Binarni dijagrami odlučivanja

Seminarski rad u okviru kursa
Automatsko rezonovanje
Matematički fakultet

Milana Kovačević
Ivan Ristović

jul 2018.

Binarni dijagrami odlučivanja (u daljem tekstu *BDD*) i njihova poboljšanja su strukture podataka za reprezentaciju bulovskih funkcija. Iako u osnovi slični binarnim drvetima, rešavaju problem velikog broja čvorova u drvetu uklanjajući redundantne grane (za bulovsku funkciju sa n argumenata, broj mogućih puteva u binarnom drvetu od korena do lista je 2^n, dok je broj čvorova znatno veći). U ovom radu ćemo detaljnije opisati ideju na kojoj su zasnovani BDD, načine za njihovu konstrukciju, kao i razne varijante BDD - *ROBDD*, *FBDD* i *ZDD*. Ovaj rad će pratiti implementacija ROBDD u jeziku *C++*, uz prateće delove koda na nekim mestima.

Sadržaj

1 Bulovske funkcije

Bulovske funkcije su funkcije koje primaju argumente koji su bulovske vrednosti i vraćaju bulovsku vrednost. Bulovske vrednosti mogu biti *true* ili *false*. U nastavku ćemo sa 1 označavati *true*, a sa 0 *false*, što je uobičajena konvencija.

Za bulovsku funkciju sa n bulovskih argumenata, postoji 2^n mogućih ulaza. Pošto je povratna vrednost takodje bulovskog tipa, zaključujemo da postoji 2^{2^n} različitih bulovskih funkcija sa n argumenata, što se vidi iz jednakosti

$$\underbrace{2 * 2 * 2 * \cdots * 2}_{2^n} = 2^{2^n}$$

Funkcije koje primaju neoznačeni broj u opsegu $[0, 2^n - 1]$ se mogu zameniti sa n bulovskih funkcija sa n argumenata. Kao primer, neka je data funkcija F koja prima i vraća neoznačeni ceo broj. Zamenjujemo funkciju F bulovskim funkcijama f_i, $i = 0, 1, \ldots, n-1$. Argumenti funkcije f_i su n binarnih cifara broja, dok je povratna vrednost f_i vrednost i-te binarne cifre rezultata funkcije F. Drugim rečima, svaka od funkcija f_i računa jednu cifru rezultata. Kao konkretan primer, s obzirom da su neoznačeni brojevi u računarima zauzimaju obično 32 bita, funkciju

```
unsigned F(unsigned n);
```

možemo zameniti sa 32 bulovske funkcije

```
bool f0(bool n0, bool n1, bool n2, ... , bool n31);
bool f1(bool n0, bool n1, bool n2, ... , bool n31);
...
bool f31(bool n0, bool n1, bool n2, ... , bool n31);
```

Naravno, moramo se uveriti da su cifre rezultata zaista jednake izlazima bulovskih funkcija. Jedan način za verifikaciju rezultata je primena obe tehnike nad svim mogućim ulazima i poredjenje dobijenih rezultata. Medjutim, čak i za ovako jednostavne funkcije reč je o oko 4 milijarde (2^{32}) mogućih ulaza. Drugi način je da se ove funkcije prikažu putem nekih struktura podataka, i da se funkcije porede tako što se porede njihove reprezentacije preko tih struktura. Drugim rečima, posmatramo funkcije kao podatke. U poglavljima koji slede će biti više reči o strukturama podataka koje se koriste za predstavljanje bulovskih funkcija. Takodje, u narednim poglavljima pod terminom *funkcija* ćemo podrazumevati bulovske funkcije, ukoliko to nije drugačije naznačeno.

2 Binarna drveta odlučivanja

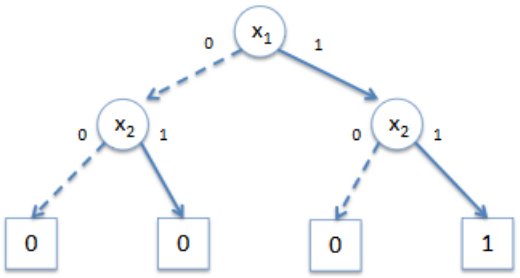
Binarna drveta odlučivanja su u osnovi jako slična binarnim drvetima. Na ovom nivou radimo sa bulovskim funkcijama. Neka je dato n promenljivih $x_1, x_2, \ldots, x_n$ koje predstavljaju ulaze u funkciju f. U korenom čvoru testiramo jednu promenljivu (bez umanjenja opštosti, krenućemo redom, te neka je to x_1). U zavisnosti od vrednosti te promenljive formiraju se dva pod-drveta - jedno u kome je $x_1 = 0$ (*nisko* pod-drvo), a drugo u kome je $x_1 = 1$ (*visoko* pod-drvo). U

svakom pod-drvetu se rekurzivno testiraju ostale promenljive na isti način. Do listova se dolazi kada više nema preostalih ulaznih promenljivih. Posmatrajući jedan put od korena do lista, dobijamo valuaciju za skup $\{x_1, \ldots, x_n\}$.

Uzmimo za primer funkciju

```
bool f_and(bool x1, bool x2) { return x1 && x2; }
```

Matematički zapis ove funkcije bi bio $x_1 \wedge x_2$. Binarno drvo odlučivanja za ovu funkciju je dat na slici 2.1. Polazeći od promenljive x_1, formiramo dve grane na osnovu toga da li je $x_1 = 0$ ili $x_1 = 1$. Od sada pa u buduće ćemo grane u kojima je vrednost 0 crtati isprekidanom linijom, a grane u kojima je vrednost 1 punom linijom, zarad preglednosti.

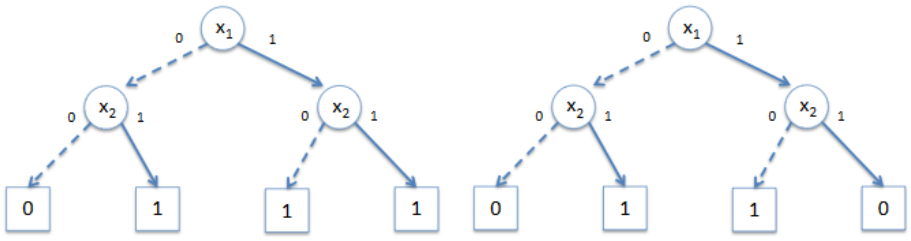


Slika 2.1: Binarno drvo odlučivanja za funkciju $\wedge$

Slično se mogu definisati i ostale korisne funkcije, na primer `f_or` ($\vee$) i `f_xor` ($\oplus$), sa dijagramima na slici 2.2.

```
bool f_or(bool x1, bool x2) { return x1 || x2; }

bool f_xor(bool x1, bool x2) {
    return (x1 && !x2) || (!x1 && x2); [1]
}
```



Slika 2.2: Binarna drveta odlučivanja za funkcije $\vee$ i $\oplus$, redom

[1] Ne možemo koristiti operator ^ jer on operiše nad promenljivima tipa *int*, a ne *bool*.

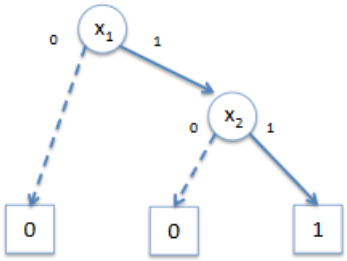
Binarna drveta odlučivanja imaju neke veoma loše osobine. Najveći problem je njihova veličina. Binarno drvo za n ulaznih promenljivih će imati 2^{n-1} unutrašnjih čvorova i 2^n listova. To znači da za neku konstantnu funkciju sa puno promenljivih imamo ogromno drvo iako svi putevi u drvetu vode do istog rezultata.

Uprkos tome, postoje i neke dobre osobine, pre svega *kanoničnost*. Ukoliko testiramo promenljive uvek u istom redosledu (tada se binarno drvo odlučivanja naziva *uredjeno*), onda je drvo jedinstveno za svaku funkciju. Stoga se test ekvivalencije dve funkcije svodi na testiranje ekvivalentnosti njihovih binarnih drveta odlučivanja. Nažalost, zbog velikog broja čvorova u drvetima, problem je eksponencijalne složenosti u odnosu na broj ulaznih promenljivih.

Kako bi se rešio problem eksplozije broja čvorova u drvetu, formiraju se unapredjenja binarnih drveta odlučivanja - binarni dijagrami odlučivanja.

3 Opšti binarni dijagrami odlučivanja (BDD)

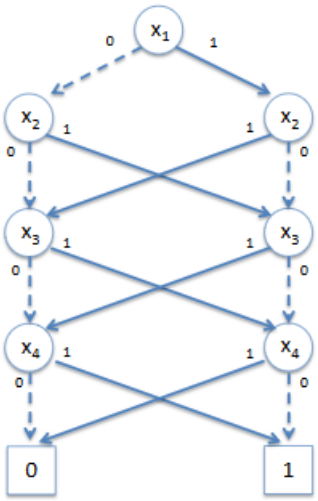
Binarni dijagrami odlučivanja (engl. *Binary Decision Diagrams*, u daljem tekstu *BDD*) [2] su unapredjenje binarnih drveta odlučivanja. Prvo unapredjenje je uklanjanje redundantnih grana u drvetu. Na primer, posmatrajmo funkciju $\wedge$ definisanu u poglavlju 2. U niskom pod-drvetu obe grane koje polaze od promenljive x_2 vode ka vrednosti 0, stoga nije potrebno ispitivati vrednost promenljive x_2 [2]. Ovakvom redukcijom se dobija drvo na slici 3.1.



Slika 3.1: BDD za funkciju $\wedge$

Drugo unapredjenje je dozvoljavanje deljenja identičnih pod-drveta. Ponovo, kako bi čitalac bolje razumeo šta ovo zapravo znači, dajemo primer. Posmatrajmo funkciju koja ima povratnu vrednost 1 ukoliko postoji neparan broj ulaznih promenljivih sa vrednošću 1, a 0 inače. Štaviše, pretpostavimo da imamo četiri ulazne promenljive x_1, x_2, x_3 i x_4. Binarno drvo odlučivanja za ovakvu funkciju bi imalo 15 $(2^4 - 1)$ unutrašnjih čvorova i 16 (2^4) listova. Medjutim, BDD za ovu funkciju ima samo 7 unutrašnjih čvorova i samo dva lista (slika 3.2).

[2] tzv. *lenjo izračunavanje*



Slika 3.2: BDD za funkciju parnosti četiri argumenta

U najgorem slučaju, BDD će biti eksponencijalno veliki u odnosu na ulaz, mada u praksi je to retko slučaj. Takodje, konstantne funkcije se mogu predstaviti samo jednim čvorom, bez obzira na broj ulaznih promenljivih.

Od sada više nećemo govoriti o stablima, jer su BDD u opštem slučaju zapravo usmereni aciklički grafovi (engl. *DAG*). Stoga ćemo u nastavku pod terminima *visoki pod-graf* ili *visoki naslednik* nekog čvora označavati pod-graf do koga vodi 1-grana iz trenutnog čvora. Slično, pod terminima *niski pod-graf* ili *niski naslednik* nekog čvora označavati pod-graf do koga vodi 0-grana iz istog čvora.

4 Redukovani uredjeni binarni dijagrami odlučivanja (ROBDD)

Uredjeni binarni dijagrami odlučivanja (engl. *Ordered Binary Decision Diagrams*, u daljem tekstu *OBDD*) su BDD u kojima se promenljive uvek testiraju u istom redosledu. OBDD se naziva *redukovani*(engl. *Reduced Ordered Binary Decision Diagrams* [4], u daljem tekstu *ROBDD*) ukoliko poseduje sledeće osobine:

- Neredundantnost - Visoki i niski naslednik svakog čvora je različit.
- Jedinstvenost - Ne postoje dva različita čvora koja testiraju istu promenljivu, a čiji su naslednici isti.

Kao što je pomenuto u poglavlju 2, binarna drveta odlučivanja poseduju osobinu kanoničnosti. Medjutim, kako postoje različiti BDD koja odgovaraju jednom binarnom drvetu odlučivanja, jasno je da BDD nemaju ovu osobinu. Uprkos tome, zbog svojih lepih i jasno definisanih osobina, ROBDD su takodje kanonični. Proširivanjem definicije kanoničnosti na ROBDD, dolazimo do sledećeg zapažanja

Zapažanje 4.1. Kanoničnost ROBDD povlači da za fiksni redosled promenljivih, svaka funkcija ima jedinstvenu reprezentaciju u vidu ROBDD. To znači da možemo ekvivalentnost dve funkcije testirati tako što ćemo porediti njihove ROBDD.

ROBDD ima najviše dva lista - 0 i 1 (*konstatni listovi* ili *listovi-konstante*). U nastavku ćemo ponekad crtati više istih listova kako bi se smanjila kompleksnost grafičkog prikaza ROBDD.

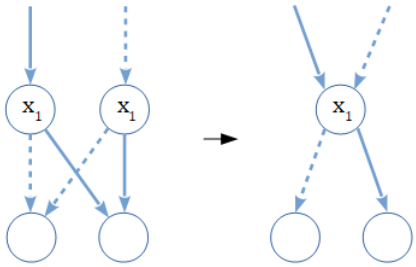
4.1 Konstrukcija ROBDD

ROBDD se može konstruisati na više načina [1, 2]. U ovom poglavlju će najpre će biti opisana naivna metoda konstrukcije (deo 4.1.1), a zatim će ukratko biti opisana efikasnija metoda (deo 4.1.2).

4.1.1 Naivna konstrukcija ROBDD

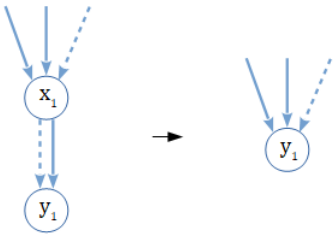
Najjednostavniji način da se konstruiše ROBDD je da se najpre konstruiše ekvivalentno uredjeno binarno drvo odlučivanja. Nakon toga se primenjuju sledeća pravila dokle god ona menjaju postojući dijagram:

- Pravilo spajanja - Svaka dva pod-grafa koja imaju izomorfan BDD se spajaju. U baznom slučaju to predstavlja spajanje listova koji imaju iste vrednosti.



Slika 4.1: Pravilo spajanja

- Pravilo eliminacije - Ukoliko su visoki i niski naslednici nekog čvora izomorfni, taj čvor se uklanja.



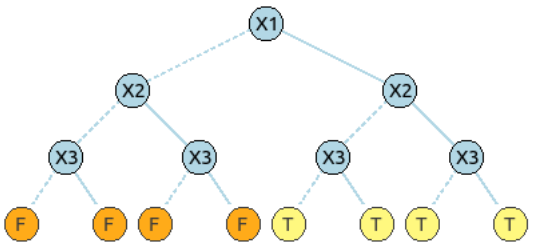
Slika 4.2: Pravilo eliminacije

Mana naivnog pristupa konstrukciji ROBDD je što je on uvek eksponencijalne složenosti. Razlog za to je skupoća konstrukcije kompletnog binarnog drveta odlučivanja. Pored velike vremenske složenosti, taj proces zauzima i eksponencijalno veliku količinu memorije u odnosu na broj promenljivih u ulaznoj formuli. U nastavku dajemo primer konstrukcije ROBDD (prateći naivni pristup koji koristi naš program, uz prateće slike izlaza programa).

Primer 4.2. Posmatrajmo funkciju

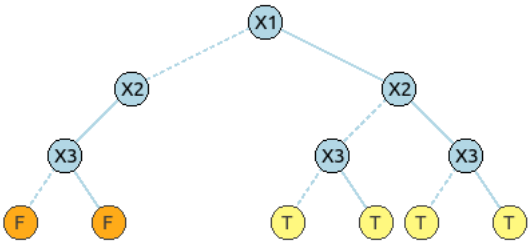
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)$$

Najpre, konstruisaćemo binarno drvo odlučivanja koje odgovara ovoj funkciji:

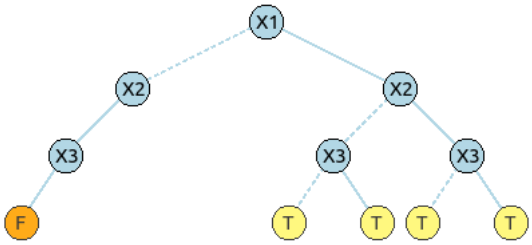


Sada počinjemo da iteriramo dokle god pravila eliminacije ili spajanja imaju efekta.

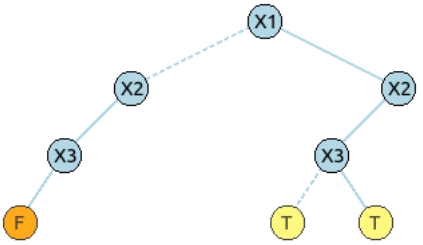
Pravilom spajanja spajamo prva dva podstabla:

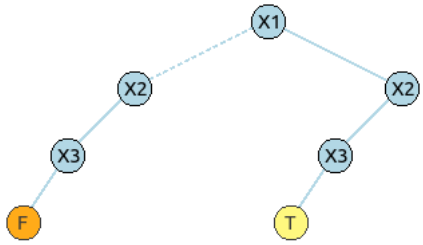


Zatim, takodje pravilom spajanja, spajamo dva F čvora.

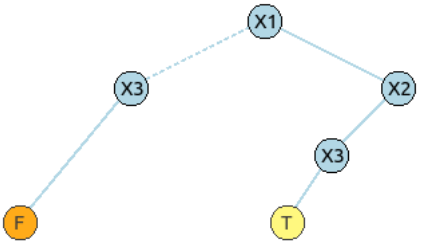


Naredna dva koraka su ekvivalentna prethodnim.

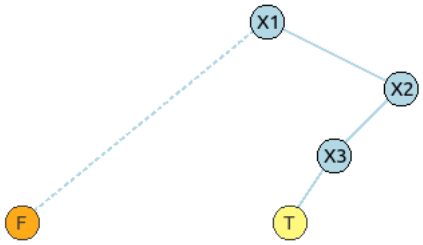




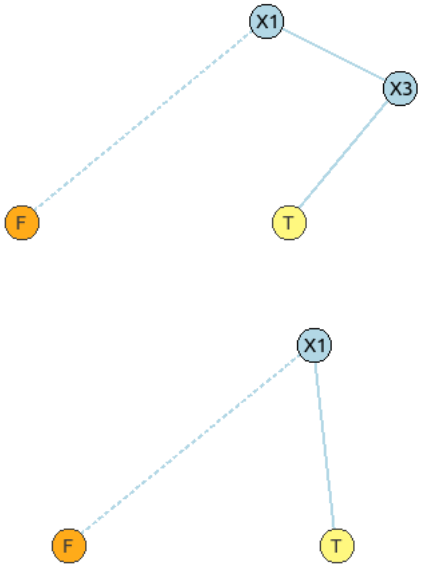
U narednom koraku, pravilom eliminacije, brišemo čvor x_2.



Zatim, pravilom eliminacije, brišemo naredni čvor x_3.



Na ekvivalentan način, uklanjamo preostale čvorove.



Poslednjim korak predstavlja generisani ROBDD. Kao što vidimo, ono je značajno manje od početnog binarnog stabla odlučivanja. Za izračunavanje vrednosti funkcije, dovoljno je ispitati samo vrednost promenljive x_1.

4.1.2 Optimalna konstrukcija ROBDD

Drugi način koji se koristi za konstrukciju ROBDD takodje ima eksponencijalnu složenost. Medjutim, on u praksi daje dobre rezultate. Za razliku od prethodne metode, preskače se medjukorak generisanja binarnog uredjenog drveta odlučivanja. Ovo dovodi do smanjenja potrebne memorije za njegovo generisanje, kao i do značajnog ubrzanja algoritma.

Postupak je rekurzivan: kreće se od bulovske funkcije koja se podeli na podfunkcije za koje se najpre generiše ROBDD. Naredni korak je spajanje ove dve funkcije u jednu, uz korišćenje pravila eliminacije i spajanja koja su opisana u prethodnoj sekciji.

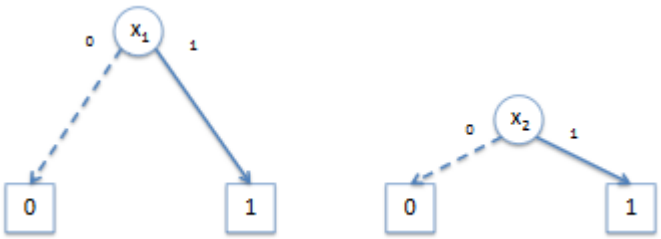
Naredni primer oslikava efikasnu metodu za generisanje ROBDD.

Primer 4.3. Posmatrajmo funkciju

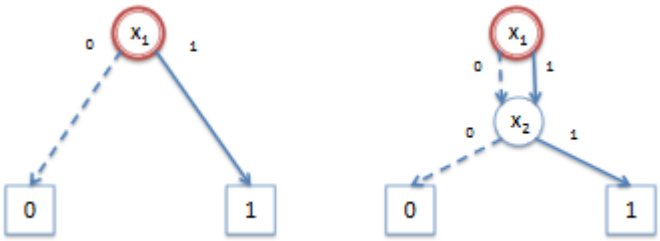
$$(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$$

Konstruisaćemo ROBDD u dva koraka, prvo posmatrajući $x_1 \vee x_2$ i konstruisati ROBDD za taj deo (rekurzivno), a onda isti postupak primeniti na drugom delu - $\overline{x_1} \vee \overline{x_2}$. Zatim čemo spojiti dobijene ROBDD u konačni rezultat.

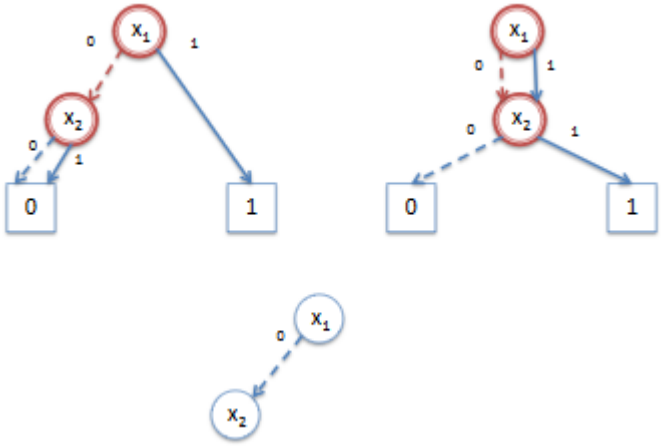
Posmatrajmo sada funkciju $x_1 \vee x_2$. Iako već znamo kako izgleda ROBDD za ovu funkciju (pogledati poglavlje 3), kako bismo ispoštovali postupak idemo dublje (rekurzivno) i dolazimo do promeljivih x_1 i x_2. Pritom, napominjemo da je redosled implicitno izabran tako da x_1 dolazi pre x_2 (iako se može izabrati drugi redosled, sve dok se on svuda poštuje). Rezultujući ROBDD su



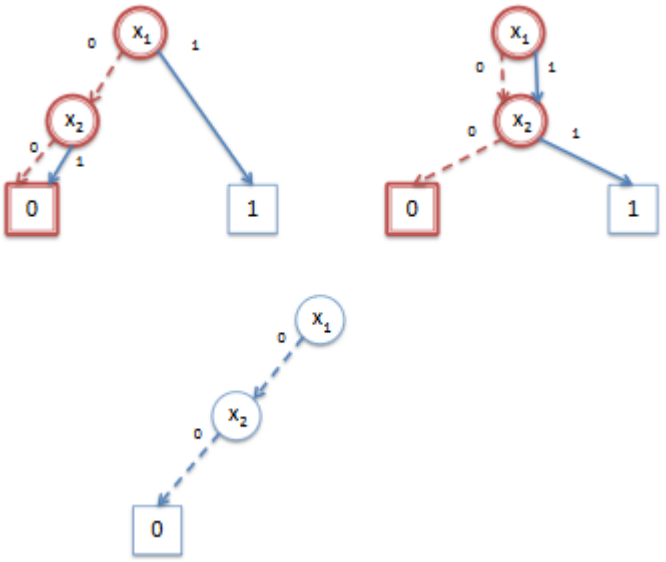
Primenjujući bilo koju bulovsku operaciju nad dva ROBDD sa istim redosledom znači krenuti od korena i pratiti paralelne puteve do listova. Kada stignemo do listova, primenjujemo funkciju nad logičkom konstantom sadržanom u listu kako bismo formirali rezultat za trenutni put. U našem primeru, krećemo od x_1 sa levog dijagrama i hipotetičkog x_1 sa desnog dijagrama:



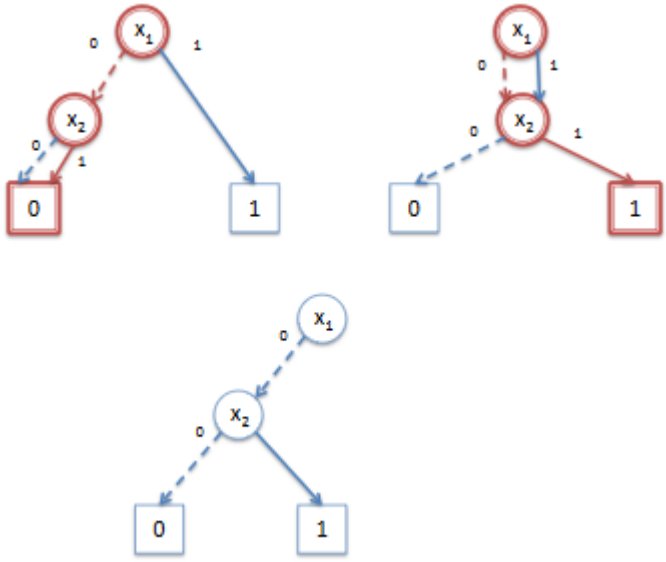
Ako pratimo niske grane u oba dijagrama, dolazimo do hipotetičkog x_2 na levom dijagramu i već posstojećeg x_2 na desnom. Napominjemo da je čvor x_2 u levom dijagramu samo pomoć za razumevanje algoritma, u implementaciji se ne bi eksplicitno pravio taj čvor. Takodje prikazujemo delimično konstruisani rezultujući ROBDD ispod dva ROBDD za promenljive:



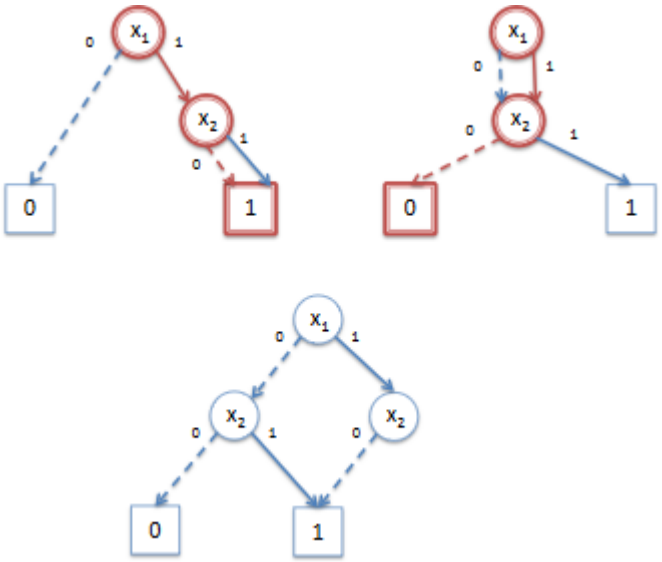
S obzirom da niske grane u oba dijagrama vode ka 0, računamo $0 \vee 0 = 0$ i registrujemo 0 kao rezultat:



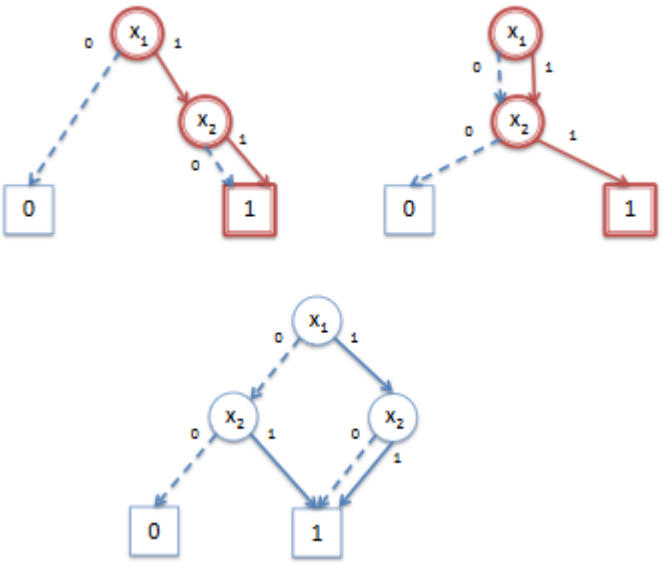
Prateći visoku granu od x_2, dolazimo do vrednosti 0 na levom i 1 na desnom dijagramu. Primenom funckije dobijamo rezultat $0 \vee 1 = 1$:



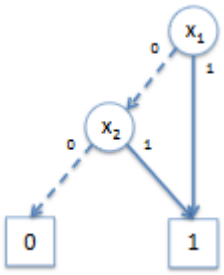
Vraćamo se nazad do x_1 i razmatramo slučaj $x_1 = 1$ i $x_2 = 0$. U ovom slučaju računamo $1 \vee 0 = 1$.



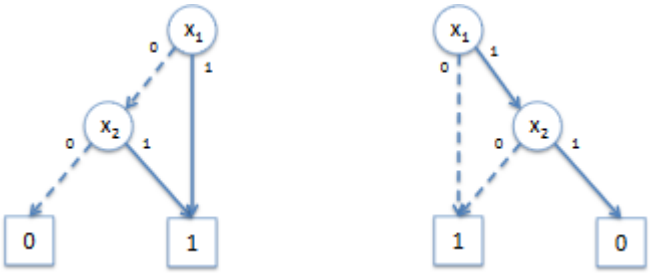
Na kraju pratimo visoke grane iz oba x_2 čvora i dolazimo do vrednosti 1:



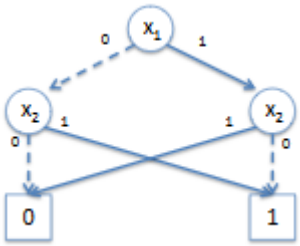
Analizirajući rezultat primećujemo da postoje izomorfizmi u konstruisanom dijagramu (pogledati čvor x_2 sa desne strane). Stoga, primenjujemo pravilo eliminacije i dobijamo sledeći rezultat:



ROBDD za funkciju $\overline{x_1} \vee \overline{x_2}$ se može dobiti invertovanjem vrednosti u listovima već dobijenog dijagrama. Podsetimo se da smo počeli od funkcije $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ i uspešno smo kreirali ROBDD za delove $(x_1 \vee x_2)$ i $(\overline{x_1} \vee \overline{x_2})$, redom:



Ponavljajući gore opisan postupak za spajanje dijagrama dolazimo do konačnog rešenja:



Pažljiv čitalac je možda uočio sličnost funkcije čiji smo ROBDD konstruisali sa funkcijom $\oplus$ koju smo sreli ranije. Ove dve funkcije su zapravo ekvivalentne. Kako bismo pokazali da se funkcije mogu porediti koristeći njihove reprezentacije preko BDD (sve dok su te reprezentacije kanonične, što ROBDD garantuje), možemo se uveriti da važi

$$(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) = x_1 \oplus x_2$$

jer je gore konstruisani dijagram identičan dijagramu za funkciju $\oplus$ (videti sliku 2.2). $\square$

5 Slobodni binarni dijagrami odlučivanja (FBDD)

Slobodni binarni dijagrami odlučivanja (engl. *Free Binary Decision Diagrams* [5], u daljem tekstu *FBDD*) koriste Šenonovu dekompoziciju u svakom čvoru (slično kao kod OBDD) ali se duž različitih putanja mogu koristiti različita uredjenja. Štaviše, svaka promenljiva se sme testirati samo jednom. Ukoliko se promenljive biraju istim redosledom duž svih putanja, rezultat je OBDD. Drugim rečima, OBDD su specijalan slučaj FBDD. Iako nisu u opštem slučaju kanonička struktura, uz modifikacije je moguće postići kanoničnost. Štaviše, FBDD se efikasno minimizuju i izlistavaju.

Dokazano je postojanje funkcija koje se mogu efikasno predstaviti pomoću FBDD (u polinomijalnom prostoru), dok je za OBDD u najgorem slučaju potreban eksponencijalan broj čvorova. Glavni problem je nepostojanje efikasne heuristike za odabir redosleda promenljivih. Iako postoji dosta pristupa za rešavanje pomenutog problema [5], nije pronadjeno značajno brže unapredjenje u odnosu na OBDD.

SAT rešavači se mogu koristiti za konstrukciju i minimizaciju FBDD. Oni se mogu shvatiti kao pretraživači binarnog stabla odlučivanja za datu funkciju. U opštem slučaju, uz fiksni izbor odabira promenljivih u implementaciji SAT rešavača, drvo koje SAT rešavač pretražuje je zapravo OBDD. U slučaju nedeterminističkog izbora promenljivih, to drvo je FBDD.

U nastavku se podrazumeva da je čitalac upoznat sa terminima iskazne logike. Prvo ćemo, uz korišćenje SAT rešavača, opisati proces konstrukcije FBDD (deo 5.1), a zatim proces minimizacije FBDD (deo 5.2).

5.1 Konstrukcija FBDD uz pomoć SAT rešavača

SAT rešavači pronalaze valuaciju v takvu da je $f(v) = 1$ za datu funkciju f, ili pokazuju nepostojanje takvog v. Tokom procesa pretrage valuacija, prave se razni izbori i implikacije. Posmatranjem procesa pronalaženja pomenute valuacije možemo definisati osnovne osobine koje proces konstrukcije FBDD mora da ispunjava:

Zapažanje 5.1. Neka je data bulovska funkcija $f : \mathbb{B}^n \to \mathbb{B}$ u KNF normalnoj formi. Tada važi:

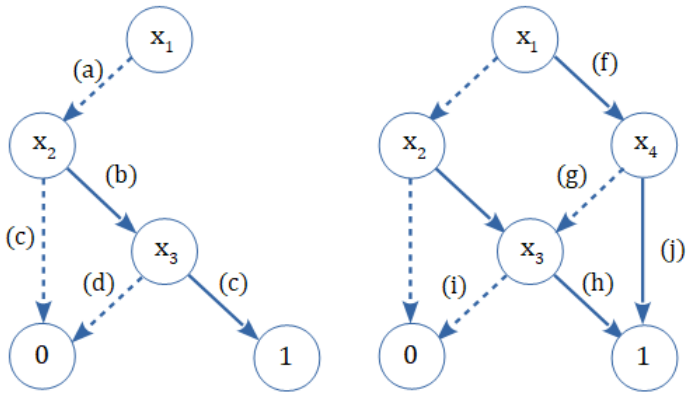
- Svaka valuacija koja zadovoljava f odgovara putu koji vodi ka vrednosti 1 u FBDD koji predstavlja f.
- Svaka konfliktna valuacija odgovara putu koji vodi ka vrednosti 0 u FBDD koji predstavlja f.
- Svaka implikacija $x_i = b$, gde je $x \in f, b \in \mathbb{B}$ odgovara putu koji vodi ka vrednosti 0 u FBDD koji predstavlja f. Ovaj put se moze konstruisati koristeći trenutnu parcijalnu valuaciju v uz $x_i = \overline{b}$.

Primer 5.2. Neka je data funkcija $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4})$. Konstruišemo FBDD za datu funkciju f posmatrajući korake SAT rešavača. Čitav proces je prikazan na levom dijagramu na slici 5.1. Slovima abecede su označeni koraci tokom konstrukcije, zarad lakšeg praćenja.

U početku rešavač uz pomoć heuristike dodeljuje promenljivoj x_1 vrednost 0 *(a)*. Ovo dovodi do toga da x_2 mora imati vrednost 1 *(b)* kako bi klauza $x_1 \vee x_2$

bila zadovoljena. S obzirom da smo upravo iskoristili implikaciju, zaključujemo da je valuacija u kojoj je $x_1 = 0$ i $x_2 = 0$ konfliktna, odnosno $f(0, 0, x_3, x_4) = 0$ *(c)*. U sledećem koraku slično kao na početku se bira vrednost za x_3, na primer 0. Ovakva dodela nas dovodi do konflikta *(d)*. Uzimajući vrednost 1 za promenljivu x_3, sve klauze su zadovoljene bez obzira na vrednost promenljive x_4 *(e)*. S obzirom da je pronadjena zadovoljavajuća valuacija, SAT rešavač prestaje s radom. □

Kao što je prikazano na levom delu slike 5.1, kreiran je parcijalni FBDD funkcije koristeći SAT rešavač koji je pronašao jednu valuaciju u kojoj funkcija f ima vrednost 1. S obzirom da želimo kompletni FBDD, potrebno je da istražimo i ostale putanje (tzv. *All Solution SAT* problem). Ovo se postiže dodavanjem *blokirajuće klauze* u skup klauza koje SAT rešavač pokušava da zadovolji svaki put kad se pronadje valuacija koja zadovoljava tekući skup klauza. U primeru 5.2, s obzirom da je pronadjena valuacija $\{0, 1, 1, x_4\}$, formira se blokirajuća klauza $(x_1 \vee \overline{x_2} \vee \overline{x_3})$, dodaje se u trenutni skup klauza i pretraga se nastavlja (videti primer 5.3). Rezultat pretrage čitavog prostora vrednosti se može videti na desnom delu slike 5.1.



Slika 5.1: Proces konstrukcije FBDD uz pomoć SAT rešavača

Primer 5.3. Nastavljamo sa konstrukcijom FBDD iz primera 5.2 dodajući blokirajuću klauzu $(x_1 \vee \overline{x_2} \vee \overline{x_3})$. Ona izaziva konflikt i povratak (engl. *backtracking*) do x_1, zaključujući $x_1 = 1$ *(f)* kako bi se zadovoljila treća klauza. Sledeći korak je dodela $x_4 = 0$ *(g)* koja dovodi do jedne zadovoljavajuće *(h)* i jedne konfliktne valuacije *(i)* zbog druge klauze. Zadovoljavajuća valuacija se koristi kako bi se formirala blokirajuća klauza $(\overline{x_1} \vee x_3 \vee \overline{x_4})$. Povratkom se dolazi do x_4 i pronalazi se poslednje rešenje *(j)*. S obzirom da je pretražen ceo prostor, pretraga se zaustavlja i FBDD je konstruisan. Napomenimo da su neke grane već bile prisutne u dijagramu, stoga nisu zapravo dodate (koraci *(h)* i *(i)*). □

FBDD dijagrami kreirani ovim postupkom mogu sadržati odredjene izomorfizme u sebi, odnosno ne moraju biti minimalni. U odeljku koji sledi će biti više reči o identifikaciji izomorfizama i redukciji FBDD.

5.2 Detekcija izomorfizama i minimizacija FBDD

FBDD dobijen postupkom opisanim u odeljku 5.1 često sadrži izomorfne pod-grafove. Za razliku od OBDD, gde se izomorfizmi mogu detektovati veoma brzo zbog fiksnog redosleda promenljivih, FBDD zbog svoje osobine nepostojanja fiksnog redosleda odabira promenljivih ne mogu da koriste iste tehnike za detekciju izomorfizama. Neke od poznatijih tehnika redukcije FBDD su *odsečne linije* (engl. *cutlines*) i *odsečni skupovi* (engl. *cutsets*) [5].

Odsečne linije se koriste nad KNF formulama dobijenim iz logičkih kola. Ukoliko su ulazi u kolo C delimično odredjeni, moguće je odrediti neke od internih signala. Skup internih signala, koji formiraju granicu sa nepoznatim signalima kola C, definiše odsečnu liniju. Na implementacionom nivou se implementiraju koristeći heš-tabele, a s obzirom da se sastoje od varijabli mogu se lako reprezentovati preko klauze (slično blokirajućim klauzama opisanim u 5.1). Kako bi se odsečne linije mogle primeniti u opštem slučaju, proces pretrage SAT rešavača bi se morao ograničiti.

Odsečni skupovi za razliku od odsečnih linija uzimaju kao ulaz KNF formulu u opštem obliku. Deo prostora pretrage koji je već pretražen se čuva kao podskup klauza u kombinaciji sa skupom promenljivih sa dodeljenom vrednošću. Odsečni skup je podskup inicijalnog skupa klauza. U njemu se nalaze one klauze koje sadrže barem jedan literal kojem je dodeljena vrednost, i barem jedan literal kojem nije dodeljena vrednost, redom. Može se pokazati da je za detekciju izomorfizama unutar FBDD dovoljno posmatrati odsečni skup.

6 Binarni dijagrami odlučivanja sa potisnutim nulama (ZDD)

Binarni dijagrami odlučivanja sa potisnutim nulama (engl. *Zero-suppressed binary decision diagrams*, u daljem tekstu *ZDD*) [3] predstavljaju vrstu BDD zasnovanih na specijalnom pravilu redukcije. ZDD su posebno efikasni pri rešavanju problema teorije skupova. Naime, za razliku od BDD koji su dobri za reprezentaciju funkcija, ZDD su dobri za reprezentaciju *pokrivanja* - s obzirom da postoji korespodencija izmedju rešenja bulovskih funkcija i familija skupova.

Definicija 6.1. *ZDD* je bilo koji uredjeni BDD sa dodatnim pravilom da *pozitivna grana* svakog unutrašnjeg čvora ne sme da bude povezana sa 0-čvorom. Ukoliko ovo pravilo nije ispunjeno ili ukoliko postoje redundantni čvorovi, rezultujući ZDD se naziva *neredukovani ZDD*.

Kao i BDD, ZDD koriste dva pravila za redukciju - *pravilo spajanja* i *pravilo eliminacije*. Pravilo spajanja izomorfnih pod-grafova je isto kao i kod BDD. Medjutim, postoji razlika u pravilu eliminacije. Definisaćemo formalno pomenuta pravila.

Definicija 6.2. *Pravilo spajanja za ZDD*: Ako unutrašnji čvorovi v_1 i v_2 testiraju istu promenljivu i njihove pozitivne i negativne grane vode ka istim čvorovima, tada eliminišemo jedan od čvorova u ili v i sve ulazne grane u eliminisani čvor preusmeravamo ka preostalom čvoru.

Definicija 6.3. *Pravilo eliminacije za ZDD*: Ako *pozitivna grana* čvora v spaja čvor v sa *0-čvorom* v_0 (konstantni čvor sa vrednošću 0), onda eliminišemo čvor v i preusmeravamo sve ulazne grane u v ka v_0.

Pomenuli smo kako postoji veza izmedju bulovskih funkcija i familija skupova. Pojasnimo malo bolje tu vezu. Svako rešenje bulovske funkcije korespondira sa odredjenim podskupom prostora valuacija. Iako se familije skupova mogu kodirati kao bulovske funkcije, nekada ih je bolje posmatrati u svom osnovnom obliku. U tome se ogleda vrednost ZDD.

7 Zaključak

U ovom radu su prikazane osnovne strukture podataka za predstavljanje bulovskih funkcija. BDD predstavljaju moćan alat za rad sa bulovskim funkcijama, i uz poboljšanje algoritama i pronalascima optimalnijih varijanti BDD, moguće je predstaviti konstruisati BDD veoma komplikovanih funkcija u prihvatljivom vremenu. Takodje, uz poboljšanje SAT rešavača koji danas mogu da za manje od sekunde provere zadovoljivost formula sa stotinama hiljada klauza, FBDD pomeraju granice broja ulaznih argumenata u funkciju. Uz ovaj rad prilažemo i implementaciju BDT i redukciju BDT u ROBDD u programskom jeziku C++.

Literatura

[1] BDD. on-line at:
https://www.cs.cmu.edu/~fp/courses/15122-f10/lectures/19-bdds.pdf.

[2] Beate Bollig, Martin Sauerho, Detlef Sieling, and Ingo Wegener. Binary Decision Diagrams. on-line at:
https://pdfs.semanticscholar.org/31dc/2ebbdb96d8e0c6b733f51df4fc2f5de85eeb.pdf.

[3] Alan Mishchenko. An Introduction to Zero-Suppressed Binary Decision D iagrams. on-line at:
https://people.eecs.berkeley.edu/~alanmi/publications/2001/tech01_zdd_.pdf.

[4] Devi Sivaraman. Reduced Ordered Binary Decision Diagrams (ROBDD). on-line at:
https://www.slideshare.net/RajeshYadav49/reduced-ordered-binary-decision-diagram-devi.

[5] Robert Wille, Gorschwin Fey, and Rolf Drechsle. Building Free Binary Decision Diagrams Using SAT Solvers. on-line at:
http://facta.junis.ni.ac.rs/eae/fu2k73/7wille.pdf.