zawiera liczbę krawędzi grafu/hipergrafu.
- Kolejne wiersze zawierają wartości macierzy incydencji grafu/hipergrafu. Wartość 1 oznacza, że krawędź reprezentowana przez j − tą kolumnę zawiera wierzchołek, reprezentowany przez i − ty wiersz. Analogicznie, wartość 0 jest rozumiana jako brak przynależności wierzchołka do danej krawędzi.

Rys. 1. Dane wejściowe systemu Hippo
Fig. 1. Input data format of Hipppo

Rysunek 1 przedstawia opis przykładowego hipergrafu H_1 o m=6 wierzchołkach (V ={v_1,v_2,v_3,v_4,v_5,v_6}) oraz n=3 hiperkrawędziach (E={E_1,E_2,E_3}). W rozpatrywanym przykładzie poszczególne krawędzie zawierają następujące wierzchołki: E_1={v_1,v_2,v_6}, E_2={v_2,v_3,v_4}, E_3={v_5,v_6}. Graficzną reprezentację hipergrafu H_1 pokazano na rys. 2. Poszczególne kolory odzwierciedlają strukturę hipergrafu (pojedynczej hiperkrawędzi odpowiada jeden kolor).



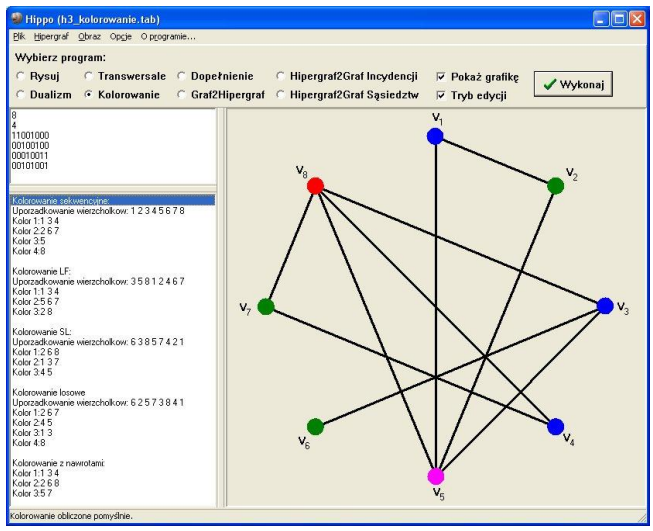
Rys. 2. Graficzna reprezentacja hipergrafu H_1 w systemie Hippo
Fig. 2. Graphic representation of the hypergraph H_1

Dodatkowo istnieje możliwość przedstawienia danych wejściowych jako macierz sąsiedztw (oczywiście ma to sens tylko w przypadku grafów). Należy jednak pamiętać, że ta reprezentacja może być stosowana jedynie w połączeniu z modułem *Graf2Hipergraf*, ponieważ pozostałe biblioteki są kompatybilne jedynie z domyślnym formatem danych.

4. Rezultat wykonania operacji w systemie Hippo

W zależności od zastosowanej biblioteki, system może zwrócić dane w różnych formatach. W przypadku modułów *Kolorowanie* oraz *Transwersale* wynikiem będzie tekstowy opis uzyskanych rezultatów (rys. 3). Pozostałe moduły zwrócą wynik w postaci macierzy incydencji (lub sąsiedztw).

Wynik jest ilustrowany graficznie, co znacznie zwiększa czytelność uzyskanych rezultatów. Istnieje także możliwość wyświetlenia wszystkich możliwych wyników kolorowania oraz transwersali poprzez wybór odpowiedniej metody (rys. 3).



Rys. 3. Przykładowy rezultat systemu Hippo
Fig. 3. Exemplary output data of Hippo

5. Struktura systemu Hippo

W niniejszym rozdziale omówiono podstawowe moduły (biblioteki) systemu *Hippo*. Opisano także sposób realizacji poszczególnych jednostek.

System *Hippo* zawiera osiem niezależnych modułów (bibliotek), przy czym każdy wykonywany jest jako niezależny proces. W skład *Hippo* wchodzą następujące biblioteki:

1. *Dopełnienie* - moduł wyznaczający dopełnienie dla zadanego grafu/hipergrafu. W przypadku grafów wyznaczenie dopełnienia sprowadza się do prostej operacji na macierzy sąsiedztw (zamiana wartości pól), natomiast w przypadku hipergrafów znalezienie dopełnienia hipergrafu jest problemem NP-trudnym [4, 8]. O ile sama analiza powiązań poszczególnych wierzchołków jest relatywnie nieskomplikowana, o tyle określenie zależności pomiędzy wierzchołkami w nowoutworzonym hipergrafie stanowi już problem NP-trudny, gdyż de facto sprowadza się do wyznaczenia klik w dopełnieniu [4]. Metoda ta została zaimplementowana w programie *Hippo*. Dodatkowo program oferuje alternatywny sposób obliczenia dopełnienia hipergrafu, poprzez wyznaczenie transwersal dokładnych (dla hipergrafów dokładnych).
2. *Kolorowanie* - moduł realizujący kolorowanie grafu/hipergrafu. Zaimplementowano sześć metod kolorowania grafów i hipergrafów, z czego cztery opierają się o algorytm zachłanny: bez uporządkowania, z uporządkowaniem typu LF (*Largest-First*), z uporządkowaniem SL (*Smallest-Last*) oraz uporządkowaniem losowym. Ponadto istnieje możliwość pokolorowania grafu lub hipergrafu z wykorzystaniem algorytmu z nawrotami [7, 9]. W tym przypadku program oblicza wszystkie możliwe kombinacje, wybierając rozwiązanie najkorzystniejsze (oczywiście kosztem złożoności obliczeniowej, w tym przypadku - wykładniczej).
3. *Transwersale* - moduł obliczający pokrycie wierzchołkowe (bazę wierzchołkową) hipergrafu [8]. Zaimplementowano cztery metody wyznaczania transwersali: szybki algorytm redukcji, algorytm zachłanny, algorytm z nawrotami, algorytm hybrydowy (połączenie szybkiego algorytmu redukcji oraz metody zachłannej) [5,6,8]. Dodatkowo istnieje możliwość wyznaczenia wszystkich transwersal dokładnych.
4. *Dualizm* – moduł wyznaczający hipergraf dualny do danego hipergrafu. Operacja ta polega na wyznaczeniu macierzy transponowanej do macierzy incydencji hipergrafu [4].
5. *Graf2Hipergraf* - moduł przeprowadzający konwersję grafu do hipergrafu. W praktyce funkcjonalność tej biblioteki sprowadza się do wyznaczenia wszystkich maksymalnych klik pełnych, a następnie zastąpienie ich hiperkrawędziami. Wejściem może

być macierz sąsiedztwa lub macierz incydencji grafu nieskierowanego.
6. *Hipergraf2Graf Incydencji* - moduł przeprowadzający konwersję hipergrafu do grafu. W praktyce jest to przekształcenie macierzy incydencji hipergrafu do macierzy incydencji grafu nieskierowanego.
7. *Hipergraf2Graf Sąsiedztw* - moduł przeprowadzający konwersję hipergrafu do grafu, której wynikiem jest graf zapisany w postaci macierzy sąsiedztw.
8. *Hipergraf2Tex* - moduł dodatkowy, przeprowadzający konwersję hipergrafu (lub grafu) opisanego za pomocą danych wejściowych do tabeli interpretowanej przez system *LaTeX*.

Poza wymienionymi modułami, *Hippo* posiada także opcje ułatwiające i usprawniające operacje na grafach i hipergrafach, np. możliwość zapisu obrazu wynikowego w postaci wektorowej, wyłączenie trybu graficznego (szczególnie przydatne w przypadku relatywnie dużych struktur, gdzie reprezentacja graficzna jest nieczytelna). Podstawowym zadaniem opracowanych bibliotek była automatyzacja procesu dekompozycji systemów dyskretnych. Dlatego też zaimplementowano najważniejsze algorytmy z zakresu kolorowania oraz pokrycia grafów i hipergrafów (badania przydatności i skuteczności opracowanych metod dekompozycji systemów dyskretnych). W kolejnym rozdziale zaprezentowano praktyczny sposób wykorzystania algorytmów i metod zaimplementowanych w pakiecie *Hippo*.

6. Zastosowanie systemu Hippo

W rozdziale zobrazowano możliwości zastosowania systemu *Hippo* w dekompozycji systemów dyskretnych na przykładzie redukcji pojemności pamięci. Operacja redukcji pojemności pamięci jest często spotykanym etapem w procesie projektowania systemów dyskretnych. Problem ten wiąże się z ograniczonym rozmiarem dedykowanych bloków pamięci oferowanych przez układy cyfrowe (np. matryce FPGA) [2, 3]. Dlatego też często stosowaną techniką jest redukcja rozmiaru mikroinstrukcji (a w efekcie całej pamięci).

Proces redukcji pojemności pamięci składa się z pięciu podstawowych etapów (szczegółowy opis można znaleźć w innych publikacjach autorki – [5, 6]):
1. Przedstawienie pamięci za pomocą hipergrafu (mikrooperacje są reprezentowane poprzez wierzchołki, mikroinstrukcje poprzez krawędzie).
2. Utworzenie zbioru C_C klas kompatybilności (zgodności) mikrooperacji oraz określenie wagi (kosztu) dla każdej klasy kompatybilności.
3. Utworzenie hipergrafu H^* dualnego do hipergrafu H.
4. Wyznaczenie najmniejszej transwersali $\tau(H^*)$ hipergrafu dualnego H^*.
5. Kodowanie klas kompatybilności realizujących minimalną transwersalę $\tau(H^*)$ hipergrafu H^* oraz wyznaczenie nowej zawartości pamięci z zakodowanymi klasami kompatybilności.

Wszystkie powyższe zadania mogą zostać w sposób automatyczny zrealizowane przez system *Hippo*. Wartość pamięci jest opisywana za pomocą macierzy incydencji hipergrafu. Utworzenie zbioru klas kompatybilności sprowadza się do wyznaczenia wszystkich zbiorów niezależnych. Krok ten wykonywany jest poprzez obliczenie dopełnienia hipergrafu (moduł *Dopełnienie* w programie *Hippo*).

Kolejny etap to utworzenie hipergrafu dualnego. W przypadku systemu *Hippo* wartość macierzy dla hipergrafu dualnego H^* jest obliczana jako macierz transponowana hipergrafu H (moduł *Dualizm*).

Najmniejsza transwersala (czyli krok 4 algorytmu) może zostać wyznaczona na kilka sposobów. Program *Hippo* oferuje cztery alternatywne sposoby wyszukiwania pokrycia wierzchołkowego (moduł *Transwersale*). Badania autorki wykazały, że w przypadku relatywnie niewielkich pamięci, sugerowana jest metoda dokładna (algorytm z nawrotami). Większe pamięci (rozmiar około 20 mikrooperacji i 50 mikroinstrukcji) użytkownik może zdekomponować poprzez zastosowanie algorytmu zachłannego. Metoda ta oferuje znalezienie rozwiązania w czasie quasi-liniowym, przy czym znalezione pokrycie może się nie okazać najefektywniejszym (taka ocena jest możliwa tylko i wyłącznie w przypadku algorytmu przeszukującego cały zbiór rozwiązań). Szczegółowe wyniki badań można znaleźć w innych publikacjach autorki, m.in. [5, 6, 7].

Ostatni etap to kodowanie klas kompatybilności oraz wyznaczenie zawartości nowej pamięci. Proces ten jest wykonywany automatycznie przez system *Hippo*, poprzez przekazanie odpowiednich parametrów, które są przekazane na etapie wyznaczania transwersal. W efekcie wynikiem programu jest określenie najmniejszej transwersali oraz kody poszczególnych klas kompatybilności (kodowanie nie ma znaczenia, stosowany jest naturalny kod binarny). Zestawienie poszczególnych kodów klas kompatybilności określa zawartość zredukowanej pamięci.

7. Podsumowanie

W referacie przedstawiono autorski program *Hippo*, wspomagający proces projektowania systemów dyskretnych. Narzędzie składa się ze zbioru bibliotek operujących na hipergrafach oraz grafach nieskierowanych. Zaimplementowano wszystkie podstawowe operacje badające te struktury (dopełnienie, kolorowanie, dualizm, pokrycie wierzchołkowe, transwersale). Opracowany program pozwala przeprowadzić automatyczną dekompozycję systemów dyskretnych (w swoich pracach oraz badaniach autorka skoncentrowała się głównie na redukcji pojemności pamięci oraz dekompozycji sieci Petriego).

Autorka jest stypendystką 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.

8. Literatura

[1] De Micheli G.: Synthesis and Optimization of Digital Circuits. McGrawHill, 1994.
[2] Łuba T.: Synteza układów cyfrowych. Praca zbiorowa pod redakcją prof. Tadeusza Łuby, Warszawa: WKŁ, 2003.
[3] Maxfield C.: The Design Warrior's Guide to FPGAs. Orlando, FL, USA: Academic Press, Inc., 2004.
[4] Berge C.: Graphs and Hypergraph. Amsterdam: North-Hols.r Mathematical Library, 1976.
[5] Wiśniewska M.: Redukcja rozmiaru mikroinstrukcji w projektowaniu sterowników mikroprogramowanych, PAK, Nr 8, s. 575-577, 2009.
[6] Wiśniewska M., Wiśniewski R., Adamski M., Halang W.: Application of hypergraphs in microcode lenght reduction of microprogrammed controllers, Second International Workshop on Nonlinear Dynamics and Sychronization - INDS '09. Klagenfurt, Austria, 2009, ss. 106-109.
[7] Wiśniewska M, Wiśniewski R., Adamski M.: Usage of hyper-graph theory in decomposition of concurrent automata. PAK,. 2007, nr 7, s. 66-68.
[8] Eiter T., Gottlob G.: Hypergraph transversal computation and related problems in logic and AI, LNCS, pp. 549—564, Springer, 2002.
[9] Aho A. V., Hopcroft J. E., Ullman J. D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, 1974.

otrzymano / received: 13.04.2011
przyjęto do druku / accepted: 06.06.2011 *artykuł recenzowany*