Metoda diferentelor booleene

Conceptul de diferenta booleana a fost introdus de Bob Brayton si este definit ca fiind rezultatul aplicarii operatiei “sau-exclusiv” asupra a doi cofactori.

Definitie

Fie functia f , unde f (x_1,x_2, …,x_i,…, x_n).

Diferenta booleana a functiei f raportat la variabila x_i este :

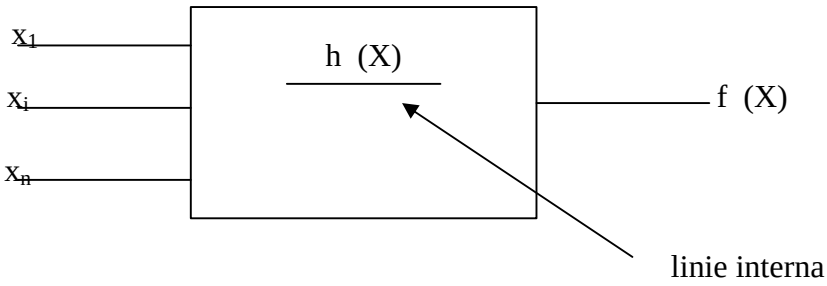
$$\frac{df(x)}{dx_i} = f(\overline{x_i}) \oplus f(x_i) = f(x_1, x_2, \ldots, 0, \ldots, x_n) \oplus f(x_1, x_2, \ldots, 1, \ldots, x_n)$$

Observatie:

Daca $\frac{df(x)}{dx_i} = 0$ atunci f este independenta de variabila x_i

Daca $\frac{df(x)}{dx_i} = 1$ atunci f este senzitiva la valoarea lui x_I

Aplicarea metodei pentru testarea liniilor unui circuit



Aplicarea metodei diferentelor booleene pentru un circuit

Pentru intrarile primare

Testarea defectului x_i blocat pe 0 (blocat pe 1) se desfasoara în doua etape :

a) Activare defect:

Aplica 1 pentru a testa daca linia x_i este blocata pe 0 , respectiv aplica 0 pentru a testa daca linia x_i este blocata pe 1.

b) Senzitivizarea iesirii la schimbarea valorii pe x_i:

Aplica astfel de valori la intrarile primare încât $\frac{df}{dx_i} = 1$.

Observatie: În continuare se va folosi notatia b-l-0 pentru blocat pe 0, respectiv b-l-1 pentru blocat pe 1.

Testele pentru x_i b-l-0 (b-l-1) : $x_i \cdot \frac{df}{dx_i} = 1$ ($\overline{x_i} \cdot \frac{df}{dx_i} = 1$) .

Pentru liniile interne

Testele pentru h b-l-0 (b-l-1) : h(x)$\cdot \frac{df(x,h)}{dh} = 1 \ (\ \overline{h(x)} \cdot \frac{df(x,h)}{dh} \ )$

Observatii:

Prin metoda diferentelor booleene se realizeaza de fapt tot senzitivizarea caii, prin activarea defectului si propagarea lui la iesire.

Diagrame de decizie binara

BDD – Binary Decision Diagrams

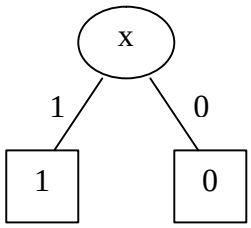
BDD-urile sunt de fapt arbori binari ce reprezinta functii booleene. Prin BDD-uri, avem la dispozitie o unealta grafica intuitiva, dar si un algoritm eficient de lucru cu functii booleene.

Nodurile unui BDD sunt variabilele functiei, ramurile reprezinta valorile (0 sau 1) pe care le poate lua variabila din care pornesc, iar frunzele reprezinta rezultatele functiei booleene.

Ramurile care merg spre stânga, reprezinta valoarea 1 (adevarat) a variabilelor din care pornesc; ramurile care merg spre dreapta reprezinta valoarea 0 (fals) a variabilelor din care pornesc.

Exemplul 1

De exemplu, pentru functia *f=x*, avem BDD-ul:



BDD pentru f=x

Observati ca rezultatul functiei este 1 daca x este 1, si 0, daca x este 0.

Un aspect important în lucrul cu BDD-uri reprezinta optimizarea si reducerea lor. Scopul acestui pas este reducerea marimii BBD-ului, deci si a necesarului de memorie utilizat de algoritmii ce folosesc BDD-uri..

Definitie OBDD:

O diagrama de decizie binara se numeste *ordonata* daca fiecare variabila este întâlnita cel mult o singura data de-a lungul fiecarei cai de la radacina la un nod terminal si variabilele sunt întâlnite în aceeasi ordine pe toate caile de acest fel.

Definitie ROBDD:

O diagrama de decizie binara se numeste *redusa* daca ea nu contine noduri care sa aiba subgrafuri izomorfice, sau noduri pentru care ambele ramuri care pornesc din nod sa pointeze spre aceeasi valoare terminala.

Exemplul 2

Functia este: $f = x_1 + x_2$, adica x_1 *SAU* x_2.

Tabela de adevar		
x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Diagrama neredusa corespunzatoare functiei este:

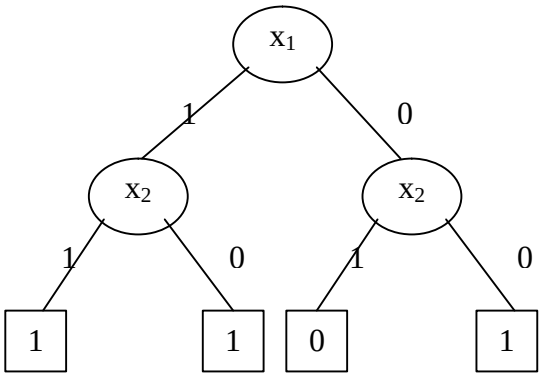


Diagrama neredusa a functiei $f = x_1 + x_2$

Pentru subgraful corespunzator lui x_2, atunci când valoarea lui x_1 este 1, se observa ca rezultatul functiei este 1 indiferent de valoarea lui x_2 . Eliminând nodurile redundante diagrama redusa rezultanta va fi :

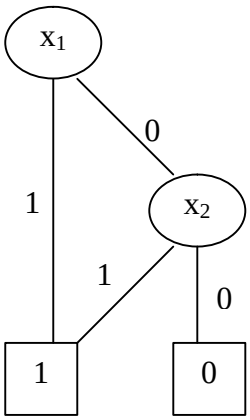


Diagrama redusa al $f=x_1 + x_2$ dupa eliminarea nodurilor redundante

Pachetul BDD

Pachetul-program BDD pe care îl vom utiliza consta în doua aplicatii separate: *bdd* si *bdd2X*. Ambele ruleaza sub Linux.

bdd2X

Acest program primeste ca parametru un fisier cu functia (functiile) booleana, creeaza BDD-ul ordonat si redus si îl afiseaza grafic. Trebuie sa fie activ mediul grafic X-Window (treceti din modul text Linux în modul grafic cu comanda *startx*).

bdd

Acest program primeste ca parametru tot un fisier cu functia (functiile) booleana si afiseaza informatii text despre BDD-ul corespunzator functiei. Lucreaza în mod text. De asemenea, este capabil sa afiseze combinatia de variabile pentru care rezultatul functiei rezulta 1.

bx

Deoarece bdd2X lucreaza implicit cu BDD-uri cu ramuri negate (un alt tip de BDD-uri), vom folosi scriptul *bx*, care executa comanda

bdd2X -o

pe fisierul primit ca parametru. Astfel, *bdd2X* va afisa BDD-ul cu ramuri nenegate.

Reguli în construirea fisierelor cu functii

Se pot folosi expresii, definitii si operatori.

Datele de intrare sunt reprezentate de una sau mai multe reguli sau definitii separate prin . sau ;

La nivelul superior avem reguli, definitii, vectori si „write”.

Un vector este o lista de expresii între paranteze drepte.

O definitie este de fapt o expresie mai speciala.
“let” asociaza unui identificator o expresie.
„write” scrie text la iesirea standard (ecran).
„=” testeaza egalitatea a doua formule.
set este pentru aflarea combinatiei de valori ale variabilelor de intrare pentru care functia da 1.
z|*b* calculeaza z_b în diferentele booleene.

Ordinea evaluarii operatorilor este de la stânga spre dreapta, tinând cont de urmatoarele prioritati:
1.functii predefinite (ex. set)
2. negare postfixata (‘)
3. negare prefixata (~)
4.and
5.or
6.implica, implicat de, echivalent, sau-exclusiv
7.constructie if-then-else (’?’ ‘:’)
8.identitate

Perechile de paranteze pot fi folosite în cazurile în care ordinea dorita intra în conflict cu regulile de mai sus, sau pur si simplu pentru a scoate în evidenta anumite operatii. Spatiile albe, adica spatiile, taburile si liniile vide pot aparea în orice numar între doua bucati de text, si la începutul si sfârsitul fisierului de intrare. Comentariile de tip C pot aparea oriunde unde sunt permise spatiile albe.

Variabilele sunt desemnate de identificatori care consta în litere, cifre sau caracterul underscore, dar trebuie sa înceapa cu o litera. Se face distinctie între literele mari si literele mici. Exista mai multe alternative, variante de alegere pentru operatori (vezi tabelul de mai jos).

Observatie: Identificatorii care reprezinta cuvinte rezervate nu pot fi folositi pentru a desemna variabile!

Variabilele introduse de constructia ‘let’ sunt pastrate într-o tabela separata de simboluri. La evaluare, aceste variabile au prioritate mai mare decât variabilele simple care au acelasi nume, ca de exemplu :

let x = a V b.

x & b == c. Unde ‘x’ din ‘x &b’ este egal cu ‘a V b’.

Constantele 0 si 1 vor fi interpretate ca fals si adevarat, respectiv.

Operatorii

operator	***sens***	***variante***
‘	negare postfixata	
~	negare prefixata	!,not
^	si	and,&
V	sau	or,+
->	implica	implies
<-	implicat de	implied by
<->	echivalent	equiv,==
*	sau exclusiv	><,xor
? :	if-then-else	
=	egal,identic	

Operatorii si echivalentele lor din tabelul de mai sus se pot folosi în fisierele-functie pe care *bx* sau *bdd* sa le proceseze.

Observatie: nu se admite *nand*, *nor*. Se va lucra cu negarea iesirii acestor porti.

Exemplul 3

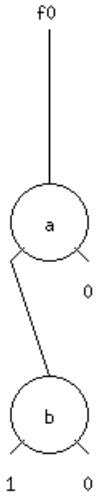
Functia: *f=a and b*

Numele fisierului: *and_g*

Continutul fisierului: *a and b. /*cu punct cu tot*/*

Comanda: *./bx and_g*

Rezultatul:



Valorile 0 si 1 nu reprezinta valorile variabilelor, ci rezultatul final, valoarea functiei. De exemplu, pentru a=1 (ramura din stânga), se ajunge la b. Daca si b=1 (ramura din stânga), rezultatul functiei – frunza – va fi 1 (afisat). Daca a=1 (stânga) si b=0 (dreapta), rezultatul functiei este 0 (afisat). Daca a=0 (dreapta), nici nu se mai verifica b, deoarece acesta este X, adica nu conteaza, rezultatul se da direct, adica 0. Observati ca aceasta aplicatie (*bdd2X* sau *bx* care îl foloseste) genereaza din start diagrama ordonata si redusa.

Exemplul 4

Functia: *f=a and b*

Numele fisierului: *and_t*

Continutul fisierului:

*set(a and b). /*cu punct cu tot*/*

Comanda: *./bdd and_t*

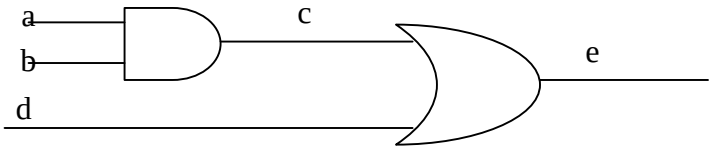
Rezultatul:
Satisfying truth-assignment is:
a := 1
b := 1

BDD si diferente booleene

Vom folosi aplicatia *bdd* pentru a gasi combinatia (testul) care detecteaza un defect presupus, precum si aplicatia *bdd2X (bx)* pentru afisarea arborelui grafic asociat testului respectiv.

Exemplul 5

Circuitul:



Expresia functiei: e = (a and b) or d

Cu bdd

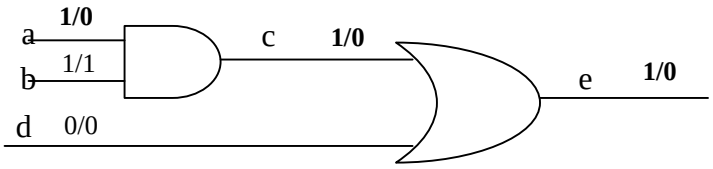
Numele fisierului: *ex5_t*

Defectul presupus: **a** s_a_0

Continutul fisierului:
let e= (a and b) or d.
*let dif = e|a xor e|a'. /*trebuie sa calculam diferenta booleana*/*
let diff = a and dif.
set (diff).

Comanda: *./bdd ex5_t*

Combinatie satisfacatoare (adica pentru care rezultatul functiei este 1):
a := 1
b := 1
d := 0
Propagarea defectului la iesire



Cu bdd2X (bx)

Continutul fisierului:
let e= (a and b) or d.
*let dif = e|a xor e|a'. /*trebuie sa calculam diferenta booleana*/*
let diff = a and dif.
diff.

Comanda: *./bx ex5_g*

Rezultatul:

Observati ca pentru a=1, b=1, d=1, functia va rezulta 0.

0 din stânga lui d nu este valoarea lui d (în directia stânga d fiind 1), este valoarea finala a functiei daca d=1 si variabilele anterioare toate cu valoarea din stânga, adica 1.

Exemplul 6

Acelasi circuit ca la ex. 5.

Defectul presupus: **c** s_a_1 (linie interna!)

Cu bdd

Numele fisierului: *ex6_t*

Continutul fisierului:
let e= (a and b) or d.
*let dif = e|(a and b) xor e|(a and b)'. /*trebuie sa calculam diferenta booleana*/*
let diff = (a and b)' and dif.
set (diff).

Comanda: *./bdd ex6_t*

Combinatie satisfacatoare:

a	:= 0			a	:= 1
b	:= X		sau	b	:= 0
d	:= 0			d	:= 0

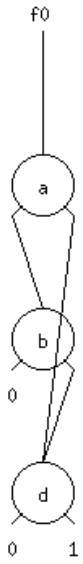
Cu bdd2X (bx)

Numele fisierului: *ex6_g*

Continutul fisierului:
let e= (a and b) or d.
*let dif = e|(a and b) xor e|(a and b)'. /*trebuie sa calculam diferenta booleana*/*
let diff = (a and b)' and dif.
diff.

Comanda: *./bx ex6_g*

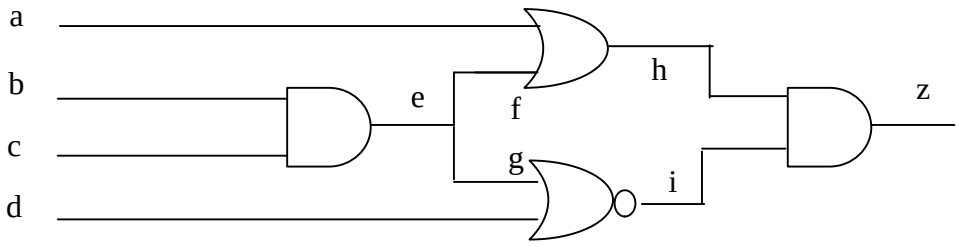
Rezultatul:



Observati ca daca a=0 (spre dreapta), atunci se sare peste b direct la rezultat, adica b=X. Acest lucru se vede si din combinatiile satisfacatoare date anterior de *bdd* în mod text.

Exercitii

1. Încercati exemplele 3,4,5 si 6 (în mod text cu *bdd* si în mod grafic cu *bx*).
2. Scrieti si testati fisierele pentru defectele **b** s_a_0, **f** s_a_0 si **h** s_a_1 din circuitul urmator:



Observatii:

- Acest circuit este cel discutat la laboratorul de diferente booleene.
- Aveti nevoie de 6 fisiere în total (pentru fiecare defect câte doua – pentru *bdd* si pentru *bx*).
- Observati rezultatele pentru f s_a_0, care este defect nedetectabil.