Implementacja BDD

Data ogłoszenia zadania: 7 kwietnia 2016
Termin przesyłania rozwiązań: 29 kwietnia 2016, godz. 23:59

1 Wstęp

Zadanie polega na zaimplementowaniu biarnych diagramów dcyzyjnych (BDD) w wybranym przez siebie języku programowania i rozwiązaniu przy pomocy tej implementacji jednego lub dwóch zadań weryfikacyjnych. W tym opisie będziemy bazować na skrypcie "An introduction to binary decision diagrams" autorstwa H. R. Andersena ([1]). W szczególności przyjmujemy oznaczenia z tego skryptu.

2 Funkcjonalności BDD do zaimplementowania.

To zadanie naśladuje polecenie z rozdziału 8 skryptu [1]. Do zaimplementowania jest szereg podstawowych funkcji BDD. Część z tych funkcji była dyskutowana na szóstych zajęciach [2]. W poniższych poleceniach u jest węzłem BDD i implicite oznacza BDD pod tym węzłem. W szczególności, jeśli u jest korzeniem, to mamy na myśli całe BDD.

print(u) drukowanie BDD na standardowym wyjściu.

mk(i, l, h) znajdowanie numeru węzła u takiego, że $var(u) = i$, $low(u) = l$, $high(u) = h$.

build(t) konstrukcja BDD z wyrażenia boole'owskiego t.

apply(op, u_1, u_2) konstrukcja BDD przez zastosowanie binarnej operacji op do BDD u_1 i u_2.

restrict(u, j, b) ograniczenie BDD u przez zastosowanie podstawienia $[b/x_j]$.

satCount(u) zwraca liczbę wartościwań spłniających u.

anySat(u) zwraca wartościowanie spełniające u.

3 Zadania weryfikacyjne

Poniżej podane są dwa zadania weryfikacyjne. Można wybrać jedno lub dwa z poniższych zadań. Przed ich wykonaniem należy samemu zaimplementowac BDD. Otrzymuje się za to 0.5 punkta.

1. Za pierwsze zadanie opisane niżej można dostać kolejne 0.5 punkta.
2. Za drugie zadanie można dostać 1 punkt.

Jeśli ktoś się zdecyduje na implementowanie pierwszego i drugiego zadania, to łącznie będzie mógł otrzymać 2 punkty za tą serię zadań. Korzystam tu z okazji, żeby przypomnieć zasady punktacji. Jeśli student otrzyma dwa punkty w ciągu semestru, to uzyska ocenę dobrą, a jeśli otrzyma trzy punkty lub więcej, to uzyska ocenę bardzo dobrą.

3.1 Zadanie na temat konika szachowego

Zaprogramuj przy pomocy swojej implementacji BDD rozwiązanie problemu konika szachowego. Mianowicie, dla planszy szachowej $n \times n$ należy stwierdzić, czy konik postawiony w lewym dolnym rogu jest w stanie przejść przez wszystkie pola szachownicy. (zadanie 7.1 ze skryptu [1]).

3.2 Zadanie na temat dypozytora Milnera

W tym zajmiemy się dyspozytorem Milnera (Milner's scheduler). Ten temat jest szczegółowy opisany w rozdziale 7 skryptu [1].

1. Przy pomocy swojej implementacji BDD zaprogramuj tranzycje dyspozytora przy pomocy formuły podanej na stronie 33 skryptu [1].
2. Sprawdż, czy dyspozytor dopuszcza *deadlock*.
3. Sprawdź, czy istotnie zadania są wykonywane w cyklach $1, 2, \ldots, N, 1, 2, \ldots, N, \ldots$. (zadanie 7.4 ze strony 35 skryptu [1]).

Bibliografia

[1] Henrik Reif Andersen. *An introduction to binary decision diagrams*. 1997. URL: `http://people.cs.aau.dk/~srba/courses/SV-08/material/09.pdf` (term. wiz. 29. 04. 2015).

[2] *Szóste laboratorium z WWK*. 2015. URL: `http://www.mimuw.edu.pl/~henrykm/doku.php?id=wkk2016.lab.6` (term. wiz. 29. 04. 2015).