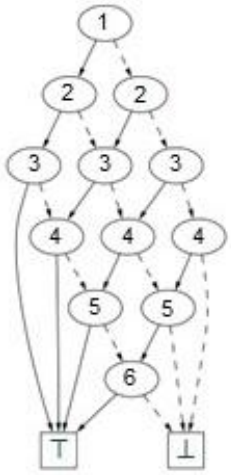
Представљање Булових функција бинарним дијаграмима одлучивања

Мастер рад

Студент: Снежана Сарић
Ментор: проф. Миодраг Живковић



Универзитет у Београду, Математички факултет
Студијски програм Информатика

Садржај

Предговор

У овом раду је објашњен концепт BDD као структуре података, посебно креиране за обраду Булових функција.

Увод рада укратко описује BDD и даје пример Булове функције. На крају поглавља су наведене неке од примена BDD.

Поглавље *Бинарни дијаграми одлучивања* дефинише теоријске концепте на којима се заснива BDD: Булову функцију, перле (енгл. *beads*), канонску форму BDD и везу са перлама, и концепт листе инструкција, која описује правила представљања чворова и правила гранања у BDD.

Након теоријске основе о концепту BDD, следи поглавље *Алгоритми,* које представља алгоритме за конструкцију BDD, алгоритме за обраду Булових функција и пример обраде графова помоћу BDD.

У наставку рада следи поглавље о *Утицају редоследа променљивих Булове функције на комплексност BDD*, и поглавље о примерима *класа Булових функција са једноставним и сложеним BDD*.

Обрада концепта BDD се заокружује у поглављу *Реализација*, у ком се описује имплементација алгоритама за конструкцију и обраду BDD.

1. Увод

Пројектовање дигиталних система и њихова формална верификација обухватају рад са функцијама са великим бројем променљивих. Ефикасна обрада Булових функција у рачунару директно зависи од структуре података и приступа који се користе за њихово представљање. Једна од структура података која је погодна за представљање Булових функција је *бинарни дијаграм одлучивања* (енгл. *Binary Decision Diagram, BDD*). То је ациклички усмерени граф који се добија упрошћавањем бинарних стабала одлучивања [1-4], а заснован је на приступу „завади па владај“ (енгл. *divide-and-conquer*). BDD је због својих особина постао један од основних избора за представљање и обраду Булових функција.

Показује се да је време јефтин, а меморија скуп ресурс у раду са BDD. Обимни и захтевни проблеми се најчешће не реше услед недостатка меморије, а не неприхватљивог времена извршавања. Из тог разлога, постоји значајан број алгоритама за конструкцију и обраду BDD који су прилагођени самом концепту BDD тако да омогућавају потребну временску и меморијску ефикасност.

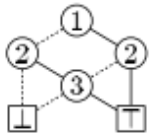
Пример 1. Булова функција M *већинског одлучивања* са n променљивих је *нетачна* када је $n/2$ или више променљивих нетачно, у супротном је тачна. Ова функција се зове и *оператор медијана* (оператор средње вредности), а формално се може записати на следећи начин:

$$M(x_1, \dots x_n) = \left\lfloor \frac{1}{2} + \frac{(\sum_{i=1}^{n} x_i) - \frac{1}{2}}{n} \right\rfloor, \quad (1)$$

Овде "$-\frac{1}{2}$" служи да одлучи о истинитосној вредности функције у корист нула, када је n паран број. Ако се изостави, тада се истинитосна вредност функције са парним бројем променљивих рачуна у корист јединица.

Ако функција има три променљиве $x_1,\ x_2,\ x_3$, тада је вредност функције нетачна када најмање две променљиве имају вредност 0, а иначе је вредност функције тачна.

На слици 1 је приказана структура дијаграма који одговара BDD функције већинског одлучивања са три променљиве.



Слика 1: BDD репрезентација функције већинског одлучивања
са три променљиве из примера 1

BDD репрезентација на слици 1 се тумачи на следећи начин:

- Број сваког чвора представља индекс променљиве. Једна променљива може индексирати више чворова. Обично се индекс променљиве посматра и као ниво BDD стабла.
- Грана сваког чвора представља истинитосну вредност променљиве. Грана која је представљена испрекиданом линијом означава истинитосну вредност *нетачно*, а грана представљена пуном линијом означава истинитосну вредност *тачно*.

Детаљан опис репрезентације Булових функција помоћу концепта BDD је дат у одељку 2.4 Структура BDD: дефиниција и опис.

BDD имају широку практичну примену због компактне репрезентације и ефикасне обраде Булових функција. Структура BDD је флексибилна - може се модификовати у зависности од специфичне примене. Неке од значајнијих модификација BDD су укратко дате у одељку 2.5, а неке од примена су:

1) Примена у графовским алгоритмима
2) Репрезентација кола и манипулација скуповима стања у оквиру верификације коректности хардвера
3) Превођење Булових функција у кола, тестирање и оптимизација секвенцијалних кола, и оптимизације логичких кола

Примена BDD у графовским алгоритмима у раду је илустрована кроз креирање и манипулацију BDD који представљају независне скупове циклуса са n чворова (видети одељак 3.3).

2. Бинарни дијаграми одлучивања

Бинарни дијаграми одлучивања BDD представљају значајну структуру која омогућава ефикасну примену Булових функција, које се користе при моделовању и решавању многобројних практичних проблема.

Први део поглавља се састоји из неколико одељака који описују појмове на којима се заснива концепт BDD. У другом делу поглавља се наводе дефиниција BDD, опис канонске форме, и различите методе представљања BDD. Последњи одељак представља различите варијанте основне структуре BDD.

2.1. Булове функције и кофактори

У овом одељку се излажу основни појмови и тврђења о Буловим изразима, истинитосној табели Булове функције, кофакторима Булове функције, и перлама. Кофактори Булове функције и перле су структуре од посебног значаја јер се користе за директно креирање и манипулацију BDD.

2.1.1. Булове функције

Дефиниција 1. *Булов израз* је израз чија истинитосна вредност може бити *тачно* и *нетачно*, а за који важи следеће:

- Свака Булова променљива x и Булова логичка константа $\top$ и $\bot$ су Булови изрази;
- Ако су x и y Булови изрази, тада је сваки од записа $\neg x,\ x \vee y,\ x \wedge y$ такође Булов израз.
- Булови изрази се могу добити коначном применом горе наведених правила.

Основни оператори се користе за дефинисање других Булових оператора. На пример, ако су x, y две Булове променљиве, тада се Булов оператор импликације $(\Rightarrow)$ дефинише на следећи начин:

$$x \Rightarrow y = (\neg x \vee y) \tag{2}$$

У Буловом изразу заграде се могу изоставити ако се дефинише приоритет Булових оператора.

Дефиниција 2. Нека је $B = \{0,1\}$**.** *Булова функција* представља пресликавање $f: B^n \longmapsto B$, где:

- $(x_1, \dots, x_n) \in B^n$ представљају променљиве Булове функције f, које имају фиксирано уређење и придружене истинитосне вредности из скупа B
- Вредност функције је такође истинитосна вредност из скупа B.

Булов израз једнозначно одређује Булову функцију; обрнуто не важи.

Уређење променљивих је од највеће важности за дефиницију и значење Булове функције. На пример, нека је дат следећи Булов израз:

$$x \Rightarrow y \tag{3}$$

Ако се изабере уређење $x < y$, тада је функција дефинисана изразом $f(x,y) = x \Rightarrow y$ тачна ако први аргумент имплицира други. Ако се изабере уређење $y < x$, онда је функција $f(y,x) = x \Rightarrow y$ тачна ако други аргумент имплицира први.

Уређење Булових променљивих игра кључну улогу у компактним репрезентацијама Булових израза помоћу BDD, као и у њиховој сложености. Одељак 4 излаже основне појмове о утицају уређења променљивих на комплексност BDD, а имплементација претраге простора свих уређења променљивих објашњена у одељку 6.5 илуструје расподелу броја чворова у BDD у односу на могућа уређења.

2.1.2. Кофактори

Кофактори омогућују директну рекурзивну дефиницију Булове функције преко једноставнијих Булових функција ([3] стр. 4, и [2], стр. 91 и 98).

Дефиниција 3. Нека је $f(x_1, \dots, x_n)$ Булова функција. Тада су

$$f_{x_i} = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$
$$f_{\neg x_i} = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \quad (4)$$

редом *позитиван* и *негативан кофактор* функције f.

Кофактори се називају још и *рестрикцијама* функције f. Ако је $f_{x_i} \neq f_{\neg x_i}$, тада се каже да f *зависи* од x_i.

Наредна лема непосредно следи из дефиниције 3.

Лема. Кофактори комутирају, тј. важи:

$$(f_{x_i})_{x_j} = (f_{x_j})_{x_i} = f_{x_i x_j}, \quad (5)$$

и сагласни су са негацијом, конјукцијом и дисјункцијом:

$$(\neg f)_{x_i} = \neg(f_{x_i}),$$
$$(f \wedge g)_{x_i} = f_{x_i} \wedge g_{x_i},$$
$$(f \vee g)_{x_i} = f_{x_i} \vee g_{x_i}. \quad (6)$$

Следи *Шенонова теорема* (енгл.*Shannon*), позната као *теорема развоја*, која чини основу већине алгоритама за обраду BDD.

Теорема 1 (Шенонова теорема развоја). Нека је $f(x_1, \dots, x_n)$ Булова функција. Тада је *Шенонов развој функције f по променљивој x_i*:

$$f(x_1, \dots, x_n) = (\neg x_i \wedge f_{\neg x_i}) \vee (x_i \wedge f_{x_i}). \quad (7)$$

2.2. Истинитосна табела Булове функције

Истинитосна табела Булове функције $f(x_1, \dots, x_n)$ представља бинарну ниску дужине 2^n, која садржи све истинитосне вредности функције f ([1], стр. 204). Табела почиње истинитосном вредношћу функције $f(0, \dots, 0)$, наставља се вредностима $f(0, \dots, 0, 0, 1)$, $f(0, \dots, 0, 1, 0)$, $f(0, \dots, 0, 1, 1)$, ..., и завршава се вредношћу $f(1, \dots, 1)$.

Дефиниција 4. *Ред* истинитосне табеле Булове функције представља број променљивих од којих она зависи.

Наредно тврђење је непосредна последица дефиниције 4.

Тврђење 1. Истинитосна табела τ реда $n > 0$ је облика $\tau = \tau_0\tau_1$, где су τ_0 и τ_1 истинитосне табеле реда $n - 1$.

Истинитосне табеле τ_0 и τ_1 из тврђења 1 одговарају редом *кофакторима* $f(0, \dots, x_n)$ и $f(1, \dots, x_n)$, а њихове истинитосне табеле τ_0 и τ_1 су редом *подтабеле* инстинитосне табеле τ.

Тврђење 2. Свака истинитосна табела реда $n > 0$ има 2^k подтабела реда $n - k$, за $0 \leq k \leq n$, које одговарају појединим вредностима првих k променљивих $(x_1, \dots, x_k)$.

2.3. Перле

Перле успостављају директну везу између истинитосне табеле Булове функције и њеног BDD. Следе дефиниције појмова и тврђења која дефинишу овај концепт ([1], стр. 204).

Тврђење 3. *Квадратна* бинарна ниска дужине 2^n је бинарна ниска која се може записати у облику $\alpha\alpha$, где је α бинарна ниска дужине 2^{n-1}.

Специјалан случај квадратне ниске је *константна* ниска, која је облика $111\dots1$ или $000\dots0$.

Дефиниција 5. *Перла* реда n је истинитосна табела β реда n која није квадратна ниска.

Пример 2. Постоје две перле реда $0\colon 0$ и 1, две перле реда $1\colon 01$ и 10, затим $16 - 4 = 12$ перли реда 2, редом:

$$0001, 0010, 0011, 0100, 0110, 0111, 1000, 1001, 1011, 1100, 1101, 1110 \tag{8}$$

Перле приказане у примеру 2 су заправо истинитосне табеле свих Булових функција $f(x_1, x_2)$ које зависе од истинитосне вредности x_1, у смислу да $f(0, x_2)$ није исто што и $f(1, x_2)$ (видети дефиницију 3).

Уопште, перле реда n су истинитосне табеле функција $(x_1, \dots, x_n)$; постоји $2^{2^n} - 2^{2^{n-1}}$ перли реда n за $n > 0$, јер постоји 2^{2^n} бинарних ниски дужине 2^n, а $2^{2^{n-1}}$ ниски дужине 2^n су квадратне ниске.

Теорема 2 (Веза између перли и истинитосних табела): Истинитосна табела τ се може записати у облику

$$\tau = \beta^k, \tag{9}$$

где је β јединствена перла, а k је степен броја 2. Тада се за истинитосну табелу τ каже да је *степен* перле β, а за перлу β да је *корен* истинитосне табеле τ.

Доказ. Теорема 2 се доказује индукцијом по дужини истинитосне табеле τ.

База индукције: истинитосне табеле дужине 1 су перле, редом 0 и 1.

Индуктивна хипотеза: Нека теорема 2 важи за произвољну истинитосну табелу τ' дужине 2^{n-1}, тј. постоји јединствена перла β тако да је $\tau' = \beta^r$, r је степен броја 2.

Корак индукције: Нека је τ истинитосна табела дужине 2^n. Ако τ није перла ни константна ниска, тада она мора бити квадратна ниска своје подтабеле дужине 2^{n-1}, односно тада је $\tau = \tau'^2$, где је τ' такође истинитосна табела. Тада по индуктивној хипотези важи да је

$$\tau = \tau'^2 = \beta^{2r}, \tag{10}$$

r је степен броја 2, а перла β је корен обе истинитосне табеле τ и τ'.

■

Пример 3. Нека су дате истинитосне табеле $\tau_1 = 1011101100001111$ и $\tau_2 = 1011101110111011$. Очигледно је истинитосна табела τ_1 перла, тј. не постоји подтабела τ' таква да је $\tau_1 = \tau'\tau'$. Међутим, истинитосна табела τ_2 јесте квадратна ниска; према теореми 2 она се може записати као:

$$1011101110111011 = (10111011)^2 = (1011)^4 \tag{11}$$

Дакле, истинитосна табела τ_2 представља степен перле 1011, а перла 1011 је њен корен.

Мање формално, може се рећи да су перле Булове функције оне подтабеле њене истинитосне табеле које су такође перле.

2.4. Структура BDD: дефиниција и опис

Бинарни дијаграм одлучивања BDD представља усмерен ациклички граф (бинарну мрежу) са дељеним ацикличким подграфовима. Наредна два поглавља детаљно описују структуру и особине BDD, као и концепте уређености и редукованости BDD.

2.4.1. Уређена форма BDD

Дефиниција 6. BDD Булове функције $f(x_1, \dots, x_n)$ је *усмерен ациклички граф* (енгл. *Directed Acyclic Graph, DAG*), $(V \cup \{0,1\}, E)$, где је:

- V скуп *унутрашњих чворова* u којима се придружује индекс i променљиве x_i
- $\{0,1\}$ скуп *завршних* чворова, где 0 одговара истинитосној вредности *нетачно*, а 1 одговара истинитосној вредности *тачно* (даље у раду $\boxed{\perp}$ и $\boxed{\top}$, редом)
- E представља скуп усмерених грана $e = (u, v)$ које повезују чворове $u, v \in V \cup \{0,1\}$, у смеру од чвора u ка чвору v. Постоје две врсте грана:
    - Грана која води од чвора u до чвора v, када променљива која је придружена чвору u има истинитосну вредност 0, зове се *0-грана* (или лева грана), а чвор v је *леви син* чвора u, $v = low(u)$. 0-грана се представља испрекиданом линијом.
    - Грана која води од чвора u до чвора v, када променљива која је придружена чвору u има истинитосну вредност 1, зове се *1-грана* (или десна грана), а чвор v је *десни син* чвора u, $v = high(u)$. 1-грана се представља пуном линијом.

Другим речима, грана може повезивати два унутрашња чвора, или унутрашњи чвор са завршним чвором.

Дефиниција 7. Нека је π задато уређење променљивих $x_1, \dots, x_n$ Булове функције $f(x_1, \dots, x_n)$. *Уређен* BDD (енгл. *Ordered BDD, OBDD*) према задатом уређењу променљивих π је усмерен ациклички граф са тачно једним чвором са улазним степеном 0, који има наредне особине:

- Постоје тачно два чвора без излазних грана, односно два чвора којима је излазни степен једнак 0: то су *завршни чворови* који садрже истинитосне вредности Булове функције $\boxed{\bot}$ и $\boxed{\top}$.

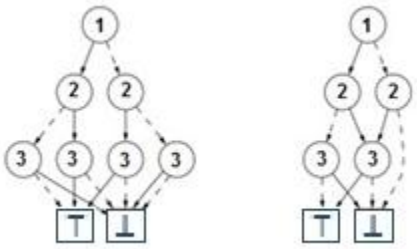
- Чворови који нису завршни зову се *унутрашњи чворови*. То су чворови којима се придружује одговарајућа променљива x_j (односно њен индекс j). У општем случају, постоји више унутрашњих чворова са истим индексом променљиве. *Улазни степен* унутрашњег чвора може бити већи или једнак 1. *Излазни степен* корена и сваког унутрашњег чвора је једнак тачно два, јер сваки унутрашњи чвор има левог и десног сина, са којима је повезан левом односно десном граном.

- Редослед у ком се појављују променљиве чворова на било ком путу у графу мора бити конзистентан са задатим уређењем променљивих, π. Другим речима, за свака два чвора u и v којима су у графу придружене редом променљиве x_i и x_j, важи да ако постоји грана од u до v, тада мора да важи $x_i <_\pi x_j$.

- Приликом графичког представљања BDD, у сваки унутрашњи чвор и корен се уписује индекс j променљиве x_j која му је придружена; ови чворови се у графу означавају са ⓙ. У завршне чворове се уписују одговарајуће истинитосне вредности $\boxed{\bot}$ и $\boxed{\top}$.

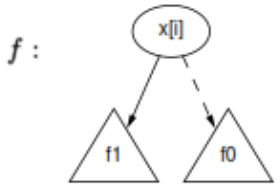
Сваки унутрашњи чвор мора бити *доступан* из корена, тј. мора да постоји пут од корена до сваког унутрашњег чвора у BDD. Пут се наставља левом граном из чвора ⓙ за $x_j = 0$, или десном граном из чвора ⓙ за $x_j = 1$. Пут кроз BDD се завршава у једном од два завршна чвора, за доделу истинитосних вредности променљивама које одговарају изабраним гранама пута. У завршном чвору се налази тражена истинитосна вредност Булове функције.

Пример 4. Нека је задато уређење променљивих $\pi = x_1 < x_2 < x_3$. На слици 2 су приказана два различита OBDD функције $f(x_1, x_2, x_3) = x_1 \wedge x_2 \vee x_1 \wedge \neg x_2 \wedge \neg x_3$, према задатом уређењу π.



Слика 2: Два OBDD функције $f_\pi(x_1, x_2, x_3) = x_1 \wedge x_2 \vee x_1 \wedge \neg x_2 \wedge \neg x_3$ из примера 4

Чвор v са ознаком x_i у OBDD одговара Шеноновом развоју у односу на променљиву x_i (видети одељак 2.1.2). Ако је корену OBDD Булове функције $f(x_1, \dots, x_n)$ придружена променљива x_i, тада су леви и десни син корена OBDD редом корени његова два под-OBDD, при чему под-OBDD чији је корен леви син представља функцију $f_{\neg x_i}$, а други под-OBDD са кореном у десном сину представља функцију f_{x_i}. Слика 3 приказује ову релацију.



Слика 3: Шенонов развој OBDD функције f у чвору x_i

Више детаља погледати у књизи [2], стр. 89.

2.4.2. Канонска форма BDD

Канонска форма BDD омогућује ефикасно извршавање алгоритама за обраду Булових функција.

Дефиниција 8. Нека су чворовима u, v редом придружене промењиве x_i, x_j, где су i, j индекси променљивих. У складу са дефиницијама 6 и 7, нека су чворови $low(u)$ и $high(u)$ редом леви и десни син чвора u, до којих се долази левом, односно десном граном из чвора u. BDD је у *канонској форми* ([1], стр. 203 и [2], стр. 92) ако су испуњена следећа два услова:

1) Редукованост:

а) Не постоји унутрашњи чвор u такав да је $low(u) = high(u)$.
б) Не постоје два различита унутрашња чвора u и v, таква да је $i = j$, $low(u) = low(v)$, $high(u) = high(v)$.

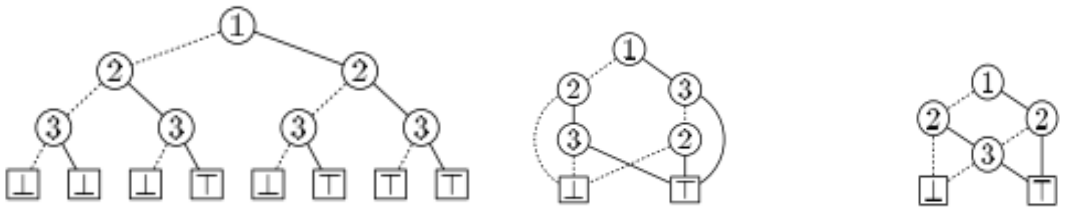
2) Уређеност - Ако постоји грана из чвора u ка чвору v, тада мора да важи да је $i < j$.

Услов уређености обезбеђује да ниједна променљива x_i није проверена два пута током евалуације вредности функције, док услов редукованости обезбеђује да нема непотребно удвостручених чворова.

Дефиниција 9. *Нередуковани бинарни граф* је граф који не испуњава услов редукованости, према дефиницији 8.

Специјалан случај нередукованог бинарног графа представља комплетно бинарно стабло Булове функције, које је илустровано на слици 4(а) у примеру 5.

Пример 5. На слици 4 су приказане различите репрезентације Булове функције већинског одлучивања (видети пример 1, одељак 1).



(а) Нередукован граф (б) Неуређен граф (в) Редукован уређен BDD

Слика 4: Различите репрезентације Булове функције већинског одлучивања из примера 5

Чвор на врху BDD представља корен дијаграма. Корену се увек придружује променљива са најмањим индексом у природном уређењу променљивих, што се лако уочава на дијаграму (в)). Дијаграм (а) не испуњава услов редукованости (на пример, други и трећи чвор којима је придружена променљива x_3 су иста, у смислу да међусобно имају исте леве и десне синове), а дијаграм (б) не испуњава услов уређености у односу на природно уређење променљивих (променљива x_3 је у дијаграму пре променљиве x_2).

У раду се даље под појмом BDD подразумева бинарни дијаграм одлучивања који је *уређен и редукован* (енгл. *Reduced Ordered Binary Decision Diagram, ROBDD*), односно бинарни дијаграм који је у канонској форми, према дефиницији 8.

2.4.3. Веза канонске форме BDD и перли

Истинитосне подтабеле функције f које су реда $n+1-k$ одговарају њеним потфункцијама $f(c_1,\dots,c_{k-1},x_k,\dots,x_n)$ тог реда, где је $k=1,2,\dots,n$. Перле функције f које су реда $n+1-k$ одговарају оним потфункцијама које зависе од променљиве x_k (то је прва променљива у низу којој није додељена истинитосна вредност, па је зато могуће гранање). Стога, свака таква перла одговара унутрашњем чвору означеним са k у BDD. Ако је k ознака унутрашњег чвора који одговара истинитосној табели $\tau' = \tau_0'\tau_1'$, његова лева и десна грана директно воде до чворова који одговарају коренима подтабела (тј. потфункција или кофактора) τ_0' и τ_1' ([1], стр. 205).

Из горе наведеног, може се закључити да сваки чвор BDD Булове функције одговара једној перли њене истинитосне табеле, чиме се доказује следећа теорема.

Теорема 3 (Јединственост BDD). Свака Булова функција f има једну и само једну репрезентацију помоћу BDD, тј. постоји тачно један BDD задате Булове функције f.

Дефиниција 10. За задату Булову функцију f, укупан број перли се означава са $B(f)$ и представља *величину* BDD, тј. укупан број чворова које има њен BDD, укључујући и завршне чворове.

Пример 6. На слици 5 је приказан BDD Булове функције већинског одлучивања (видети пример 1, одељак 1). Слика 5(а) представља описану канонску форму BDD, а слика 5(б) репрезентацију BDD помоћу перли.

(а) Редукован уређен BDD

(б) Редукован уређен BDD, представљен перлама

Слика 5: BDD са ознакама променљивих
које одговарају перлама у сваком чвору из примера 6

Може се уочити да је истинитосна табела функције већинског одлучивања са три променљиве 0001011, идентична редоследу завршних чворова у нередукованом дијаграму на слици 4(а), при чему важи пресликавање $0 \mapsto \boxed{\bot}, 1 \mapsto \boxed{\top}$. Различите подтабеле истинитосне табеле већинског одлучивања су редом:

$$\{00010111, 0001, 0111, 00, 01, 11, 0, 1\} \tag{12}$$

од којих су све осим $\{00, 11\}$ перле. Одатле следи да је скуп перли функције већинског одлучивања:

$$\{00010111,\ 0001,\ 0111,\ 01,\ 0,\ 1\} \tag{13}$$

што илуструје дијаграм (б) на слици 5. Функција већинског одлучивања има укупно $B(f) = 6$ перли, па самим тим и њен BDD има укупно 6 чворова.

2.4.4. Представљање BDD листом инструкција

Један начин да се представи BDD је помоћу *листе инструкција* ([1], стр. 206). Листа инструкција директно описује структуру BDD и правила гранања.

Дефиниција 11 (Листа инструкција). BDD функције $f(x_1, \dots, x_n)$ може се задати у облику *листе инструкција* $I_{s-1},\ I_{s-2}, \dots, I_1,\ I_0$, при чему је свака инструкција облика $I_k = (\overline{v_k}\ ?\ l_k : h_k)$, за $k > 1$, где v представља индекс променљиве, а l и h су индекси инструкција.

Инструкција $(\bar{v}\ ?\ l : h)$ се тумачи на следећи начин: ако је $x_v = 0$, изврши инструкцију I_l; у супротном изврши инструкцију I_h.

Индекси инструкција l_k и h_k морају испунити следеће услове:

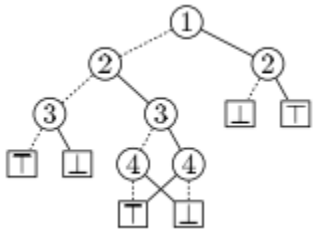
$$l_k < k\ ,\ h_k < k\ ,\ v_{l_k} > v_k\ ,\ v_{h_k} > v_k,\ \text{за}\ \ s > k \geq 2 \tag{14}$$

Условом (14) се захтева да се све гране крећу наниже ка променљивама са већим индексом.

Инструкције $I_1,\ I_0$ се третирају као посебни случајеви; то су просте инструкције које представљају завршне чворове, редом $\boxed{\top}$ и $\boxed{\bot}$. За њих важи да је $l_k = h_k = k$, а индекс променљиве v_k има немогућу вредност, $n + 1$, која се може сматрати нивоом BDD на коме се налазе само завршни чворови.

Ако је $f(x_1, \dots, x_n) = 1$ тако да је резултујући BDD само завршни чвор $\boxed{\top}$, тада је $s = 2$, јер тада инструкција I_1 која одговара завршном чвору $\boxed{\top}$. У супротном, s представља величину BDD, и тада се корен увек представља инструкцијом I_{s-1}.

Пример 7. Нека је дата функција чија је истинитосна табела 1100100100001111. Две директне подтабеле 11001001 и 00001111 су перле, и одговарају редом левом и десном чвору нумерисаним променљивом индекса 2. Процесом контруисања канонске форме из скупа перли задате функције добија се BDD на слици 6:



Слика 6: BDD Булове функције из примера 7

BDD приказан на слици 6 се може представити листом од $s = 9$ инструкција:

$$I_8 = (\overline{1}\ ?\ 7 : 6), \qquad I_5 = (\overline{3}\ ?\ 1 : 0), \qquad I_2 = (\overline{4}\ ?\ 0 : 1)$$
$$I_7 = (\overline{2}\ ?\ 5 : 4), \qquad I_4 = (\overline{3}\ ?\ 3 : 2), \qquad I_1 = (\overline{5}\ ?\ 1 : 1)$$

$$I_6 = (\overline{2}\ ?\ 0 : 1), \qquad I_3 = (4\ ?\ 1 : 0), \qquad I_0 = (\overline{5}\ ?\ 0 : 0) \tag{15}$$

где је $v_8 = 1,\ l_8 = 7, h_8 = 6, v_7 = 2,\ l_7 = 5,\ h_7 = 4,\ \dots,\ v_0 = 5,\ l_0 = h_0 = 1.$

2.5. Варијанте BDD

Постоје разне модификације BDD, развијене са циљем да се превазиђу слабости BDD као структуре података, које су уочене код представљања и обраде одређених Булових функција. То се најпре односи на функције које за ма који редослед променљивих производе експоненцијално велики BDD. Општа структура BDD се модификује и оптимизује на разне начине, да би обрада таквих функција била ефикаснија ([1-4]).

Модификације се најчешће односе на одустајање од неког од захтева које треба да испуни канонска форма BDD. На пример, варијанта BDD која дозвољава различита уређења променљивих на различитим путањама, уместо фиксираног редоследа, назива се *слободни* BDD (енгл. *Free BDD, FBDD*). Затим, *индексирани* BDD (енгл. *Indexed BDD, IBDD*) дозвољава да се променљива провери више пута на истом путу, чиме се губи на јединствености.

Неке модификације се примењују на структурне елементе BDD, у зависности од самих алгоритама који их користе. На пример, у неким BDD који се користе при решавању одређених проблема, већина десних грана води до чвора $\boxed{\perp}$. У том случају се користи структура *нула-потиснутих* BDD (енгл. *Zero-Suppressed BDD, ZBDD*, [1], стр. 249), чији се чворови интерпретирају другачије него код обичних BDD - када постоји грана између чворова i и j, за $j \geq i + 1$, то значи да је вредност Булове функције нетачна, осим уколико не важи да је:

$$x_{i+1} = \cdots = x_{j-1} = 0 \tag{16}$$

Ова структура користи другачије правило редукције - чвор се изоставља ако његов леви син води до вредности функције, а не изоставља се када су његови наследници идентични. На пример, BDD који представља независне скупове циклуса са n чворова (видети одељак 3.4) има много чворова којима десна грана води до чвора $\boxed{\perp}$. ZBDD у том случају представљају ефикасну замену за обичну структуру BDD.

Други пример је структура *диференцијалног* BDD (енгл.*Differential BDD, Δ-BDD*), у коме се чворовима не придружују индекси променљивих, него разлика у индексима у односу на вишу (претходну) променљиву у уређењу.

3. Алгоритми

Ово поглавље садржи детаљан опис алгоритама за конструкцију и манипулацију BDD, и алгоритма за проналажење независних скупова циклуса са n чворова помоћу BDD.

3.1. Алгоритми за конструкцију BDD

Овај одељак представља различите технике које се користе за конструкцију BDD. Алгоритми се разликују према приступу конструкцији BDD и форми улазних података. Улаз алгоритма може бити једна или више Булових функција, које се представљају:

- истинитосном табелом у форми бинарне ниске (видети одељак 2.2), или
- нередукованим бинарним графом (видети одељак 2.4.2, дефиницију 9).

У овом одељку су описане следеће технике за конструкцију BDD:

- Конструкција разлагањем перли
- Редукција одоздо на горе
- Конструкција BDD полазећи од кофактора

За сваки алгоритам се претпоставља природан редослед променљивих Булове функције.

Имплементација прва два од три алгоритма горе је описана у поглављу 6.

3.1.1. Конструкција разлагањем перли

Конструкција BDD разлагањем перли ([1], стр. 205) гради BDD директним приступом од корена ка завршним чворовима. Главна идеја алгоритма је креирање свих перли истинитосне табеле Булове функције, и креирање одговарајућег чвора за сваку перлу, јер чворовима једнозначно одговарају перле (видети одељак 2.4).

Алгоритам користи помоћни алгоритам чији је улаз истинитосна табела Булове функције, у форми бинарне ниске. Улога овог алгоритма је креирање корена BDD и позив главног рекурзивног алгоритма који конструише BDD из перли.

У сваком пролазу главног алгоритма, перла текућег чвора w се дели на две једнаке подниске, и свака подниска се проверава да ли је перла, или квадрат:

- Уколико је перла, алгоритам проверава да ли је већ креиран чвор u коме је та перла придружена (у неком од претходних пролаза алгоритма):
    - Уколико није, креира се чвор v коме се та перла придружује.
    - Уколико јесте, текући чвор w се спаја на чвор u тако да чвор w постаје отац чвора u (нови чвор се не креира за текућу перлу).
- Уколико је подниска константна, директно се спаја на завршни чвор одговарајуће истинитосне вредности, тј. $\boxed{\bot}$ или $\boxed{\top}$.

- Уколико је подниска квадратна, а није константна, она се дели на два једнака дела, и само половина подниске се даље испитује истим процесом, при чему се за њу не креира нови чвор.

- Перла која је придружена новокреираном чвору v се даље дели на две једнаке подниске, леву и десну, и за сваку подниску се понавља исти процес. За чвор v се креирају одговарајући леви односно десни син уколико су лева односно десна подниска перле. Поступак се понавља док не остану подниске дужине 1, које су константне ниске и које одговарају једном од завршних чворова, $\boxed{\top}$ и $\boxed{\bot}$.

Специјалан случај представља истинитосна табела Булове функције која је константна ниска. Тада се не улази у процес креирања BDD, већ се директно исписује њена истинитосна вредност.

Овим приступом BDD се гради ниво по ниво, без креирања редундантних чворова.

У наставку је дат псеудокод помоћног алгоритма, која има улогу да провери унос истинитосне табеле, креира иницијалан BDD, и позове главни алгоритам за конструкцију BDD из перли Булове функције.

Псеудокод помоћног алгоритма

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својом истинитосном табелом дужине 2^n.

***Algoritam inicijalizacija_BDD*(*istinitosna_tabela*)**

Улаз: $istinitosna\_tabela$ - бинарна ниска дужине n, при чему је n степен броја два

Излаз: нема; алгоритам позива рекурзивни алгоритам BDD_iz_perli

begin

kreirati prazan bdd

Ако је истинитосна табела константна бинарна ниска 111...1 или 000...0, она се приказује
и излази се из алгоритма

if length$(istinitosna\_tabela) == 1$ o**r** $istinitosna\_tabela$ je konstantna niska **then**
print $istinitosna\_tabela$
exit

while $istinitosna\_tabela$ je kvadratna niska **do**
$istinitosna\_tabela =$ prva polovina niske $istinitosna\_tabela$

kreirati čvorove $koren$, $\boxed{\top}$, $\boxed{\bot}$
$koren.perla = istinitosna\_tabela$
$\boxed{\top}.perla = 1$
$\boxed{\bot}.perla = 0$

dodati $koren$, $\boxed{\top}$, $\boxed{\bot}$ u bdd

$leva\_podniska =$ prva polovina niske $istinitosna\_tabela$
$desna\_podniska =$ druga polovina niske $istinitosna\_tabela$

$BDD\_iz\_perli(leva\_podniska, koren, levo, bdd)$
$BDD\_iz\_perli(desna\_podniska, koren, desno, bdd)$

end

Псеудокод главног алгоритма за креирање BDD из перли је дат у рекурзивном облику ради прегледности.

Псеудокод главног алгоритма

Algoritam BDD_iz_perli $(istinitosna\_tabela, u, grana, bdd)$

Улаз:
- $istinitosna\_tabela$ - бинарна ниска дужине n која није константна, при чему је n степен броја два
- u - чвор чија се перла поделила на две подниске које се испитују
- $grana$ - ознака да ли је ниска која се испитује лева (вредност $levo$) или десна (вредност $desno$) подниска чвора u
- bdd - "статичка" променљива која садржи тренутно изграђен BDD.

Излаз: bdd - завршни BDD

begin

```
# База рекурзије - излази се када су бинарне ниске које се проверавају дужине 1
if length(istinitosna_tabela) == 1 then

    if grana == levo then
        # Завршни чвор је онај који одговара истинитосној вредности константне
        # ниске
        u.levi_sin = završni čvor    # (⊤ или ⊥)
    else
        u.desni_sin = završni čvor    # (⊤ или ⊥)

    return bdd

# Испитује се да ли је бинарна ниска квадратна
while istinitosna_tabela je kvadratna niska do

    if istinitosna_tabela je konstantna niska then
        if grana == levo then
            # Преспаја се на завршни чвор који одговара истинитосној
            # вредности константне ниске
            u.levi_sin = završni čvor     # (⊤ или ⊥)
        else
            u.desni_sin = završni čvor    # (⊤ или ⊥)
        break
    else
        istinitosna_tabela = prva polovina niske istinitosna_tabela

# У овом кораку, бинарна ниска која се проверава је перла; испитује се да ли већ постоји
# чвор у bdd коме је та перла придружена, и ако не постоји, креира се нови чвор коме се
# перла придружује, а новокреирани чвор се додаје у bdd

if postoji čvor v u bdd tako da v.perla == istinitosna_tabela then

    if grana == levo then
```

$u.levi\_sin = v$
else
$u.desni\_sin = v$
else
kreirati novi čvor w
$w.perla = istinitosna\_tabela$

if $grana == levo$ **then**
$u.levi\_sin = w$
else
$u.desni\_sin = w$

dodati čvor w u bdd
$u = w$

Свака перла се дели на леву и десну подниску, које се проверавају истим процесом у
рекурзивном позиву
$leva\_podniska =$ prva polovina niske $istinitosna\_tabela$
$desna\_podniska =$ druga polovina niske $istinitosna\_tabela$

$BDD\_iz\_perli(leva\_podniska, u, levo, bdd)$
$BDD\_iz\_perli(desna\_podniska, u, desno, bdd)$

end

Доказ коректности

Доказ се изводи индукцијом по $log_2 k$ истинитосне табеле дужине $n = 2^k$.

База индукције: за $k = 0$, истинитосна табела је дужине $2^0 = 1$, што одговара перлама дужине 1, редом 1 и 0.

Индуктивна хипотеза: Алгоритам даје тачан BDD за све бинарне ниске дужине мање од 2^k.

Корак индукције: Ако је дата бинарна ниска дужине 2^k, $k > 0$, јављају се два случаја:

1) Бинарна ниска није квадратна ниска - тада је она перла, па се према алгоритму она дели на две подниске дужине $\frac{2^k}{2} = 2^{k-1} < 2^k$, за које важи индуктивна хипотеза. Корени BDD добијених за две подниске постају делови (под-дијаграми) резултујућег BDD који одговара истинитосној табели.

2) Бинарна ниска је квадратна ниска - тада је она 2^m умножак перле, $m \geq 1$. Према индуктивној хипотези, алгоритам за ту перлу даје BDD, у ком мења ознаку чвора, после чега BDD одговара истинитосној табели.

■

Пример 8. Нека је дата истинитосна табела Булове функције, 1100100100001111. Како она није квадратна ниска, креира се чвор који је корен BDD. Обрада перле 1100100100001111 је завршена, па се прелази на њену леву и десну подниску.

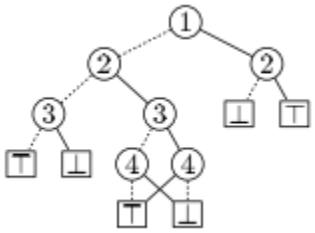
Две директне подниске корена (лева: 11001001 и десна: 00001111) нису квадратне ниске. Како још увек не постоји чвор који одговара некој од ове две перле, за сваку се креира по један чвор, који се у BDD постављају као леви односно десни син корена.

Свака перла се даље дели на две подниске једнаких дужина које се проверавају на исти начин. Скуп резултујућих подниски је:

$$\{1100,\ 1001,\ 0000,\ 1111\} \tag{17}$$

Из скупа се издвајају перле од квадратних ниски - константне ниске 0000 и 1111се директно везују на завршне чворове са истинитосним вредностима, редом $\boxed{\bot}$ и $\boxed{\top}$. Перле 1100 и 1001 се придружују чворовима – редом левом и десном сину чвора чија је перла 1100100. Поделом перли 1100 и 1001 добијају се подниске редом $11, 00, 10, 01$, где су прве две подниске константне и директно се везују на завршне чворове одговарајуће истинитосне вредности, а перле $10, 01$ се придружују редом левом и десном сину чвора 1001, деле се до дужине 1, и везују на одговарајуће завршне чворове.

Слика 7 приказује резултујући BDD.



Слика 7: BDD конструисан разлагањем перли из примера 8

Чворови нумеришу одговарајуће перле: чвор са ознаком 1 је перла 1100100100001111, чворови са ознаком 2 су редом, 11001001 и 00001111, чворови са ознаком 3 су перле 1100 и 1001, и чворови са ознаком 4 одговарају перлама 10 и 01. Нумерација чворова одговара индексима променљивих које су им придружене, а тиме и нивоу BDD на коме се чвор налази.

Сложеност

Сложеност наведеног алгоритма се процењује на основу броја променљивих (дужине истинитосне табеле). Ако је истинитосна табела константно тачна или константно нетачна, алгоритам се извршава у константном времену. У најгорем случају, сваким дељењем бинарне ниске на две подниске једнаких дужина, идентификује се перла за коју се креира нови чвор. Ако је n број променљивих Булове функције, тада је дужина њене истинитосне табеле једнака 2^n, па је у најгорем случају време извршавања $O(2^n)$.

3.1.2. Редукција одоздо на горе

Један начин конструисања BDD из задате истинитосне табеле Булове функције је примена *редукције* ([2], стр. 92-98) на *комплетно уређено* бинарно стабло одлучивања (видети пример 5, слику 4(а)), у смеру од завршних чворова ка корену. Комплетно бинарно стабло увек садржи редундантне чворове - ако је n веће од 1, тада комплетно бинарно стабло има укупно 2^n листова, а у завршном BDD остају само два листа, односно завршни чворови $\boxed{\top}$ и $\boxed{\bot}$.

Нека чворовима u, v редом одговарају променљиве x_i, x_j. Редундантност има два облика:

- Леви и десни син чвора u могу да одговарају функцијама са идентичним подтабелама. У овом случају, одлука о истинитосној вредности у чвору u не садржи нову информацију која би утицала на истинитосну вредност функције, па је чвор u редундантан.

- Више чворова може да одговара једној истој функцији (са истом подтабелом). Овим је заправо иста информација о функцији представљена неколико пута у дијаграму, па се такви чворови сматрају редундантним.

Наредне дефиниције прецизније формулишу редундантност.

Дефиниција 12. Нека су P_1 и P_2 два OBDD. Каже се да су P_1 и P_2 изоморфни ако постоји бијективно пресликавање ϕ из скупа чворова графа P_1 у скуп чворова графа P_2, тако да за сваки чвор v графа P_1 важи једно од наредних тврђења:

1) Два чвора v и $\phi(v)$ су завршни чворови са идентичном ознаком, или
2) $var(v) = var(\phi(v))$, $\phi\big(high(v)\big) = high(\phi(v))$, $\phi\big(low(v)\big) = low(\phi(v))$,

где $var(v)$ означава променљиву чвора v, $high(v)$ одговара десном сину чвора v, а $low(v)$ одговара левом сину чвора v.

Дефиниција 13. OBDD се сматра редукованим ако:

1) Не садржи чвор v такав да је $high(v) = low(v)$
2) Не постоји пар чворова u, v за које важи да су корени подграфова OBDD који су изоморфни.

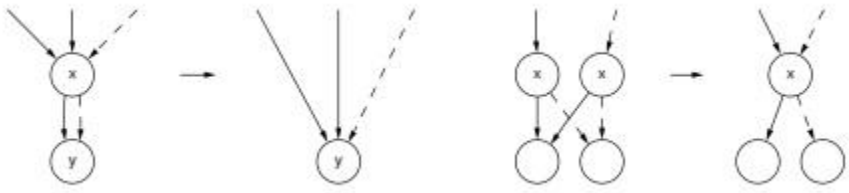
Дефиниција 14. Алгоритам редукције одоздо на горе примењује следећа два правила на OBDD:

1) *Правило елиминације* - Ако лева и десна грана чвора u коме је придружена променљива x_i показују на исти чвор v коме је придружена променљива x_j, чвор u се елиминише, а све долазне гране чвора u се преусмеравају на чвор v.

2) *Правило спајања* - Ако постоје два унутрашња чвора u и v којима је придружена променљива x_i, њихове десне гране воде до чвора w, и њихове леве гране воде до чвора z, тада се један од чворова u, v елиминише, а његове долазне гране се преусмеравају на чвор који остаје.

Другим речима, алгоритам редукције уклања непотребне и дуплиране чворове, и преусмерава гране које их везују.

Илустрација примене наведена два правила је дата на слици 8.



(а) Примена правила елиминације (б) Примена правила спајања

Слика 8: Примена правила редукције

Теорема 4. OBDD P је редукован ако и само ако на њега не може да се примени ниједно правило редукције.

Доказ. Нека је OBDD редукован. Тада се по дефиницији не може применити правило елиминације. Правило спајања такође није применљиво, јер би у супротном OBDD садржао два изоморфна подграфа ([2], стр.93).

Обрнуто, нека је P OBDD на који се не може применити ниједно правило редукције. Тада по

правилу елиминације из дефиниције 14, P не садржи чвор u такав да је $high(v) = low(v)$. Нека P садржи пар различитих чворова u и v, тако да су они корени два изоморфна подграфа у стаблу. Тада се јављају два случаја:

1. случај: Важи да је $high(u) = high(v)$, и $low(u) = low(v)$. Овде се може применити правило спајања, што је контрадикција, јер је претпостављено да се не може применити ниједно правило редукције.

2. случај: Важи да је $high(u) \neq high(v)$, или $low(u) \neq low(v)$. Из дефиниције 12 следи да су два OBDD, чији су корени два различита чвора $high(u)$ и $high(v)$, такође изоморфна. Поново се јављају два различита случаја за чворове $high(u)$ и $high(v)$. Како ова два OBDD зависе само од променљивих које се појављују након променљиве u у редоследу, процес се зауставља након највише n корака, при чему је n број променљивих у полазном OBDD. На крају, тврђење 1) из дефиниције 12 мора да буде испуњено, будући да постоји само један завршни чвор $\boxed{\bot}$ и један завршни чвор $\boxed{\top}$.

■

Алгоритам редукције једноставно примењује горе дефинисана правила докле год је то могуће. Свака примена правила смањује величину OBDD за најмање један чвор. Када ниједно правило више није примењиво, OBDD је тражени редукован BDD.

Смер редукције од завршних чворова ка корену обезбеђује да ниједно правило није поново примењиво у подручју графа које је већ прегледано. Примена правила је систематична јер почиње над чворовима којима се придружује последња променљива у редоследу (са највећим индексом), наставља се над чворовима којима се придружује њена претходна променљива у редоследу, и тако редом до корена, који има најмањи индекс променљиве.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим OBDD (нередукованим бинарним графом). Претпоставља се да је сваком чвору v у OBDD додељен јединствен целобројни идентификатор, $v.id$. Додатно, претпоставља се да су изабрана тачно два завршна чвора $\boxed{\top}$ и $\boxed{\bot}$, који имају најмањи идентификатор id међу свим $\boxed{\top}$ чворовима и свим $\boxed{\bot}$ чворовима на последњем нивоу прослеђеног OBDD.

Algoritam Redukcija_odozdo_na_gore $(obdd)$

Улаз: $obdd$ - OBDD функције $f(x_1, \dots, x_n)$
Излаз: $redukovan\ obdd$ функције $f(x_1, \dots, x_n)$

begin

$svi\_obdd\_čvorovi =$ svi unutrašnji čvorovi u $obdd$
$sve\_promenljive =$ sve promenljive u $obdd$

Због правила спајања, неопходно је да чворови у истом нивоу буду сортирани растуће,
према свом јединственом идентификатору

neka su indeksi promenljivih u skupu $sve\_promenljive$ sortirani u opadajućem poretku

# Пролазак ниво по ниво кроз све променљиве у OBDD, од завршних чворова према
# корену
for each $promenljiva$ **in** $sve\_promenljive$ **do**

$čvorovi\_promenljive = svi\_obdd\_čvorovi[promenljiva]$
neka su čvorovi u skupu $čvorovi\_promenljive$ sortirani u rastućem poretku prema jedinstvenom identifikatoru id

# Правило елиминације, испитује се сваки чвор у посматраном нивоу
for each čvor u **in** $čvorovi\_promenljive$ **do**

    # Проверава се да ли оба сина чвора који се испитује показују на
    # исти чвор
    if $u.levi\_sin.id == u.desni\_sin.id$ **then**

        # Примена правила елиминације
        preusmeriti dolazeće grane čvora u na čvor $u.levi\_sin$
        izbrisati u iz $svi\_obdd\_čvorovi$

Правило спајања, примењује се на преостале чворове, након елиминације за
текући ниво
$čvorovi\_promenljive$ = ažurirati listu preostalih čvorova u tekućem nivou nakon eliminacije

Припрема главних чворова на које ће се вршити спајање осталих редундантних
чворова на текућем нивоу

if length $(čvorovi\_promenljive) > 0$ **then**

    $jedinstveni\_čvorovi = []$
    # Додаје се први чвор у нивоу као главни чвор
    $v = čvorovi\_promenljive[0]$
    dodati glavni čвор v u niz $jedinstveni\_čvorovi$

    # Поставља се индикатор да је чвор који се проверава спојен са једним од
    # главних чворова из низа $jedinstveni\_čvorovi$
    $spojeni =$ **False**

    # Проверавају се остали чворови нивоа - да ли су кандидати за спајање са
    # неким главним чвором, или постају главни чворови
    for each preostali čvor $test\_w$ **in** $čvorovi\_promenljive[1:]$ **do**

        # Провера да ли чвор $test\_w$ има иста оба сина као главни чвор
        # v из низа $jedinstveni\_čvorovi$
        for each v **in** $jedinstveni\_čvorovi$ **do**

            if $v.levi\_sin.id == test\_w.levi\_sin.id$ **and**
              $v.desni\_sin.id == test\_w.desni\_sin.id$ **then**

                # Примена правила спајања
                preusmeriti sve dolazeće grane čvora $test\_w$ na čvor v
                izbrisati čvor $test\_w$ iz $obdd$
                $čvorovi\_promenljive$ = ažurirati listu preostalih čvorova
                $spojeni =$ **True**

        # За чвор $test\_w$ који је био кандидат за спајање није пронађен
        # одговарајући главни чвор v, па се чвор $test\_w$ додаје у низ главних
        # чворова $jedinstveni\_čvorovi$
        if not $spojeni$ **then**
            dodati $test\_w$ u niz $jedinstveni\_čvorovi$

else

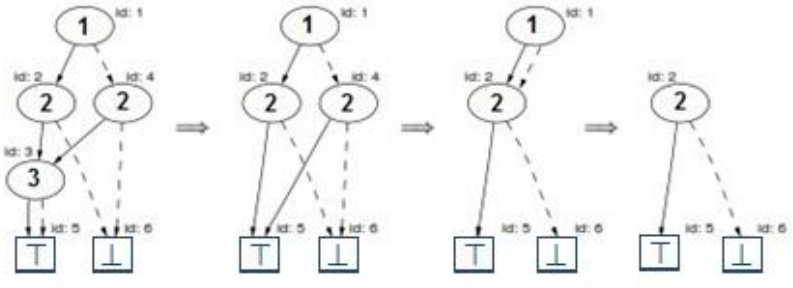
$spojeni =$ **False**

return $redukovan\ obdd$

end

Пример 9. Нека је дат OBDD на слици 9, чијим су чворовима v додељени позитивни цели бројеви, $v.id$. У првом проласку, алгоритам препознаје да су леви и десни син јединог чвора којем је придружена променљива x_3 идентични. Према правилу елиминације, чвор чији је идентификатор $x_3.id$ једнак 3 се уклања. У другом проласку, примењује се правило спајања на чворове којима је придружена променљива x_2, чији су идентификатори једнаки редом $x_2.id = 2$ и $x_2.id = 4$. Чвор чији је идентификатор једнак 2 је задржан, јер је то најмањи идентификатор међу свим чворовима на том нивоу (којима је придружена променљива x_2), а који су кандидати за спајање. Последњи корак алгоритма примењује правило елиминације на чвор чији је идентификатор једнак 1, и којем је придружена променљива x_1. Уочава се да је уклоњен чвор првобитно био корен задатог уређеног бинарног графа, а да након редукције корен траженог BDD постаје чвор променљиве x_2, чији је идентификатор $x_2.id = 2$.

Слика 9 приказује описану трансформацију.



Слика 9: Редукција OBDD из примера 9

Сложеност

Време извршавања алгоритма зависи од времена потребног за сортирање подскупова чворова. Ако се OBDD састоји од $B(f)$ чворова, тада је горња граница времена извршавања алгоритма редукције $\mathcal{O}(B(f) * \log B(f))$. Најгори случај се јавља када се редукује комплетно бинарно стабло одлучивања, што захтева израчунавање свих могућих вредности функције, односно 2^n операција. Из тог разлога је овај приступ конструкције BDD погодан само за функције са релативно малим бројем променљивих.

3.1.3. Конструкција BDD полазећи од кофактора

Конструкција BDD из кофактора Булове функције ([2], стр. 98-102 и стр.112-120) је приступ којим се директно конструише BDD Булове функције $f(x_1, \dots, x_n)$ смером одозго на доле, примењујући Шенонов развој (видети одељак 2.1.2, теорему 1). Овај приступ је ефикаснији и бржи од алгоритма Редукције одоздо на горе (видети одељак 3.1.2), јер не захтева конструкцију OBDD и додатне обраде чворова (елиминацију и спајање).

Алгоритам који конструише BDD полазећи од кофактора Булове функције прво конструише корен BDD, коме се додељује прва променљива x_i у задатом редоследу променљивих дате Булове

функције.

Према Шеноновом развоју

$$f(x_1, \dots, x_n) = \left(\neg x_i \wedge f_{\neg x_i}\right) \vee \left(x_i \wedge f_{x_i}\right), \tag{18}$$

конструишу се две потфункције (кофактори) $f_{\neg x_i}$ и f_{x_i}, као синови корена. За сваки кофактор се одређује прва променљива у редоследу. Приликом креирања сваког новог чвора, проверава се да ли је кофактор већ придружен неком чвору у до тада креираном BDD, да би се избегло креирање редундантних чворова (тиме је задовољено правило елиминације у дефиницији 13). Додатно, за оне кофакторе за које је већ креиран чвор, нови чвор се не креира (тиме је задовољено правило спајања у дефиницији 13). Поступак се рекурзивно понавља, док се не дође до кофактора који представљају константе, односно 1 и 0.

Овај поступак се примењује за релативно мало n, али је ефикаснији у односу на алгоритам Редукција одоздо на горе (видети одељак 3.1.2).

Пример 10. Нека је дата Булова функција:

$$f(x_1, \dots, x_4) = x_2 \wedge (x_3 \vee \neg x_4) \ \vee \ \neg x_1 \wedge \neg x_2 \wedge x_4 \ \vee \ x_1 \wedge \neg x_2 \wedge \neg x_4, \tag{19}$$

и нека је редослед променљивих задат са $\pi = x_1 < x_2 < x_3 < x_4$, а приоритет оператора је задат са $\{\neg, \wedge, \vee\}$, почев од оператора са највећим приоритетом, у опадајућем редоследу. Слика 10 приказује редослед у ком су креирани чворови приступом одозго на доле.

Поступак почиње конструисањем корена BDD коме се додељује променљива x_1 , као прва променљива у редоследу. Потом се израчунавају кофактори у односу на променљиву x_1, f_{x_1} и $f_{\neg x_1}$:

$$\begin{aligned} &f_{x_1} = \neg x_2 \wedge \neg x_4 \vee x_2 \wedge (x_3 \vee \neg x_4), \text{ и} \\ &f_{\neg x_1} = \neg x_2 \wedge x_4 \vee x_2 \wedge (x_3 \vee \neg x_4), \end{aligned} \tag{20}$$

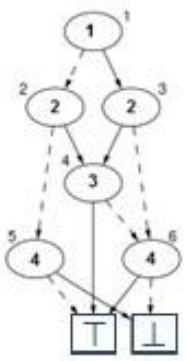
Како за ова два кофактора не постоје чворови у BDD, за сваки од њих се креира чвор којем се придружује променљива x_2, (јер оба кофактора зависе од те променљиве), а то је прва следећа променљива у редоследу. У наредном кораку, рачунају се кофактори у односу на променљиву x_2:

$$\begin{aligned} &f_{\neg x_1 x_2} = x_3 \vee \neg x_4 = \ f_{x_1 x_2} \\ &f_{x_1 \neg x_2} = \neg x_4, \\ &f_{\neg x_1 \neg x_2} = x_4 \end{aligned} \tag{21}$$

Рачун (21) доводи до три кофактора за које још увек не постоје чворови у до тада креираном BDD, па се за њих креирају чворови који су на слици 10 обележени редом идентификаторима 4, 5, и 6. Први кофактор зависи од променљиве x_3, па се рачунају кофактори у односу на променљиву x_3:

$$\begin{aligned} &f_{x_1 x_2 x_3} = f_{\neg x_1 x_2 x_3} = 1, \\ &f_{x_1 x_2 \neg x_3} = f_{\neg x_1 x_2 \neg x_3} = f_{x_1 \neg x_2 x_3} = f_{x_1 \neg x_2 \neg x_3} = \neg x_4, \\ &f_{\neg x_1 \neg x_2 x_3} = f_{\neg x_1 \neg x_2 \neg x_3} = x_4. \end{aligned} \tag{22}$$

У последњем кораку, сви кофактори су константне функције, па се директно представљају завршним чворовима са одговарајућом истинитосном вредношћу.



Слика 10: BDD функције $f(x_1, \dots, x_4)$ из примера 10

Овај алгоритам прихвата произвољан редослед променљивих, али је потребно да Булова функција буде задата изразом. У том смислу, овај алгоритам представља уопштење првог алгоритма за конструкцију BDD из Булове функције (ако се сматра да је истинитосна табела функције у ствари израз у савршеној дисјунктивној нормалној форми), за произвољан фиксиран редослед променљивих.

3.2. Алгоритми засновани на BDD

У овом одељку су представљени алгоритми за ефикасно решавање практичних проблема употребом BDD. Следе алгоритми који се односе на решења Булове једначине $f(x_1, \dots, x_n) = 1$:

- Израчунавање укупног броја решења (видети одељак 3.2.1)
- Генерисање скупа свих решења (видети одељак 3.2.2)
- Бирање једног случајног решења међу свим решењима са униформном расподелом вероватноћа (видети одељак 3.2.3)
- Израчунавање генератрисе скупа решења (видети одељак 3.2.4)
- Израчунавање полинома поузданости (видети одељак 3.2.5)
- Решавање проблема линеарног Буловог програмирања (видети одељак 3.2.6).

За сваки алгоритам се претпоставља природан редослед променљивих Булове функције, и да је BDD у канонској форми (према дефиницији 8).

3.2.1. Број решења Булове једначине

Једна од могућих примена BDD је израчунавање укупног броја решења Булове једначине $f(x_1, \dots, x_n) = 1$, као и генерисање сваког таквог решења. Решење једначине $f(x_1, \dots, x_n) = 1$ у BDD Булове функције $f(x_1, \dots, x_n)$ представља пут од корена до завршног чвора $\boxed{\top}$, на ком променљиве $x_1, \dots, x_n$ узимају истинитосне вредности 0 или 1, а укупан број решења те једначине представља број јединица у корену BDD ([1], стр. 207).

Другим речима, број јединица у перли чвора v коме је придружена променљива x_i, $1 \leq i \leq n$, у BDD представља број решења једначине $f(x_1, \dots, x_n) = 1$, у односу на природан редослед променљивих и фиксиране вредности променљивих $x_1, \dots, x_{i-1}$.

Алгоритам за израчунавање укупног броја решења се заснива на приступу одоздо на горе, од

завршних чворова према корену. Број решења у сваком чвору се рачуна као збир броја решења (односно броја јединица у перли) у левом и десном сину чвора, узимајући у обзир и разлику у нивоима између чвора и његових синова.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD.

Algoritam Broj_rešenja (bdd)

Улаз: bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$
Излаз: bdd са бројем решења c једначине $f(x_1, \dots, x_n) = 1$ за сваки чвор v у bdd, $v.c$ (c је цео број који представља број јединица у перли сваког чвора v у BDD).

begin

Иницијално се поставља број решења у завршним чворовима $\boxed{\perp}$ и $\boxed{\top}$
$\boxed{\perp}.c = 0$
$\boxed{\top}.c = 1$

Пролазак кроз све унутрашње чворове у BDD, од завршних чворова према корену

for each čvor v **in** bdd **do**

$l = v.levi\_sin$ # леви син текућег чвора v
$h = v.desni\_sin$ # десни син текућег чвора v

$l.indeks$ представља индекс променљиве чвора l, односно ниво у bdd на ком се
чвор l налази; слично важи за $h.indeks$

$l.c$ односно $h.c$ представљају редом број решења у левом односно десном сину
текућег чвора v

Број решења c у чвору v
$v.c = 2\text{^}(l.indeks - v.indeks - 1) * l.c + 2\text{^}(h.indeks - v.indeks - 1) * h.c$

return bdd sa brojem rešenja c u svakom čvoru v

end

Доказ коректности

Коректност алгоритма се доказује индукцијом уназад по нивоу BDD, почев од нивоа $n + 1$, где је n број променљивих Булове функције. Претпоставка је да је разлика између нивоа сваког оца и сина једнака 1, ради једноставности доказа.

База индукције: На нивоу $n + 1$ у BDD постоје тачно два завршна чвора $\boxed{\perp}$ и $\boxed{\top}$, чије су перле редом 0 и 1. Перла чвора $\boxed{\top}$ има једну јединицу, односно у том чвору постоји једно решење Булове једначине.

Индуктивна хипотеза. За сваки чвор v на нивоу већем од k у BDD важи да је број c њему придружен једнак броју решења једначине у чвору v, односно броју јединица у перли која одговара чвору v.

Корак индукције: Нека је дат чвор u на нивоу k, и нека су v, w редом леви и десни син чвора u. Према одељку 2.3, број јединица у чвору u једнак је збиру броја јединица у перлама синова v и w. Како је број јединица у синовима v и w према индуктивној хипотези једнак броју решења у сваком од та два чвора, то важи да је и њихов збир (број јединица у перли чвора оца u) једнак укупном броју решења у чвору u.

■

Пример 11. На слици 5(б) је приказан BDD Булове функције већинског одлучивања са три променљиве из примера 6.

Алгоритам иницијално поставља број решења у завршним чворовима $\boxed{\bot}$ и $\boxed{\top}$ на 0 и 1. Број јединица у чвору са перлом (01) је једнак 1, и представља збир броја решења у завршним чворовима који су његови синови. Овде се перле чвора пишу у загради због означавања чвора и приступања његовим атрибутима, у складу са псеудокодом. Број решења у чворовима са перлама (0001) и (0111) се израчунава, редом:

$$(0001).c = 2^{(0).indeks-(0001).indeks-1} * (0).c + 2^{(01).indeks-(0001).indeks-1} * (01).c = 2^{3-2-1} = 1, \text{ и}$$
$$(0111).c = 2^{(01).indeks-(0111).indeks-1} * (01).c + 2^{(1).indeks-(0111).indeks-1} * (1).c =$$
$$= 2^{3-2-1} * 1 + 2^{4-2-1} * 1 = 1 + 2 = 3 \quad (23)$$

Види се да је израчунат број решења једнак броју јединица у перли чвора. У завршном кораку, рачуна се број решења за корен, који је у овом примеру чвор са перлом (00010111) (истинитосном табелом Булове функције). Дакле, укупан број решења у корену BDD из примера 6 је:

$$(00010111).c =$$
$$= 2^{(0001).indeks-(00010111).indeks-1} * (0001).c + 2^{(0111).indeks-(000101111).indeks-1} * (0111).c =$$
$$= 2^{2-1-1} * 1 + 2^{2-1-1} * 3 = 1 + 3 = 4 \quad (24)$$

Сложеност

Алгоритам извршава $B(f)$ операција над n-битним бројевима, при чему је $B(f)$ укупан број чворова у BDD. Дакле, у најгорем случају сложеност алгоритма је $O(n * B(f))$.

Овај алгоритам има многобројне примене. На пример, ако се овај алгоритам примени на BDD који представља независне скупове циклуса са n променљивих C_n, тада је израчунат број решења у корену BDD управо укупан број независних скупова циклуса C_n (видети одељке 3.4 и 6.4).

3.2.2. Сва решења Булове једначине

Овај алгоритам враћа сва решења Булове једначине $f(x_1, \dots, x_n) = 1$, из сваког чвора до завршног чвора $\boxed{\top}$. Алгоритам конструише сва решења од текућег чвора v до завршног чвора $\boxed{\top}$ помоћу свих решења израчунатих у његовом левом и десном сину, на основу наредних правила:

1) Скуп свих решења $v.sva\_rešenja$ у текућем чвору v је иницијално празан;

2) Сва решења левог сина чвора v додају се у скуп $v.sva\_rešenja$, при чему се сваком таквом решењу дописује 0 на почетак решења, односно $0x \dots x$, ако је разлика нивоа између левог сина и чвора v већа од један. Овде је број "x"-ева који се дописује једнак разлици нивоа - 1.

3) Сва решења десног сина чвора v додају се у скуп $v.sva\_rešenja$, при чему се сваком таквом

решењу дописује 1 на почетак решења, односно $1x \dots x$, ако је разлика нивоа између десног сина и чвора v већа од један. Овде је број "x"-ева који се дописује једнак разлици нивоа - 1.

У правилима 2) и 3), разлика нивоа текућег чвора и његовог сина означава да не постоји чвор коме је придружена променљива са индексом који одговара нивоу између њих, односно тај ниво је "прескочен". Из тога следи да таква променљива у решењу може имати било коју истинитосну вредност, 0 или 1. Број "прескочених" нивоа је број карактера "x" који се дописују, и он је једнак разлици нивоа текућег чвора и његовог сина - 1.

Сва решења Булове једначине се налазе у корену BDD.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD. Претпоставља се да је сваком чвору v у BDD придружен број решења (број јединица) у том чвору, $v.c$.

Algoritam Sva_rešenja (bdd)

Улаз: bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$
Излаз: bdd са свим решењима једначине $f(x_1, \dots, x_n) = 1$ од сваког чвора v у bdd до завршног чвора $\boxed{\top}$, при чему се решење представља у облику ниске која садржи 0, 1, и/или x.

begin

Иницијално се постављају сва решења у завршним чворовима $\boxed{\bot}$ и $\boxed{\top}$
$\boxed{\bot}.sva\_rešenja = []$
$\boxed{\top}.sva\_rešenja = []$

# Пролазак кроз све унутрашње чворове у BDD, од завршних чворова према корену
for each čvor v **in** bdd **do**

$v.sva\_rešenja = []$
$l = v.levi\_sin$ # леви син текућег чвора v
$h = v.desni\_sin$ # десни син текућег чвора v

$l.indeks$ представља индекс променљиве чвора l, односно ниво у bdd на ком се
налази чвор l; слично важи за $h.indeks$

$l.sva\_rešenja$ односно $h.sva\_rešenja$ представљају редом сва решења у левом
односно десном сину текућег чвора v

# Ако је леви или десни син чвор $\boxed{\bot}$, тада је скуп решења празан
if $l == \boxed{\bot}$ **then**
$l.sva\_rešenja = []$

if $h == \boxed{\bot}$ **then**
$h.sva\_rešenja = []$

Сва решења $sva\_rešenja$ у чвору v
$v.sva\_rešenja = 0 \mid x\text{^}(l.indeks - v.indeks - 1) \mid (l.sva\_rešenja)$
unija $1 \mid x\text{^}(h.indeks - v.indeks - 1) \mid (h.sva\_rešenja)$

return bdd sa skupom svih rešenja $sva\_rešenja$ u svakom čvoru v

end

Операција "|" у псеудокоду алгоритма представља конкатенацију решења (ниски којима се решења представљају) левог и десног сина, према правилима 1) - 3). Израз $x\text{^}(l.indeks - v.indeks - 1)$ израчунава број "прескочених" нивоа између текућег чвора v и његовог сина, и надовезује "x" у решењу на место променљивих чији индекс одговара "прескоченом" нивоу. Операција "^" овде означава колико пута се "x" понавља (на пример, $x\text{^}2$ је једнако xx).

Доказ коректности

Доказ коректности је аналоган доказу претходног алгоритма из одељка 3.2.1.

■

Пример 12. На слици 5(б) је приказан BDD Булове функције већинског одлучивања са три променљиве из примера 6.

Алгоритам полази од завршних чворова и поставља $\boxed{\bot}.sva\_rešenja = []$ и $\boxed{\top}.sva\_rešenja = []$.

У чвору $\boxed{\bot}$ не постоји ниједно решење; зато је скуп решења тог чвора празан.

Алгоритам прелази на ниво који одговара највећем индексу променљиве, ниво 3. На том нивоу постоји само један чвор, чија је перла (01). Сва решења из чвора (01) до чвора $\boxed{\top}$ су:

$$(01).sva\_rešenja = 0 \mid x\text{^}(\boxed{\bot}.indeks - (01).indeks - 1) \mid (\boxed{\bot}.sva\_rešenja)$$
$$\textbf{unija}\ 1 \mid x\text{^}\left(\boxed{\top}.indeks - (01).indeks - 1\right) \mid \left(\boxed{\top}.sva\_rešenja\right) =$$
$$= [] \ \textbf{unija}\ 1 \mid x\text{^}0 \mid [] = [1] \qquad (25)$$

Алгоритам прелази на чворове на нивоу 2. На том нивоу се налазе чворови чије су перле редом (0001) и (0111). Решења се рачунају за сваки чвор:

$$(0001).sva\_rešenja = 0 \mid x\text{^}(\boxed{\bot}.indeks - (0001).indeks - 1) \mid (\boxed{\bot}.sva\_rešenja)$$
$$\textbf{unija}\ 1 \mid x\text{^}\left((01).indeks - (0001).indeks - 1\right) \mid \left((01).sva\_rešenja\right) =$$
$$= [] \ \textbf{unija}\ 1 \mid x\text{^}0 \mid [1] = [11]$$

$$(0111).sva\_rešenja = 0 \mid x\text{^}((01).indeks - (0111).indeks - 1) \mid ((01).sva\_rešenja)$$
$$\textbf{unija}\ 1 \mid x\text{^}\left(\boxed{\top}.indeks - (0111).indeks - 1\right) \mid \left(\boxed{\top}.sva\_rešenja\right) =$$
$$= [01] \ \textbf{unija}\ [1x] = [01, 1x] \qquad (26)$$

Алгоритам прелази на ниво 1, где се налази корен. Сва решења израчуната у корену су уједно и сва решења Булове једначине.

$$(00010111).sva\_rešenja = 0 \mid x\text{^}((0001).indeks - (00010111).indeks - 1) \mid ((0001).sva\_rešenja)$$
$$\textbf{unija}\ 1 \mid x\text{^}\left((0111).indeks - (00010111).indeks - 1\right) \mid \left((0111).sva\_rešenja\right) =$$
$$= 0 \mid [11] \ \textbf{unija}\ 1 \mid [01, 1x] = [011, 101, 11x]. \qquad (27)$$

Сложеност

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD, и нека је N број решења једначине $f(x_1, \dots, x_n) = 1$. Тада је временска сложеност алгоритма $O(n * N)$, јер сваки унутрашњи чвор BDD у најгорем случају води до бар једног решења, а свако решење исписује n бита.

3.2.3. Случајни избор решења Булове једначине

Овај алгоритам генерише случајно решење Булове једначине $f(x_1, \dots, x_n) = 1$ са униформном расподелом вероватноће ([1], стр. 208).

Пре позива овог алгоритма, потребно је да се примени алгоритам Број решења једначине $f(x_1, \dots, x_n) = 1$ на BDD Булове функције (видети одељак 3.2.1), како би се за сваки чвор у BDD израчунало колико има јединица у том чвору. Број јединица у сваком чвору се користи да би се обезбедило да свако решење има једнаку вероватноћу избора.

Нека су чворови v, w редом леви и десни син чвора u, и нека је број јединица у левом сину једнак p, а број јединица у десном сину једнак q. Алгоритам полази од корена BDD, који је увек почетни чвор случајног решења, и генерише случајан број

$$rand \in [1, p * 2\text{^}(u.indeks - v.indeks - 1) + q * 2\text{^}(u.indeks - w.indeks - 1)] \quad (28)$$

где је $u.indeks$ ниво чвора u, односно индекс променљиве која је придружена чвору u. Уколико је случајан број $rand$ мањи или једнак $p * 2\text{^}(u.indeks - v.indeks - 1)$, бира се леви син као наредни чвор у случајном решењу; у супротном се бира десни син. Процес се наставља од изабраног сина док се не дође до завршног чвора $\boxed{\top}$.

Случајно решење поред 0 и 1 садржи и m "x"-ева, који се уписују када је разлика између нивоа чвора оца и чвора сина већа од 1. Појава карактера "x" у случајном решењу значи да променљива, чији индекс одговара нивоу који је "прескочен", може имати било коју истинитосну вредност у том решењу. Стога се свако "x" независно замењује са 0 или 1, чиме се добија потпуно случајно решење. На пример, може се дефинисати да, ако је случајан број мањи од $1/2$, бира се 0, у супротном се бира 1.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD, и нека је сваком чвору v у BDD придружен број решења, $v.c$.

Algoritam Uniformno_slučajno_rešenje $(robdd)$

Улаз: bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$, са израчунатим бројем решења c у сваком чвору v

Излаз: Низ вредности променљивих $x_1, \dots, x_n$, који представљају решење једначине $f(x_1, \dots, x_n) = 1$ изабрано на случајан начин, са униформном расподелом вероватноћа

begin

neka je v čvor u bdd
$v = bdd.koren$

while v != $\boxed{\top}$ **do**

$rand$ = slučajan broj u intervalu $[1,\ v.levi\_sin.c * 2\text{^}(v.levi\_sin.indeks - v.indeks - 1) +$
$v.desni\_sin.c * 2\text{^}(v.desni\_sin.indeks - v.indeks - 1))]$

if $rand <= v.levi\_sin.c$ **then**
$v.resenje = 0|x\text{^}(v.levi\_sin.indeks - v.indeks - 1)|(v.levi\_sin.resenje)$
$v = v.levi\_sin$

else

$v.resenje = 1|x\text{^}(v.desni\_sin.indeks - v.indeks - 1)|(v.desni\_sin.resenje)$
$v = v.desni\_sin$

svako x se nezavisno zamenjuje sa 0 ili 1 sa verovatnoćom $1/2$ u $v.rešenje$

return $v.rešenje$

end

Доказ коректности

Коректност алгоритма се доказује индукцијом по броју решења у чвору BDD.

База индукције: Завршни чвор $\boxed{\top}$ има једну јединицу, па важи да тај чвор има једно решење.

Индуктивна хипотеза: Нека је чвор v на нивоу мањем или једнаком k, и нека има укупно $v.c$ јединица ($v.c \leq C$, где је C укупан број решења једначине који се налази у корену). Важи да алгоритам даје свако од ових решења са једнаком вероватноћом $1/v.c$.

Корак индукције: Нека је чвор v на нивоу мањем или једнаком k и нека су му синови u, w на нивоима већим од k. Према индуктивној хипотези је за u и w изабрано по једно решење са униформном расподелом вероватноћа. Према алгоритму, прво од њих се бира са вероватноћом

$$p = u.c * 2\text{^}(u.indeks - v.indeks - 1)/(u.c * 2\text{^}(u.indeks - v.indeks - 1) + w.c * 2\text{^}(w.indeks - v.indeks - 1)) \quad (29)$$

Ако је разлика нивоа између чвора v и чвора који је изабран између u и w већа од 1, вредности сваке недефинисане променљиве (означене са "x") замењују се на случајан начин са 0 или 1, са једнаком вероватноћом $1/2$.

На тај начин је постигнуто да је свако решење у чвору v изабрано са једнаком вероватноћом $1/v.c$.

■

Пример 13. Нека је задат BDD Булове функције већинског одлучивања са три променљиве, из примера 6 (видети слику 5(б)).

Корен BDD је иницијално први чвор случајног решења. Нека је у првој итерацији изабран случајан број 4 из интервала $[1, 1+3]$. Како је случајан број већи од броја јединица у левом сину корена (чвор са перлом 0001), бира се десни син корена (чвор са перлом 0111), односно $x_1 = 1$ јер је разлика нивоа између корена и десног сина једнака 1, па нема додатних "x"-ева.

Нека је у наредној итерацији изабран случајан број 2 из интервала $[1, 2]$. Алгоритам бира десног сина за следећи чвор у решењу, односно $x_2 = 1$. Како је овде десни син завршни чвор $\boxed{\top}$, разлика између нивоа је једнака 2>1, па се на место променљиве x_3 уписује "x", што означава да x_3 може бити 0 или 1 са једнаком вероватноћом. Решење креирано на овај начин је $x_1x_2x_3 = 11x$.

На сличан начин се бира и случајна истинитосна вредност 0 или 1 за променљиву x_3. Нека је изабран случајан број 0.3 из интервала $[0, 1]$. Како је $0.3 < 0.5$, бира се истинитосна вредност 0, па је потпуно случајно решење са униформном расподелом вероватноће $x_1x_2x_3 = 110$.

Сложеност

Сложеност алгоритма зависи од броја нивоа у BDD. У најгорем случају, између свака два нивоа је разлика највише 1, па је временска сложеност алгоритма $O(n)$, где је n број променљивих задате Булове функције.

3.2.4. Генератриса скупа решења Булове једначине са различитим бројем решења

Генератриса скупа решења Булове једначине $f(x_1, \dots, x_n) = 1$ Булове функције f је функција која приказује колико има различитих решења са различитим бројем јединица. Другим речима, генератриса приказује колико има различитих решења са k јединица међу вредностима променљивих $x_1, x_2, \dots, x_n$, за свако $0 \leq k \leq n$. Наредна дефиниција даје прецизнији опис генератрисе Булове једначине $f(x_1, \dots, x_n) = 1$ ([1], вежба 25).

Дефиниција 15. Нека је задата Булова функција $f(x_1, \dots, x_n)$. Генератриса скупа решења једначине $f(x_1, \dots, x_n) = 1$ је функција

$$G(z) = \sum_{x_1=0}^{1} \dots \sum_{x_n=0}^{1} z^{x_1+x_2+\dots+x_n} f(x_1, \dots, x_n) \tag{30}$$

односно, *полином* степена n са ненегативним коефицијентима a_j:

$$G(z) = a_0 + a_1 z + \dots + a_n z^n, \quad \text{за } 0 \leq j \leq n \tag{31}$$

где је сваки коефицијент a_j број решења са j јединица, односно, постоји a_j решења $f(x_1, \dots, x_n) = 1$ таквих да је

$$x_1 + x_2 + \dots + x_n = j. \tag{32}$$

Алгоритам за креирање генератрисе користи приступ одоздо на горе. При израчунавању генератрисе одређеног унутрашњег чвора, алгоритам полази од већ израчунатих генератриса у левом и десном сину тог чвора. Генератриса у сваком чвору BDD садржи број решења са различитим бројем јединица од тог чвора до завршног чвора $\boxed{\top}$. Генератриса Булове функције се налази у корену BDD.

Уочава се да, како је чвор ближи завршним чворовима, то његова генератриса има мање a_j коефицијената. Свака генератриса унутрашњег чвора даје тачан број решења са одговарајућим бројем јединица, од тог чвора до завршног чвора $\boxed{\top}$.

Пре креирања генератрисе, неопходно је да се израчуна број решења (односно број јединица) у сваком чвору BDD, Дакле, прво се позива алгоритам Број решења једначине $f(x_1, \dots, x_n) = 1$ (видети одељак 3.2.1), а потом алгоритам који креира генератрису за сваки чвор BDD.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD.

Algoritam Generatrisa_rešenja (bdd)

Улаз: bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$
Излаз: bdd са листом $generatrisa$ за сваки чвор - то је листа коефицијената генератрисе a_j, $[a_0, a_1, \dots, a_n]$, за $0 \leq j \leq n$, задатог BDD Булове функције

begin

Почетни коефицијенти генератрисе у завршним чворовима
$\boxed{\bot}.a = [0]$
$\boxed{\top}.a = [1]$

$generatrisa = []$

# Пролазак кроз све унутрашње чворове у BDD, од завршних чворова према корену
for each čvor v **in** bdd **do**

$l = v.levi\_sin$ # леви син текућег чвора v
$h = v.desni\_sin$ # десни син текућег чвора v

$l.indeks$ представља индекс променљиве чвора l (односно његов ниво у BDD)
слично важи за $h.indeks$

$l.a$ представља генератрису чвора l; слично важи за $h.a$

Генератриса чвора v је садржана у $v.a$
$v.a = (1+z)^{l.indeks - v.indeks - 1} * l.a + (1+z)^{h.indeks - v.indeks - 1} * z * h.a$
dodati $v.a$ u niz $generatrisa\ [v.indeks]$

return bdd sa generatrisom $v.a$ za svaki čvor v

end

Доказ коректности

Коректност алгоритма се доказује индукцијом.

База индукције: Коефицијенти генератрисе у завршним чворовима $\boxed{\bot}$ и $\boxed{\top}$ су редом 0 и 1.

Индуктивна хипотеза: За сваки чвор v на нивоу већем од k, алгоритам тачно израчунава његову генератрису $v.a$

Корак индукције: Нека је w чвор на нивоу мањем или једнаком k, при чему су његови синови редом чворови u и v на нивоима већем од k. Алгоритам израчунава полином $w.a$ према изразу:

$$w.a = [a_0, a_1, \dots, a_{n-k}] = (1+z)^{u.indeks - w.indeks - 1} * u.a + (1+z)^{v.indeks - w.indeks - 1} * z * v.a \quad (33)$$

Израз $(1+z)^m$, где је $m = v.indeks - w.indeks - 1$ (односно $m = u.indeks - w.indeks - 1$)) одговара променљивама са индексима између $w.indeks$ и $v.indeks$, које могу да имају као вредност произвољну комбинацију нула и јединица, а од тих комбинација (укупно $\binom{m}{k}$) имају у себи тачно k јединица, $k = 0, 1, \dots, m$. Из тога следи да је полином $w.a$ генератриса решења за чвор w.

■

Пример 14. Нека је задат BDD Булове функције већинског одлучивања са три променљиве, из примера 6 (видети слику 5(б)).

Нека је генератриса чвора представљена листом својих коефицијената a_j , $[a_0, a_1, \dots, a_n]$, за $0 \le j \le n$. Иницијално, генератрисе завршних чворова $\boxed{\bot}$ и $\boxed{\top}$ су редом $\boxed{\bot}.a = [0]$ и $\boxed{\top}.a = [1]$.

Рачунају се коефицијенти генератрисе чвора са перлом (01), $(01).a$:

$$(01).a = (1+\mathrm{z})^{\boxed{\perp}.indeks-(01).indeks-1} * \boxed{\perp}.a + (1+\mathrm{z})^{\boxed{\top}.indeks-(01).indeks-1} z * \boxed{\top}.a =$$
$$= (1+\mathrm{z})^{4-3-1} * [0] + (1+\mathrm{z})^{4-3-1} z * [1] = (1+\mathrm{z})^{0} z * [1] = z = [0,1] \quad (34)$$

Прелази се на ниво за један ближи корену, и рачунају се коефицијенти генератриса за чворове са перлама, редом (0001) и (0111):

$$(0001).a = (1+\mathrm{z})^{\boxed{\perp}.indeks-(0001).indeks-1} * \boxed{\perp}.a + (1+\mathrm{z})^{(01).indeks-(0001).indeks-1} z * (01).a =$$
$$= (1+\mathrm{z})^{4-2-1} * [0] + (1+\mathrm{z})^{3-2-1} z * [0,1] =$$
$$= (1+\mathrm{z})^{0} z * [0,1] = z * (0+z) = z^2 = [0,0,1] \quad (35)$$

$$(0111).a = (1+\mathrm{z})^{(01).indeks-(0111).indeks-1} * (01).a + (1+\mathrm{z})^{\boxed{\top}.indeks-(0111).indeks-1} z * \boxed{\top}.a =$$
$$= (1+\mathrm{z})^{3-2-1} * [0,1] + (1+\mathrm{z})^{4-2-1} z * [1] = (1+\mathrm{z})^{0} * [0,1] + (1+\mathrm{z})^{1} z * [1] =$$
$$= [0,1] + (z+z^2) * [1] = z + z + z^2 = 2z + z^2 = [0,2,1] \quad (36)$$

Коначно, генератриса корена и задате Булове функције је

$$(00010111).a = (1+\mathrm{z})^{(0001).indeks-(00010111).indeks-1} * (0001).a +$$
$$+ (1+\mathrm{z})^{(0111).indeks-(00010111).indeks-1} z * (0111).a =$$
$$= (1+\mathrm{z})^{2-1-1} * [0,0,1] + (1+\mathrm{z})^{2-1-1} z * [0,2,1] =$$
$$= (1+\mathrm{z})^{0} * [0,0,1] + (1+\mathrm{z})^{0} \mathrm{z} * [0,2,1] =$$
$$= [0,0,1] + \mathrm{z} * [0,2,1] = z^2 + \mathrm{z} * (2z + z^2) = 3z^2 + z^3 = [0,0,3,1]. \quad (37)$$

Генератриса $[0,0,3,1]$ у корену задатог BDD има следећу интерпретацију:

- постоји 0 решења са нула јединица,
- постоји 0 решења са једном јединицом,
- постоји 3 решења са две јединице, и
- постоји 1 решење са три јединице.

Сложеност

Ако се генератриса Булове функције $f(x_1, \dots, x_n)$ посматра као полином степена n, тада сложеност алгоритма зависи од множења листи (са највише $n+1$ коефицијената) два полинома степена n (односно множења коефицијената генератриса левог и десног сина сваког чвора у BDD), и броја чворова у BDD.

Сложеност једноставне имплементације је $O(B(f) * (n+1)^2)$, али се за велико n сложеност може побољшати на $O(B(f) * ((n+1)\, log(n+1))$, помоћу Брзе Фуријеове трансформације за множење полинома ([5], стр. 214).

3.2.5. Полином поузданости Булове функције

Булова функција се може представити и помоћу полинома, који је њена мултилинеарна репрезентација дата у дефиницији 16.

Дефиниција 16. Мултилинеарна репрезентација Булове функције $f(x_1, x_2, \dots, x_n)$ се дефинише полиномом $F(x_1, \dots, x_n)$:

$$F(x_1, \dots, x_n) = (1-x_1) F_{x_1=0}(x_2, \dots, x_n) + x_1 F_{x_1=1}(x_2, \dots, x_n) \quad (38)$$

где су $F(1) = 1, F(0) = 0,$ и $F_{x_1=0}$ и $F_{x_1=1}$ целобројне мултилинеарне репрезентације $f(0, x_2, \dots, x_n)$ и $f(1, x_2, \dots, x_n)$.

Теорема 5. Свака Булова функција $f(x_1, \dots, x_n)$ се може јединствено представити помоћу *полинома* са целобројним коефицијентима, $F(x_1, \dots, x_n)$, који има следећа својства:

- $F(x_1, \dots, x_n) = f(x_1, \dots, x_n)$, кад год је свако $x_j = 0$ или $x_j = 1$
- $F(x_1, \dots, x_n)$ је мултилинеаран израз у који свака променљива x_j улази са степеном највише 1, за све $j = 1, 2, \dots, n$.

Дефиниција 17. Полином поузданости $F(p_1, \dots, p_n)$ Булове функције $f(x_1, \dots, x_n)$ представља вероватноћу да је $f(x_1, \dots, x_n) = 1$ ако је дата вероватноћа p_i да је променљива x_i једнака 1, $1 \le i \le n$.

Тврђење 4. Полином поузданости $F(p_1, \dots, p_n)$ Булове функције $f(x_1, \dots, x_n)$ задовољава рекурентну релацију (36).

Последица тврђења 4 је да је полином поузданости Булове функције исто што и њена мултилинеарна репрезентација.

Алгоритам у наставку израчунава вредност полинома поузданости Булове функције на основу BDD Булове функције и низа вероватноћа p_i, задатих за сваку променљиву x_i, $1 \le i \le n$. Вероватноћа p_i је вероватноћа да је $x_i = 1$ у решењу функције $f(x_1, \dots, x_n) = 1$.

Полином поузданости се израчунава помоћу рекурзивне релације (38) за сваки чвор BDD, приступом од завршних чворова према корену ([1], стр. 208).

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD.

Algoritam Polinom_pouzdanosti(bdd, P)

Улаз:
- bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$
- низ P - низ вероватноћа $p[indeks]$ да променљива x_{indeks} има вредност 1, $1 \le i \le n$

Излаз: bdd, са вероватноћом $v.p$ решења функције $f(x_1, \dots, x_n) = 1$ за сваки чвор v у bdd.

begin

```
# Иницијално се постављају вероватноће у завршним чворовима ⊥ и ⊤
⊥.p = 0
⊤.p = 1

# Пролазак кроз све унутрашње чворове у BDD, од завршних чворова према корену

for each čvor v in bdd do

    l = v.levi_sin   # леви син текућег чвора v
    h = v.desni_sin  # десни син текућег чвора v

    # l.p представља вероватноћу да чвор l има истинитосну вредност 1
    # слично важи за h.p
```

$\#\ v.p$ је вероватноћа да текући чвор v има истинитосну вредност 1
$v.p = (1 - p[indeks]) * l.p + p[indeks] * h.p$

return bdd sa verovatnoćom $v.p$ da čvor v ima istinitosnu vrednost 1, za svaki čvor v

end

Доказ коректности

Коректност алгоритма се доказује индукцијом.

База индукције: Вероватноће у завршним чворовима $\boxed{\bot}$ и $\boxed{\top}$ су редом 0 и 1.

Индуктивна хипотеза: За сваки чвор v на нивоу већем од k, алгоритам тачно израчунава вероватноћу да v има истинитосну вредност 1.

Корак индукције: Нека је w чвор на нивоу мањем или једнаком k, при чему су његови синови редом чворови u и v на нивоима већим од k. Алгоритам израчунава вероватноћу да w има истинитосну вредност 1 према изразу:

$$w.p = (1 - p[indeks]) * u.p + p[indeks] * v.p, \tag{39}$$

где је $p[indeks]$ вероватноћа да променљива x_{indeks} има вредност 1, $1 \leq i \leq n$.

Нека је $(1 - p[indeks]) * u.p = a$ и $p[indeks] * v.p = b$. Вредност израза a се рачуна помоћу вероватноће $u.p$, тј. да леви син u чвора w има истинитосну вредност 1. Тада се узима вероватноћа да чвор w има истинитосну вредност 0, тј. $1 - p[indeks]$.

На сличан начин важи да се вредност израза b рачуна помоћу вероватноће $v.p$, тј. да десни син v чвора w има истинитосну вредност 1. Тада се узима вероватноћа да чвор w има истинитосну вредност 1, тј. $p[indeks]$.

Збир вредности $a + b$ представља вероватноћу да чвор w има истинитосну вредност 1 у зависности од вероватноћа својих синова и вероватноће $p[indeks]$ променљиве којој је чвор w додељен, па следи да важи индуктивна хипотеза.

■

Пример 15. Нека је задат BDD Булове функције $f(x_1, \dots, x_n)$ већинског одлучивања са три променљиве, из примера 6 (видети слику 5(б)), и нека све променљиве x_i имају вредност 1 са вероватноћом 0.5, $1 \leq i \leq n$.

Иницијално, постављају се вероватноће завршних чворова $\boxed{\bot}.p = 0$ и $\boxed{\top}.p = 1$.

Рачуна се вероватноћа да чвор са перлом (01) има истинитосну вредност 1 у решењу $f(x_1, \dots, x_n) = 1$:

$$(01).p = (1 - p[3]) * \boxed{\bot}.p + p[3] * \boxed{\top}.p = (1 - 0.5) * 0 + 0.5 * 1 = 0.5 \tag{40}$$

Даље, рачунају се вредности полинома поузданости за чворове са перлама редом, (0001) и (0111):

$$(0001).p = (1 - p[2]) * \boxed{\bot}.p + p[2] * (01).p = (1 - 0.5) * 0 + 0.5 * 0.5 = 0.2 \tag{41}$$

$$(0111).p = (1 - p[2]) * (01).p + p[2] * \boxed{\top}.p = (1 - 0.5) * 0.5 + 0.5 * 1 = 0.75 \tag{42}$$

Коначно, вероватноћа да је $f(x_1, x_2, x_3) = 1$ када све променљиве имају истинитосну вредност 1 у решењу са вероватноћом 0.5, износи

$$(00010111).p = (1 - p[1]) * (0001).p + p[1] * (0111).p \\ = (1 - 0.5) * 0.25 + 0.5 * 0.75 = 0.5 \quad (43)$$

Сложеност

Како се полином поузданости израчунава за сваки чвор, неопходно је проћи све чворове датог BDD, чиме се долази до сложености $O(B(f))$, где је $B(f)$ број чворова у BDD.

Постоје разне примене израчунавања полинома поузданости одређене Булове функције. На пример, нека је дат BDD независних скупова циклуса са n променљивих, C_n (видети одељак 3.4). Вредност полинома поузданости у корену BDD је вероватноћа да је случајан подскуп тих чворова независан, ако су задате вероватноће укључивања појединих чворова циклуса у скуп.

3.2.6. Линеарно Булово програмирање

Проблем линеарног Буловог програмирања ([1], стр. 209) се дефинише на следећи начин.

Дефиниција 18 (Проблем линеарног Буловог програмирања). Нека је задата Булова функција $f(x_1, \dots, x_n)$ и низ константи $w = (w_1, \dots, w_n)$. Одредити низ $x = (x_1, \dots, x_n)$ за који

$$w_1 x_1 + \cdots + w_n x_n \quad (44)$$

има максималну вредност, при услову да је $f(x_1, \dots, x_n) = 1$.

Дефиниција 19. Израз $w_1 x_1 + \cdots + w_n x_n$ из дефиниције 18 зове се тежина низа $(x_1, \dots, x_n)$.

Алгоритам за решавање проблема линеарног Буловог програмирања проналази решење једначине $f(x_1, \dots, x_n) = 1$ највеће тежине, разматрајући сваки чвор у правцу од завршних чворова према корену. Идеја иза овог приступа је да се оптимално тежинско решење може наћи за било коју перлу (чвор у BDD), за коју су позната оптимална тежинска решења њене леве и десне (под)перле.

Псеудокод

Нека је задата Булова функција $f(x_1, \dots, x_n)$ својим BDD, и нека је сваком чвору v у BDD додељен јединствен целобројни идентификатор, $v.id$.

Алгоритам користи помоћне низове m и t, дужине једнаке броју чворова у BDD, и низ W дужине за један веће од броја променљивих. Низ W служи за прелазе између несуседних нивоа, низ m за чување већих тежина, а низ t за чување информације који син текућег чвора v (леви или десни) има већу тежину.

Algoritam Rešenje_maksimalne_težine (bdd, w)

Улаз:
- bdd - редукован и уређен BDD функције $f(x_1, \dots, x_n)$,
- низ тежина $w = (w_1, \dots, w_n)$ за сваку променљиву $(x_1, \dots, x_n)$

Излаз: низ $rešenje = x_1, \dots, x_n$ - решење једначине $f(x_1, \dots, x_n) = 1$ са максималном тежином

begin

Иницијализација помоћних низова

Низ m се користи за записивање максималних вредности тежине у чворовима
$m = []$

Низ t се користи за записивање вредности променљивих за које се достижу максималне
вредности тежине у чворовима
$t = []$
$rešenje = []$

Низ W се користи за прелазе између несуседних нивоа
$W_{n+1} = 0$

for $j = n$ **downto** 1 **do**
$W_j = W_{j+1} + max(w_j, 0)$

Привремене локалне променљиве за израчунавање вредности тежине чвора помоћу
редом, левог и десног сина
$tmp\_m\_l = 0$
$tmp\_m\_h = 0$

Обрађују се унутрашњи чворови, од завршних чворова према корену

for each čvor v **in** bdd **do**

$l = v.levi\_sin$ # леви син текућег чвора v
$h = v.desni\_sin$ # десни син текућег чвора v

$v.indeks$ представља индекс променљиве која је придружена текућем чвору v
слично важи за $l.indeks$ и $h.indeks$

# Рачуна се тежина чвора v помоћу тежине левог сина
if $l != \boxed{\perp}$ **then**
$tmp\_m\_l = m[l.id] + W_{v.indeks+1} - W_{l.indeks}$

# Рачуна се тежина чвора v помоћу тежине десног сина
if $h != \boxed{\perp}$ **then**
$tmp\_m\_h = m[h.id] + W_{v.indeks+1} - W_{v_h} + w_{v.indeks}$

Подразумеван приступ је да се бира леви син, и тада се поставља вредност 0 у
низу t; за десног сина се поставља вредност 1. Аналогно, подразумевана тежина
која највише доприноси тежини текућег чвора v је тежина левог сина
$tmp\_m = tmp\_m\_l$
$tmp\_t = 0$

```
        # Ако је десни син "тежи", он се бира у решење
        if l == ⊥ or tmp_m_h > tmp_m_l then
            tmp_m = tmp_m_h
            tmp_t = 1

        m[v.id] = tmp_m
        t[v.id] = tmp_t

    # Одређивање решења x, користећи низове w и t, полазећи од корена и крећући се наниже
    # у BDD

    u = bdd.koren
    indeks = 1
    rešenje = ""

    # Обилази се ниво по ниво
    while indeks < n + 1:

        # Ако је текући индекс мањи од индекса текуће променљиве, тада је разлика
        # између нивоа већа од 1, и тада се истинитосна вредност променљиве бира на
        # основу њене тежине
        while indeks < u.indeks − 1 do

            indeks += 1
            if w[indeks]>0 then rešenje += "1" else rešenje += "0"

        # У текуће решење се уписује вредност променљиве која се налази у низу t, и у
        # зависности од те вредности иде се левим, односно десним сином, док се не
        # стигне до чвора ⊤

        if u != ⊤ then
            indeks += 1
            rešenje+= t[u.id]

            if t[u.id] then
                u = u.levi_sin
            else
                u = u.desni_sin

    return rešenje

end
```

Рачунање тежине чвора v помоћу тежине левог и десног сина, редом l и h се своди на

$$m[v.id] = max(m[l.id],\ m[h.id] + w_{v.indeks}), \tag{45}$$

када су $l.indeks$ и $h.indeks$ најчешће истовремено једнаки $v.indeks + 1$ (односно, када је разлика између нивоа чвора и његових синова једнака 1).

Доказ коректности

База индукције: Завршни чвор ⊤ има једну јединицу, па важи да тај чвор има једно решење.

Индуктивна хипотеза: Нека је чвор v на нивоу већем или једнаком k, и нека важи да алгоритам проналази решење максималне тежине до тог чвора.

Корак индукције: Нека је чвор v на нивоу мањем од k и нека су му синови u, z на нивоима већем или једнаким k. За сваки чвор u и z су према индуктивној хипотези израчуната решења максималне тежине. Према алгоритму, тежина чвора v је једнака

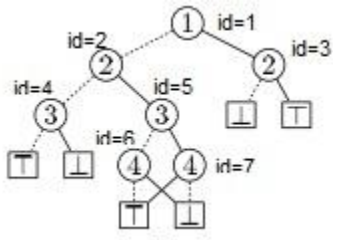
$$m_v = max(m_u, m_z + w_{v.indeks}) \tag{46}$$

Овде m_u означава тежину левог сина u чвора v. Ако се у решењу максималне тежине иде левим сином чвора u, то значи да је то уједно и тежина чвора v, јер променљива x_{indeks} која је придружена чвору v тада добија вредност 0 у решењу (односно, чвор v није део решења). Ако се у решењу максималне тежине иде десним сином чвора v, то значи да променљива која је придружена чвору v добија вредност 1 у решењу (односно, чвор v јесте део решења), и тада се за чвор v рачуна нова тежина помоћу израза $m_z + w_{v.indeks}$, где је тежина $w_{v.indeks}$ задата тежина променљиве.

Како се за резултујућу тежину чвора v бира максимум тежина његових синова, а у решење се бира онај син чија је тежина изабрана, следи да је решење у чвору v решење максималне тежине.

■

Пример 16. Нека је дата Булова функција чије је истинитосна табела 1100100100001111, за коју је изграђен BDD (видети пример 8 и слику 7) са јединственим идентификатором за сваки чвор:



Слика 11: BDD из примера 16

и нека су дате тежине сваке променљиве (x_1, x_2, x_3, x_4), редом $w = (1, -2, -3, 4)$.

Прво се креира помоћни низ $W = (W_1, \dots, W_5) = (5, 4, 4, 4, 0)$, и иницијализују се помоћни низови $t = []$ и $m = []$, а потом се рачунају тежине сваког унутрашњег чвора, редом по нивоима, од завршних чворова према корену.

За сваки чвор се иницијално поставља да се узима леви син: $tmp\_t = 0$.

Прво се обрађују чворови на ниву 4 који су придружени променљивој x_4.

Обрада чвора чији је идентификатор једнак 6, $v_{\mathrm{id}=6}$: како леви син води до чвора $\boxed{\top}$, рачуна се његова тежина помоћу левог сина:

$$m_{\mathrm{id}=6} = \ tmp\_m\_l = m_{l.id} + W_{v.indeks+1} - W_{l.indeks} = 0 + 0 - 0 = 0 \tag{47}$$

Како је десни син једнак $\boxed{\bot}$, тежина чвора помоћу десног сина се не рачуна, па су коначне вредности чвора $v_{\mathrm{id}=6}$ редом $m_{\mathrm{id}=6} = 0,\ t_{\mathrm{id}=6} = 0$.

Обрада чвора чији је идентификатор једнак 7, $v_{\mathrm{id}=7}$: како леви син води до чвора $\boxed{\bot}$, тежина чвора помоћу левог сина се не рачуна. Како је десни син једнак $\boxed{\top}$, тежина чвора помоћу десног сина износи:

$$m_{\mathrm{id}=7} = tmp\_m\_h = m_{h.id} + W_{v.indeks+1} - W_{h.indeks} + w_{v.indeks} = \\ = m_{\boxed{\top}.id} + W_5 - W_5 + w_4 = 0 + 4 = 4 \tag{48}$$

Коначне вредности чвора $v_{\mathrm{id}=7}$ су редом $m_{\mathrm{id}=7} = 4,\ t_{\mathrm{id}=7} = 1.$

Прелази се на чворове на нивоу 3.

Обрада чвора чији је идентификатор једнак 4, $v_{\mathrm{id}=4}$: тежина чвора се рачуна на сличан начин као за чвор $v_{\mathrm{id}=6}$. Коначне вредности су:

$$m_{\mathrm{id}=4} = m_l + W_{v_{\check{\mathrm{c}}\mathrm{vor}+1}} - W_{v_l} = m_{\boxed{\top}} + W_4 - W_5 = 0 + 4 - 0 = 4, \\ t_{\mathrm{id}=4} = 0 \tag{49}$$

Обрада чвора чији је идентификатор једнак 5, $v_{\mathrm{id}=5}$: како су и леви и десни син различити од чвора $\boxed{\bot}$, рачунају се тежине помоћу оба сина:

помоћу левог сина: $l\_m_{\mathrm{id}=5} = m_l + W_{v.indeks+1} - W_{l.indeks} = m_{\mathrm{id}=6} + W_4 - W_4 = 0 + 4 - 4 = 0$ (50)

помоћу десног сина: $h\_m_{\mathrm{id}=5} = m_h + W_{v.indeks+1} - W_{h.indeks} + w_{v.indeks} = \\ = m_{\mathrm{id}=7} + W_4 - W_4 + w_3 = 4 + 4 - 4 - 3 = 1$ (51)

Како је $max(l\_m_{\mathrm{id}=5}, h\_m_{\mathrm{id}=5}) = max(0,1) = 1$, бира се десни син, и коначне вредности су: $m_{\mathrm{id}=5} = 1,\ t_{\mathrm{id}=5} = 1$.

Сличним поступком се рачунају вредности и за чворове $v_{\mathrm{id}=2},\ v_{\mathrm{id}=3}$ и $v_{\mathrm{id}=1}$ у наведеном редоследу, и износе редом:

$$m_{\mathrm{id}=2} = 4,\ t_{\mathrm{id}=2} = 0, \\ m_{\mathrm{id}=3} = 2,\ t_{\mathrm{id}=3} = 1, \\ m_{\mathrm{id}=1} = 4,\ t_{\mathrm{id}=1} = 0 \tag{52}$$

Након израчунавања низова t и m, BDD се обилази од корена наниже, да би се реконструисало решење са највећом тежином, помоћу вредности у низу t.

Вредности елемената низа t [$t_{\mathrm{id}=1}$, $t_{\mathrm{id}=2}$, $t_{\mathrm{id}=3}$, $t_{\mathrm{id}=4}$, $t_{\mathrm{id}=5}$, $t_{\mathrm{id}=6}$, $t_{\mathrm{id}=7}$] су редом $[0,0,1,0,1,0,0,1]$. Корену одговара вредност $t_{\mathrm{id}=1} = 0$, што значи да се од корена иде левим сином у решењу, односно да променљива x_1 има вредност 0. Леви син корена је чвор $v_{\mathrm{id}=2}$, коме одговара вредност $t_{\mathrm{id}=2} = 0$, што значи да се поново иде левим сином до чвора $v_{\mathrm{id}=4}$, а променљива x_2 има вредност 0. Чвору $v_{\mathrm{id}=4}$ одговара вредност $t_{\mathrm{id}=4} = 0$, па променљива x_3 има вредност 0, а левим сином се од чвора $v_{\mathrm{id}=4}$ стиже до завршног чвора $\boxed{\top}$. Како је разлика нивоа између чвора $v_{\mathrm{id}=4}$ и $\boxed{\top}$ већа од 1, то значи да је ниво који одговара променљивој x_4 "прескочен", односно не постоји чвор између њих којем би променљива x_4 била придружена. У овом случају, одлука о вредности променљиве x_4 се доноси на основу прослеђених тежина сваке променљиве: ако је w_4 веће од нуле, променљивој x_4 се додељује вредност 1, а у супротном се додељује 0. Како је $w_4 = 4$, тада је $x_4 = 1$, па је решење максималне тежине 0001. Тежина решења одговара вредности $m_{\mathrm{id}=1}$ у корену, и једнака је 4.

Сложеност

Алгоритам се састоји из два дела: у првом делу се пролази кроз сваки чвор BDD, па извршавање захтева $O(B(f))$ корака. У другом делу се обрађује низ који садржи чворове највећих тежина -

читају се чворови који улазе у решење максималне тежине, као и њихове истинитосне вредности, што захтева додатних $O(n)$ корака. Дакле, сложеност алгоритма је $O(n + B(f))$.

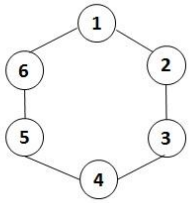
3.3. Независни скупови циклуса са n чворова

У овом одељку се демонстрира употреба BDD за представљање независних скупова циклуса C_n са n чворова помоћу BDD.

Дефиниција 20. Независни скуп графа $G = (V, E)$ је подскуп $U \subseteq V$, такав да за било која два чвора $u, v \in U$ у G не постоји грана (u, v).

Нека C_n означава циклус са n чворова. Ако су $v_1, \dots, v_n$ чворови циклуса, онда циклус има n грана: $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n), (v_n, v_1)$. *Независни скупови циклуса* су скупови чворова у којима не постоји грана која повезује два суседна чвора.

Пример 17. На слици 12 је приказан циклус C_6:



Слика 12: Циклус C_6 из примера 17

Ниска $x_1 \dots x_6 = 000000$ одговара празном скупу, ниска $x_1 \dots x_6 = 001010$ одговара скупу $\{3, 5\}$. Оба подскупа су неки од независних скупова чворова по дефиницији 20, док ниска $x_1 \dots x_6 = 011010$ одговара скупу $\{2, 3, 5\}$, али овај скуп није независан скуп чворова.

Теорема 6. Нека је Булова функција $f(x_1, \dots, x_n)$ задата условом да је $f(x_1, \dots, x_n) = 1$ ако и само ако је подскуп чворова циклуса C_n са индексима једнаким индексима i променљивих x_i , $1 \le i \le n$, за које је $x_i = 1$ независан скуп. Тада је број решења једначине $f(x_1, \dots, x_n) = 1$ једнак броју независних скупова у C_n.

Доказ. Доказ се заснива на чињеници да постоји једнозначно пресликавање између решења једначине $f(x_1, \dots, x_n) = 1$ и скупа независних скупова циклуса C_n. Пресликавање придружује n-торки $(x_1, \dots, x_n)$ скуп чворова са оним индексима k за које је $x_k = 1,\ k = 1, \dots, n$.

■

Следи теорема 7 која је последица теореме 6.

Теорема 7. BDD који представља Булову функцију $f(x_1, \dots, x_n)$ из теореме 6 испуњава следеће услове:

1) Пут од корена BDD до завршног чвора $\boxed{1}$ је такав да не постоје три чвора u, v, w којима су придружене узаступне променљиве, таква да је

$$v = high(u), w = high(v) \tag{53}$$

2) Пут од корена BDD који почиње десном граном корена се не сме завршити у чвору

$\boxed{\top}$ (јер је тада $x_n = x_1 = 1$, па одговарајући скуп није независан).

Другим речима, у низу $x = (x_1, \dots, x_n)$ не постоје две узастопне јединице (узастопним се сматрају и (x_1, x_n) ако и само ако је $f(x_1, \dots, x_n) = 1$).

Опис алгоритма

Алгоритам изграђује BDD који представља независне скупове циклуса C_n почев од корена према завршним чворовима, примењујући правила из теореме 7. У наставку су дата правила директне изградње овог BDD, која обезбеђују да у сваком кораку алгоритма важе редукованост и уређеност BDD, односно да је BDD у канонској форми након извршавања сваког корака алгоритма.

1) (*Правило корена BDD*.) Прво се изграђује леви подграф корена, који одговара случају када је $x_1 = 0$, а потом десни подграф корена, када је $x_1 = 1$ (видети услов 2) из теореме 7).

2) (*Правило првих нивоа*.) Нови чворови се додају на ниво из опсега $[1, 3)$ у леви подграф, односно на ниво из опсега $[1, 4)$ у десни подграф при сваком гранању, осим када је нови чвор десни син оца; тада се за његовог десног сина не креира нови чвор, већ се преспаја на завршни чвор $\boxed{\bot}$ (видети услов 1) из теореме 7).

3) (*Правило обрасца левог сина*.) За сваки нови чвор који се додаје на ниво из $[3, n-2]$ у леви подграф, односно који се додаје на ниво из $[4, n-3]$ у десни подграф, важи следећи образац:

- ако је v десни син чвора оца u, тада је

$$v.desni\_sin = \boxed{\bot}, \text{ и } v.levi\_sin = u.levi\_sin.levi\_sin \tag{54}$$

- ако је v леви син чвора оца u, тада су леви и десни син чвора v нови чворови у BDD.

4) (*Правило последњих нивоа*.) За чворове који се додају на последње нивое BDD важе посебна правила, која омогућују представљање гране (x_n, x_1) у BDD, тако да важи теорема 7, и каноничност BDD.

(Леви подграф, ниво $n-1$.) Чворови који се додају у леви подграф на ниво $n-1$ су редом:

- чвор који је леви син оца са нивоа $n-2$
  - леви син је чвор $\boxed{\top}$
  - десни син је нови чвор који припада нивоу n
- чвор који је десни син оца са нивоа $n-2$
  - леви син је чвор $\boxed{\top}$
  - десни син је чвор $\boxed{\bot}$

(Леви подграф, ниво n.) Чвор који се додаје на последњи ниво n (током додавања чворова у леви подграф на ниво $n-1$) је једини чвор на том нивоу. На њега се везују чворови са нижих нивоа из левог и десног подграфа. Његови синови су редом:

- леви син је чвор $\boxed{\bot}$
- десни син је чвор $\boxed{\top}$

(Десни подграф, ниво $n-2$.) Чворови који се додају у десни подграф на ниво $n-2$ су редом:

- чвор који је леви син оца са нивоа $n-3$
  - леви син је чвор на последњем нивоу n (додат током изградње левог подграфа)
  - десни син је нови чвор који припада нивоу $n-1$

- чвор који је десни син оца са нивоа $n-2$
  - леви син је чвор на последњем нивоу n (додат током изградње левог подграфа)
  - десни син је чвор $\boxed{\perp}$

(Десни подграф, ниво $n-1$.) Чвор који се додаје на ниво $n-1$ је једини чвор на том нивоу у десном подграфу. Његови синови су редом:
  - леви син је чвор на последњем нивоу n (додат током изградње левог подграфа)
  - десни син је чвор $\boxed{\perp}$

Следе дефиниције које се односе на израчунавање броја решења једначине $f(x_1, \dots, x_n) = 1$ ако је познат број променљивих n.

Дефиниција 21. *Фибоначијев број* F_n је цео број за који важи рекурентна релација

$$F_n = F_{n-1} + F_{n-2}, F_0 = 0, F_1 = 1, \tag{55}$$

Фибоначијеви бројеви $F_0, F_1, F_2, \dots, F_{n-1}, F_n$ чине Фибоначијев низ, у коме је сваки елемент низа једнак збиру претходна два елемента.

Дефиниција 22. *Лукасов број* L_n је цео број за који важи рекурентна релација

$$L_n = L_{n-1} + L_{n-2}, L_0 = 2, L_1 = 1 \quad , \tag{56}$$

Лукасови бројеви $L_0, L_1, \dots, L_{n-1}, L_n$ чине Лукасов низ, у коме је сваки елемент низа једнак збиру претходна два елемента.

Лукасов број L_n може изразити помоћу Фибоначијевих бројева F_{n+1} и F_{n-1}:

$$L_n = F_{n+1} + F_{n-1} \tag{57}$$

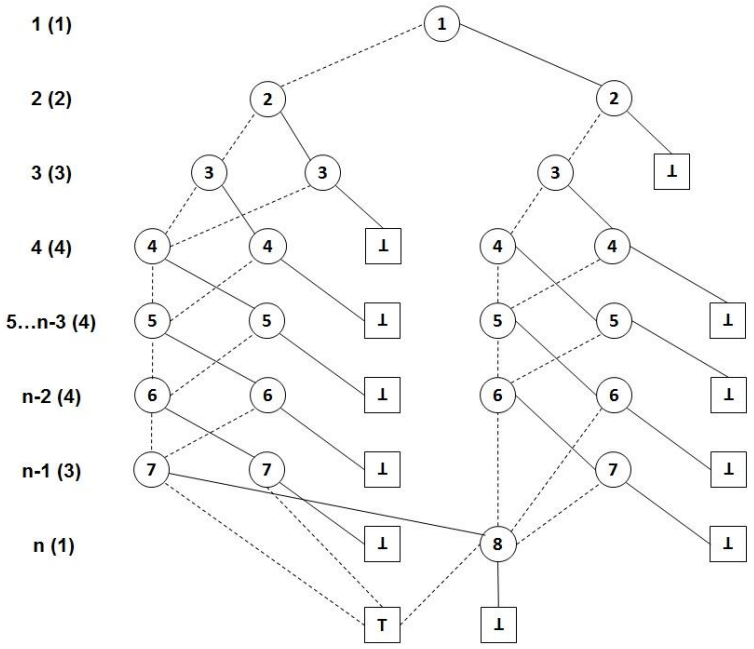
Табела 1 приказује првих 11 чланова Фибоначијевог и Лукасовог низа.

n	F_n	L_n
0	0	2
1	1	1
2	1	3
3	2	4
4	3	7
5	5	11
6	8	18
7	13	29
8	21	47
9	34	76
10	55	123

Табела 1: Чланови Фибоначијевог и Лукасовог низа за $n = 10$

Може се показати да је број решења једначине $f(x_1, \dots, x_n) = 1$ једнак Лукасовом броју L_n ([1], вежба 15(а)). У поглављу 6 је приказана провера ове чињенице за $n \leq 30$.

Пример 18. На слици 13 је приказан BDD независних скупова циклуса C_8.



Слика 13: BDD независних скупова циклуса C_8

Види се примена правила изградње BDD и образац описан у правилу 2) који прате унутрашњи чворови. Број чворова по нивоима BDD из примера 18 је приказан у загради поред броја нивоа лево од BDD:

- нивои $[1, 4]$ имају редом 1, 2, 3, и 4 чвора
- нивои $[5,\ n-3]$ имају по 4 чвора
- ниво $n-2$ има 4 чвора
- ниво $n-1$ има 3 чвора
- ниво n има 1 чвор.

Сложеност

Сложеност алгоритма за конструкцију BDD који представља независне скупове циклуса C_n зависи од броја чворова n у циклусу. Из обрасца по ком се чворови додају (видети теорему 7 и правила у делу *Опис алгоритма*), сваки наредни ниво добија по 4 чвора, почев од нивоа 4 (односно променљиве x_4), закључно са нивоом $n-2$ (односно са променљивом x_{n-2}). Нивои 1, 2, 3 и 4 имају редом 1, 2, 3 и 4 чвора, док последња два нивоа имају редом укупно 3 чвора и 1 чвор.

Укупан број чворова $B(f)$ у BDD са завршним чворовима за $n = 3$ је једнак 6, а за $n > 3$ је једнак

$$B(f) = 10 + 4(n-3-5+1) + 8 + 2 = 20 + 4n - 28 = 4n - 8 \qquad (58)$$

Из наведеног следи да је сложеност алгоритма линеарна по n, па је сложеност алгоритма за конструкцију BDD $O(n)$. Међутим, сложеност примене алгоритма за бројање решења (видети одељак 3.2.1) је експоненцијална, јер број L_n независних скупова у C_n расте експоненцијално са n.

4. Утицај редоследа променљивих на комплексност BDD

У овом поглављу се представља утицај редоследа променљивих Булове функције на комплексност њеног BDD. Следе две главне теореме ([2], стр. 123).

Теорема 8. Нека је S_j скуп потфункција f који се добија доделом истинитосних вредности променљивама $x_1, \dots, x_{j-1}$ и које суштински зависе од x_j. BDD Булове функције f за задат редослед променљивих $x_1, \dots, x_n$ има тачно $|S_j|$ чворова које означава променљива x_j.

Доказ теореме 8 може се наћи у раду [2], стр. 95-96 и 123.

Теорема 9 (Последица теореме 8). Нека је $(i_1, \dots, i_n)$ пермутација скупа $\{1, \dots, n\}$, и нека је P редукован уређен BDD Булове функције f, за задат редослед променљивих $x_{i_1}, \dots, x_{i_n}$. Додатно, нека је S_{i_j} скуп потфункција функције f који се добија доделом истинитосних вредности променљивама $x_{i_1}, \dots, x_{i_{j-1}}$ и које суштински зависе од x_{i_j}. Тада BDD P има тачно $|S_{i_j}|$ чворова означених са x_{i_j} .

4.1. Веза између редоследа променљивих и величине BDD

Величина BDD, па самим тим и комплексност манипулације BDD, директно зависи од претпостављеног редоследа променљивих ([1-4]). Следи неколико екстремних примера, где се број чворова у BDD значајно мења променом редоследа променљивих.

Пример 19. Нека је дата Булова функција:

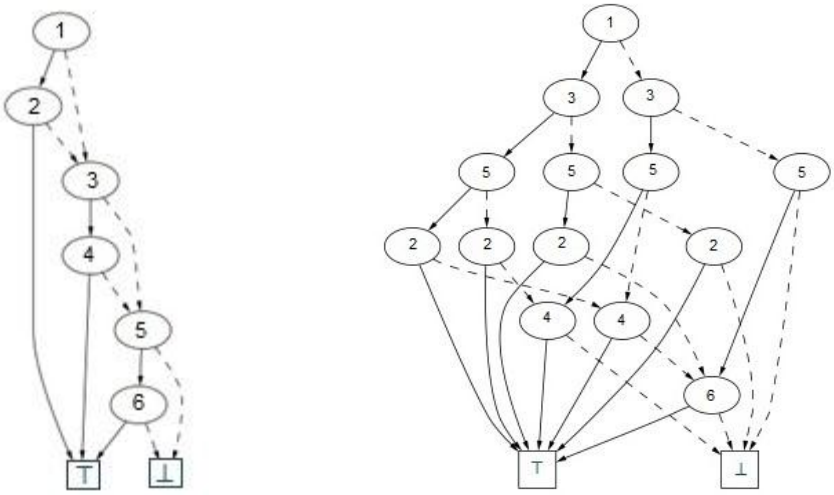
$$f(x_1, \dots, x_{2n}) = x_1x_2 + x_3x_4 + \cdots + x_{2n-1}x_{2n}. \tag{59}$$

Природан редослед. За задат редослед променљивих $x_1, x_2, \dots, x_{2n-1}, x_{2n}$, BDD садржи тачно $2n + 2$ чворова, па комплексност линеарно расте у зависности од броја променљивих n. Слика 14(а) приказује BDD када је $n = 3$. Овај BDD има компактну структуру јер за свако $k \in \{1, 2, \dots n-1\}$ након читања додељених истинитосних вредности променљивама $x_1, x_2, \dots, , x_{2k}$, постоје две могућности:

а) Услед доделе истинитосних вредности променљивама $x_1, x_2, \dots, , x_{2k}$, већ је познато да је истинитосна вредност функције f једнака 1 (односно тачно).
б) Претпоставка а) не важи (није познато да је истинитосна вредност функције f тачно). Тада се вредност функције може одредити користећи само истинитосне вредности преосталих $x_{2k+1}, x_{2k+2}, \dots, , x_{2n}$.

Другим речима, за свако k и произвољну доделу вредности променљивама $x_1, x_2, \dots, , x_{k-1}$, увек постоји тачно једна потфункција $f_k(x_k, \dots, x_{2n})$ која суштински зависи од x_k. Ова функција је:

$$f_k = \begin{cases} x_kx_{k+1} + x_{k+2}x_{k+3} + \cdots + x_{2n-1}x_{2n}, & \text{када је } k \text{ непарно} \\ x_k + x_{k+1}x_{k+2} + \cdots + x_{2n-1}x_{2n}, & \text{када је } k \text{ парно} \end{cases} \tag{60}$$



(а) Редослед променљивих $x_1, x_2, x_3, x_4, x_5, x_6$

(б) Редослед променљивих $x_1, x_3, x_5, x_2, x_4, x_6$

Слика 14: Булова функција $f(x_1, \dots, x_n) = x_1x_2 + x_3x_4 + x_5x_6$ у примеру 19

Из теореме 8 следи да BDD ове функције на сваком нивоу има тачно један чвор.

"Непаран-паран" редослед. За редослед променљивих $x_1, x_3, \dots, x_{2n-1}, x_2, x_4, \dots, x_{2n}$, BDD је знатно комплекснији, што илуструје пример на слици 14(б). Наредна анализа се ослања на теореме 8 и 9.

Нека је $k \leq n$. Постоји 2^{k-1} различитих константних вектора $(a_1, a_3, \dots, a_{2k-3}) \in \{0,1\}^n$ који се могу доделити $k-1$ променљивама $x_1, x_3, \dots, x_{2k-3}$. Ови вектори воде до потфункција:

$$\begin{gathered} f(x_1, \dots, x_n)|_{x_1=a_1, x_3=a_3, \dots, x_{2k-3}=a_{2k-3}} = \\ = a_1x_2 + a_3x_4 + \cdots + a_{2k-3}x_{2k-2} + a_{2k-1}x_{2k} + \\ + x_{2k+1}x_{2k+2} + \cdots + x_{2n-1}x_{2n} \end{gathered} \tag{61}$$

Из (61) следи:

а) Свака од 2^{k-1} потфункција $f(x_1, \dots, x_n)|_{x_1=a_1, x_3=a_3, \dots, x_{2k-3}=a_{2k-3}}$ суштински зависи од x_{2k-1}, k- те променљиве у редоследу.
б) Све ове потфункције су у паровима различите.

Дакле, у случају да је $k \leq n$, BDD функције f има тачно 2^{k-1} чворова који су означени променљивом x_k. Аналогно се може проверити да, у случају да је $k > n$, постоји тачно 2^{2n-k} чворова који су означени променљивом x_k. Укупан број чворова је:

$$B(f) = 2 * 2\sum_{k=1}^{n} 2^{k-1} + 2 = 2 * (2^n - 1) + 2 = 2^{n+1} \tag{62}$$

Дакле, величина BDD функције f расте експоненцијално за овај распоред променљивих. Да би се доказао експоненцијалан раст BDD, довољно је доказати експоненцијалан раст броја чворова које означава одређена променљива x_i.

Следи интуитивно објашњење разлога за експоненцијални раст броја чворова у примеру 19. Насупрот природном редоследу, након читања вредности $x_1, x_3, \dots, x_k$ за k непарно у редоследу "непаран-паран", још увек се не може знати вредност функције у одређеним случајевима. За сваку доделу вредности првој променљивој, вредност функције f још увек није одређена. Обе вредности тачно и нетачно су једнако могуће уз одговарајућу доделу вредности осталим променљивама. Из тога следи да не постоји грана из ма које преостале променљиве која директно води до завршног

чвора. Комплексности BDD додатно доприноси чињеница и да за сваку од две доделе вредности првим променљивама $x_1, x_3, \dots, x_k$, увек постоји додела вредности преосталим променљивама таква да се вредности резултујуће функције разликују.

4.2. Проналажење оптималног редоследа променљивих Булове функције

Пре саме конструкције одговарајућег BDD Булове функције, редослед њених променљивих мора бити познат. Како постоје фамилије функција код којих величина BDD може бити и линеарне и експоненцијалне комплексности у зависности од задатог редоследа променљивих, проналажење оптималног редоследа је од кључног значаја за ефикасну манипулацију BDD.

Како је проблем проналажења најбољег редоследа променљивих NP-комплетан ([1], стр. 240), развили су се различити приступи за проналажење задовољавајућег редоследа променљивих ([1], стр. 239-249, и [2], стр. 145-170).

Детаљна обрада и имплементација алгоритама за оптимизацију редоследа променљивих није тема овог рада.

Претрага простора пермутација редоследа променљивих и величина BDD

Претрага простора пермутација представља примену свих могућих $n!$ редоследа променљивих на истинитосну табелу Булове функције, за коју се претпоставља природан редослед променљивих. Одељак 6.5 пружа увид како се мења величина BDD неколико примера Булових функција када се примене свих могућих $n!$ редоследа променљивих.

Тврђење 5. Истинитосна табела пермутације природног редоследа променљивих је и сама пермутација истинитосне табеле природног редоследа променљивих.

Из тврђења 5 следи опис алгоритма:

- Нека је дата истинитосна табела Булове функције $f(x_1, \dots, x_n)$, $f_0 f_1 \dots f_{2^n-1}$, за коју се претпоставља природан редослед променљивих $x_1, \dots, x_n$, и нека је дата пермутација природног редоследа променљивих, $x_{i_1}, \dots, x_{i_n}$.
- Израчунати пермутовану истинитосну табелу за пермутован редослед променљивих, према формули:

$$j_k = 2^{n-1} * x_{i_1} + 2^{n-2} * x_{i_2} + \cdots + 2^{n-j} * x_{i_j} + \cdots + 2^1 * x_{i_{n-1}} + x_{i_n}, \quad 0 \le k \le 2^n - 1 \tag{63}$$

  Овде k представља индекс у f_k, а $f_{j_1} f_{j_k} \dots f_{j_{2^n-1}}$ представља пермутовану истинитосну табелу истинитосне табеле $f_0 f_1 \dots f_{2^n-1}$.

- За пермутовану истинитосну табелу, креирати BDD и израчунати број његових чворова.
- Поступак поновити за сваку пермутацију редоследа променљивих, и упоредити промену броја чворова у BDD за различите редоследе.

Имплементација овог алгоритма налази се у одељку 6.5.

5. Класе функција са једноставним и сложеним BDD

У овом поглављу се разматрају различити типови функција, чија природа директно утиче на комплексност BDD који их представља. Условно се може говорити о две посебне класе – *пријатељске функције,* које имају *једноставан* BDD, и *комплексне функције*, које имају *сложен* BDD.

5.1. Функције са једноставним BDD

Пријатељске функције ([1], стр. 213, и [2], стр. 125) чине фамилију функција за које је познато да имају BDD прихватљиво мале величине, односно *једноставан* BDD.

5.1.1. Симетричне функције

Дефиниција 23. *Симетрична Булова функција* је Булова функција чија истинитосна вредност не зависи од пермутације њених променљивих,

Из дефиниције 23 следи да истинитосна вредност симетричне Булове функције зависи од укупног броја јединица у улазу. Другим речима, за симетричне функције сви редоследи променљивих су *еквивалентни* у смислу комплексности BDD.

Теорема 10. Величина BDD је *независна* од редоследа променљивих Булових симетричних функција.

Доказ. Нека је $f(x_1, x_2, \ldots, x_n)$ Булова симетрична функција. На основу теореме 9 закључује се да за свако k и доделу истинитосних вредности $k-1$ променљивама $x_{i_1}, \ldots, x_{i_{k-1}}$, једино је важно колико променљивих међу наведенима има истинитосну вредност тачно (односно 1). Пошто број јединица међу првим променљивама може да буде 0,1,..., или $k-1$, број чворова на нивоу k је увек ограничен са k. Дакле, потфункције су облика:

$$f_{x_{i_1}=a_{i_1},\ldots,x_{i_{k-1}}=a_{i_{k-1}}} \tag{64}$$

Као последица, важи да свака променљива x_k симетричне функције у BDD има највише k чворова којима је придружена, $k = 1,2,\ldots,n$. Сумирањем по k добија се да величина BDD симетричних функција у најгорем случају достиже $B(f) = O(n^2)$.

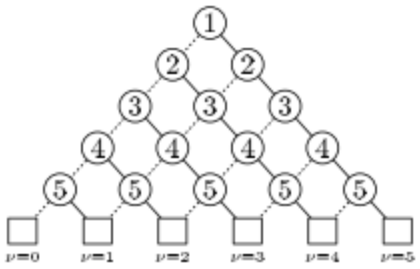
■

Пример 20. Нека је дата произвољна симетрична функција пет променљивих

$$f(x) = v(x_1 + \cdots + x_5), \tag{65}$$

где је $v: \{0,1,2,3,4,5\} \to \{1,0\}$ произвољно пресликавање.

Слика 15 приказује примену троугаоног обрасца изградње бинарног стабла ове функције.



Слика 15: Троугаони образац изградње стабла фунцкије из примера 20

у коме су листови постављени на ⊤ и ⊥, што зависи од вредности функције v. Из обрасца се види да постоје редундантни и еквивалентни чворови који се могу уклонити. Другим речима, BDD симетричне функције од n променљивих има највише $\binom{n+1}{2} + 2$ чворова.

Постоји неколико познатих потклаcа класе симетричних функција:

1) *Функције прага* - њихова вредност је 1 за улазне бинарне векторе са k или више јединица, за фиксирано k;
2) *Функције фиксних вредности* - њихова вредност је 1 за улазне бинарне векторе са тачно k јединица, за фиксирано k;
3) *Бројачке функције* - њихова вредност је 1 за улазне бинарне векторе са бројем јединица који је конгруентан са $k \bmod m$, за фиксирано k, m;
4) *Функције парности* - њихова вредност је 1 ако улазни бинарни вектор има непаран број јединица.

5.1.2. Функције прага

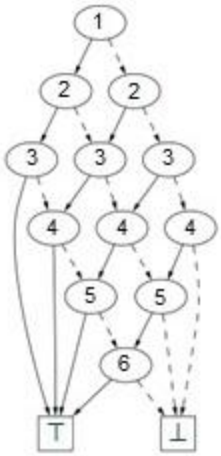
Дефиниција 24. Булова *функција прага* је свака функција облика:

$$T_k^n(x_1, \dots, x_n) = \begin{cases} 1, & \text{ако је } x_1 + \dots + x_n \geq k \\ 0, & \text{иначе} \end{cases} \tag{66}$$

Након фиксирања вредности произвољних i променљивих које садрже најмање k јединица, вредност функције је већ одређена. Нека је чвор v чвор који је достигнут након доделе вредности првим i променљивама које имају највише $k-1$ јединица. Чвор v садржи $high$ грану која води до завршног чвора ⊤. Из наведеног следи да за свако i постоји највише k чворова које означава променљива x_i.

Пример 21. Слика 16 у наставку приказује функцију већинског одлучивања са 6 променљивих за $k = 3$ (ова специјална функција за коју је $k = n/2$ зове се функција медијане). У овом примеру (по дефиницији 24), за две вредности i постоје тачно три чвора које означава x_i, и та три чвора садрже следеће информације:

- Чвор лево: До сад, тачно две јединице су фиксиране.
- Чвор у средини: До сад, тачно једна јединица је фиксирана.
- Чвор десно: До сад, ниједна јединица није фиксирана.



Слика 16: Функција већинског одлучивања T_3^6 из примера 21

Теорема 11. Нека је дата симетрична функција $f(x_1, \dots, x_n)$, и нека се функција g добија из функције f поистовећивањем x_{k+1} са x_k, тј. у којој су две суседне променљиве постављене као једнаке:

$$g(x_1, \dots, x_n) = f(x_1, \dots, x_{k-1}, x_k, x_k, x_{k+2}, \dots, x_n). \tag{67}$$

Тада за функцију g („контракцију“ функције f), важи да је $B(g) \leq B(f)$.

Доказ. Нека BDD функција f и g имају a_j и b_j чворова са ознаком j, редом, за $1 \leq j \leq n$. Свака подтабела f реда $n+1-k$ је облика $\alpha\beta\gamma\delta$, где су $\alpha, \beta, \gamma,$ и δ подтабеле реда $n-1-k$. Одговарајуће подтабеле функције g су $\alpha\alpha\delta\delta$, па су оне перле ако и само ако важи $\alpha \neq \delta$. У том случају, или је $\alpha\beta\gamma\delta$ перла, или је $\alpha\beta = \gamma\delta$ перла. Као последица добија се $b_k \leq a_k + a_{k+1}$, и $b_{k+1} = 0$. Такође је $b_j \leq a_j$, за $1 \leq j < k$, јер је свака перла у g реда већег од $n+1-k$ “скупљање” из најмање једне такве перле из f. Додатно, важи и $b_j \leq a_j$ за $j > k+1$, јер су подтабеле над $(x_{k+2}, \dots, x_n)$ идентичне, иако се можда не појављују у g.

■

Пример 22. Нека је дата симетрична функција $f(x_1, \dots, x_6)$, и нека је процесом *сажимања* добијена функција $f(x_1, x_1, x_3, x_3, x_3, x_6)$. Одговарајућа *функција прага* је $[2x_1 + 3x_3 + x_6 \geq t]$ има BDD величине $\leq \binom{6+1}{2} + 2$, за било коју вредност t, јер је она сажета верзија дате симетричне функције $f(x_1, \dots, x_6) = [x_1 + \cdots + x_6 \geq t]$, чији BDD има највише $\binom{6+1}{2} + 2$ чворова (видети пример 20). Понављањем поцеса сажимања, добија се да функција као што је $f(x_1, x_1, x_3, x_3, x_3, x_6)$ са поновљеним променљивама има мали BDD кад год је број $B(f)$ првобитне функције мали.

Из горе наведеног следи тврђење 6.

Тврђење 6. Било која функција прага, са ненегативним целобројним тежинама:

$$f(x_1, \dots, x_n) = [w_1x_1 + \cdots + w_nx_n \geq t] \tag{68}$$

може се добити сажимањем симетричне функције са $w_1 + w_2 + \cdots + w_n$ променљивих, тако да је величина њеног BDD $O(w_1 + w_2 + \cdots + w_n)^2$ ([1], стр. 213, и вежба 170).

5.2. Функције са сложеним BDD

Овај одељак описује функције које имају сложен BDD и концепт квази-профила Булове функције.

5.2.1. Комплексне функције

Постоје Булове функције за које је познато да имају BDD експоненцијалне величине, за ма који редослед променљивих. За процену најгорег случаја посматра се *профил* датог BDD ([1], стр. 233), који представља низ $(b_0, b_1, \dots, b_{n-1}, b_n)$, када постоји b_k унутрашњих чворова који се гранају на променљивој x_{k+1} , и b_n завршних чворова. Уочава се да важи:

$$B(f) = b_0 + b_1 + \cdots + b_{n-1} + b_n \tag{69}$$

Такође је $b_0 \leq 1$, $b_1 \leq 2$, $b_2 \leq 4$, $b_3 \leq 8$, а у општем случају важи

$$b_k \leq 2^k \tag{70}$$

јер сваки чвор има само две гране. Даље, $b_n = 2$ кад функција f није константна, и $b_{n-1} \leq 2$, јер постоје само два дозвољена избора за леву и десну грану чвора n. Познато је да је b_k број перли реда $n-k$ у истинитосној табели функције f, односно број различитих потфункција над променљивама $x_{k+1}, \dots, x_n$ које зависе од x_{k+1} , након што су вредности променљивих $x_1, \dots, x_k$ фиксиране. Број перли реда m је највише $2^{2^m} - 2^{2^{m-1}}$, па да мора да важи да је:

$$b_k \leq 2^{2^{n-k}} - 2^{2^{n-k-1}} \text{ , за } 0 \leq k \leq n \tag{71}$$

Пример 23. За $n = 11$, из (70) и (71) следи да је $(b_0, b_1, \dots, b_{11})$ највише

$$(1,\ 2,\ 4,\ 8,\ 16,\ 32,\ 64,\ 128,\ 240,\ 12,\ 2,\ 2) \tag{72}$$

Стога је укупан број чворова BDD $B(f) \leq 1 + 2 + \dots + 128 + 240 + \dots + 2 = 255 + 256 = 511$. Ова горња граница се добија помоћу истинитосне табеле

$$00000000\ 00000001\ 00000010\ \dots\ 11111110\ 11111111, \tag{73}$$

или помоћу било које ниске дужине 2^{11}, која представља пермутацију 256 могућих 8-битних бајтова (јер су све 8 -битне перле очигледно присутне, и због тога што су све подтабеле дужина $16, 32, \dots, 2^{11}$ очигледно перле). Ово се може непосредно проверити покретањем библиотеке из одељка 6.5, где се као улаз функције прослеђује истинитосна табела (73).

Дефиниција најгорег случаја је дата теоремом 12.

Теорема 12. За сваку Булова функцију $f(x_1, \dots, x_n)$ важи $B(f) \leq U_n$, где је

$$U_n = 2 + \sum_{k=0}^{n-1} min\left(2^k,\ 2^{2^{n-k}} - 2^{2^{n-k-1}}\right) = 2^{n-\lambda(n-\lambda n)} + 2^{2^{\lambda(n-\lambda n)}} - 1\ . \tag{74}$$

Штавише, функције f_n за које је $B(f_n) = U_n$ постоје за свако n.

Доказ теореме 12 се може наћи у књизи [1], стр. 234 и вежба 112.

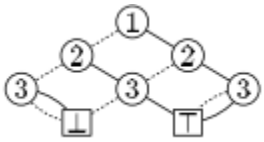
■

5.2.2. Квази-профил BDD

Концепт значајан за разматрање приликом одређивања комплексности BDD је тзв. "квази BDD" (енгл.*QDD*). Свака функција има јединствен QDD, који је сличан BDD осим у томе што корену увек одговара променљива најмањег индекса (x_1), и сваки чвор са индексом променљиве k, за $k < n$, се грана на два чвора са индексом променљиве $k+1$; стога је сваки пут од корена до завршног чвора дужине n.

Да би ово било могуће, дозвољава се да $low(v)$ и $high(v)$ синови чвора v у QDD буду идентични. QDD мора бити редукован, у смислу да различити чворови не смеју имати идентичне синове low и $high$.

Пример 24. QDD за Булову функцију већинског одлучивања са три променљиве (дате у примеру 1) је:



Слика 17: *QDD* Булове функције већинског одлучивања у примеру 24

Приказани QDD има 2 чвора више него њему одговарајући BDD на слици 1.

Дефиниција 25. *Квази-профил* функције је низ $(q_0, \ldots, q_{n-1}, q_n)$, где q_{k-1} представља број k чворова у QDD.

Тврђење 7. Број q_k је број различитих *подтабела* реда $n-k$ у истинитосној табели.

Уочава се аналогија са бројем b_k, који представља број различитих перли у BDD. Како је свака перла уједно и подтабела (одељак 2.3), важи:

$$q_k \geq b_k \text{, за } 0 \leq k \leq n. \tag{75}$$

Теорема 13. Важи следећа релација између броја различитих перли у BDD и елемената квази-профила QDD:

$$q_k \leq 1 + b_0 + \ldots + b_{k-1}, \text{ и } q_k \leq b_k + \ldots + b_n, \text{ за } 0 \leq k \leq n \tag{76}$$

Доказ. Нека је $N_k = b_0 + \ldots + b_{k-1}$ број чворова у BDD чија је ознака (индекс променљиве) j, за $j \leq k$. Збир улазних степена тих чворова је најмање N_k, а збир излазних степена је $2 * N_k$, са додатним показивачем на корен. Дакле, највише $N_k + 1$ грана може ићи са виших k нивоа до нижих нивоа. Свака подтабела реда $n-k$ одговара једној таквој грани. Стога је $q_k \leq N_k + 1$.

Додатно, мора да важи $q_k \leq b_k + \ldots + b_n$, јер свака подтабела реда $n-k$ одговара јединственој перли реда $\leq n-k$.

■

Теорема 14 (Последица теореме 13). Сваки елемент квази-профила одређује једну доњу границу величине BDD:

$$B(f) \geq 2q_k - 1 \text{, за } 0 \leq k \leq n. \tag{77}$$

Доказ. Нека је $Q(f) = q_0 + \ldots + q_{n-1} + q_n$ укупна величина QDD функције f. Очигледно је $Q(f) \geq B(f)$, према (76). Са друге стране, $Q(f)$ не може бити ни много веће од $B(f)$, јер (77) имплицира да је

$$Q(f) \leq \frac{n+1}{2}\,(B(f)+1) \tag{78}$$

■

5.2.3. Функција са скривеним адресирањем бита

Наредна дефиниција даје опис Булове функције која се зове *функција са скривеним адресирањем бита*, ([1], стр. 235, и [2], стр. 134). За ову функцију је доказано да има велики BDD, без обзира на редослед променљивих.

Дефиниција 26. (Функција са скривеним адресирањем бита): Булова функција $h_n : \mathcal{B}^n \to \mathcal{B}$ која враћа истинитосну вредност променљиве чији је индекс једнак укупном броју јединица међу променљивама назива се функција са скривеним адресирањем бита. Нека је $vx = x_1 + \cdots + x_n$. Тада је:

$$h_n(x_1, \ldots, x_n) = \begin{cases} x_{x_1+\cdots+x_n} = x_{vx}, & \text{ако је } vx > 0 \\ 0, & \text{ако је } vx = 0 \end{cases} \tag{79}$$

Теорема 15. Функција са скривеним адресирањем бита има експоненцијално велики BDD, за било коју пермутацију променљивих.

Доказ. Показује се да је $B(h_n)$ асимптотски једнак $c\lambda^n + O(n^2)$, где је $\lambda \approx 1.32472\ldots$ позитиван корен једначине $\lambda^3 = \lambda + 1$, а константа c је једнака $7\lambda - 1 + 14/(3 + 2\lambda) \approx 10.75115$. (Комплетан доказ се налази у вежби 125 у раду [1]).

■

6. Реализација

За реализацију алгоритама за конструкцију и манипулацију BDD из поглавља 3 коришћен је програмски језик *Python,* верзија 2.7.7 [6], и објектно оријентисана методологија. Имплементиране су главне класе које реализују BDD. Сваки алгоритам је имплементиран као засебна библиотека која се покреће независно од других библиотека, а која може да позива главне класе и (или) функције других библиотека.

Одељак о имплементацији сваког алгоритма садржи:

- име библиотеке,
- преглед шта се унутар библиотеке користи из других библиотека, односно њихових функција,
- процес реализације и извршавања алгоритма,
- пример извршавања алгоритма.

Свака функција садржи спецификацију улазних и излазних података, заједно са детаљним коментарима сваког дела имплементације.

6.1. Структура програма, главне класе и помоћне функције

Овај одељак садржи упутство за покретање имплементације алгоритама из поглавља 3, затим детаљан опис класа које представљају BDD, и кратак преглед помоћних функција које врше проверу улаза функције која их позива.

6.1.1. Поступак покретања библиотека . *py*

Свака библиотека се састоји из три дела, у редоследу у ком су делови наведени:

- З*аглавље библиотеке* - укључује друге библиотеке и/или главне класе, и/или *python* библиотеке, чиме њихове функције постају доступне библиотеци која их позива
- *Тело библиотеке* - садржи имплементацију једне или више функција потребних за реализацију алгоритма који библиотека имплементира. Ове функције такође могу позивати функције из других библиотека
- *Тестирање* - покреће извршавање библиотеке и позива функције из других библиотека

Покретање извршавања библиотека састоји се у мањим изменама дела [*Тестирање*]. Део [*Тестирање*] у свим библиотекама има следећи облик:

```
'''
# ***************  Testing  **********************

Позиви функција                                                    (80)

'''
```

Овде троструки наводници **'''** означавају блок коментар, односно све линије кода у оквиру њих постају коментар, а карактер "#" представља линијски коментар. За покретање извршавања

библиотеке, потребно је **уписати** карактер "#" испред троструких наводника у делу [*Тестирање*] (из 80), тако да тај део изгледа тачно како је дато у (81):

```
#'''
# ***************  Testing  **********************

Позиви функција                                                    (81)

#'''
```

Овим се постиже да линије програмског кода које се налазе између троструких наводника више *нису* коментари. Овај поступак важи за покретање свих библиотека. Након завршетка рада са одређеном библиотеком, неоходно је избрисати карактере "#", тако да део [*Тестирање*] има облик (80).

Пример 25. Слика 18 илуструје структуру и садржај библиотеке *random_solutions.py* (видети одељак 6.3.3) која имплементира алгоритам за генерисање случајног решења једначине $f(x_1, \dots, x_n) = 1$ (видети одељак 3.2.3). (Неке функције су изостављене ради једноставности илустрације).

[*Заглавље библиотеке*]

```
from bdd_classes import Node, BDD
from bdd_from_beads import robdd_from_beads, call_bdd_beads
from count_solutions_C import count_solutions
import math, random
```

[*Тело библиотеке*]

```
def rnd_sol(bdd):
def call_random_solutions(k, bdd):
def complete_random_solution(random_solution):
def printRandomSolutions(random_solutions):
```

[*Тестирање*]

```
'''

# ***************  Testing  **********************

bdd_test = call_bdd_beads('1100100100001111')
count_solutions(bdd_test)
one_solution = call_random_solutions(1, bdd_test)
random_solutions = call_random_solutions(1000, bdd_test)

'''
```

Слика 18: Структура библиотеке *random_solutions.py*.

Покретање библиотеке *random_solutions.py* се врши тако што се упише карактер "#" испред оба трострука наводника у делу [*Тестирање*], а потом се програм покрене на стандардан начин (командом *Run*, *Build*, и слично, зависно од окружења у коме се библиотеке тестирају).

Напомена. Библиотеке неће радити уколико се позивају из каснијих *Python* верзија (верзије након *Python 2.7.7*), услед одређених некомпатибилности.

6.1.2. Класа Node

Класа Node је класа која представља чвор BDD. Следи опис њених атрибута и функција за обраду.

Атрибути

Основни атрибути класе Node су:

- nodeId - идентификатор инстанце класе Node, јединствен за сваки чвор у BDD
- varIndex - индекс променљиве која означава чвор
- nodeBead - перла чвора
- leftBranch - леви син чвора
- rightBranch - десни син чвора.

Приликом креирања чвора, идентификатор чвора nodeId добија вредност од променљиве Node.nextIdNum класе Node, која је заједничка за све њене инстанце. Вредност идентификатора се придружује новој инстанци класе Node, након чега се његова вредност ажурира (увећава се за 1). Овај поступак је неопходан како би сваки новокреиран чвор имао јединствен идентификатор.

За креирање нове инстанце класе BDD, потребно је ресетовати идентификатор nextIdNum следећим кодом:

$$\text{Node.nextIdNum} = 1, \tag{82}$$

чиме се обезбеђује да нумерисање чворова при извршавању било које библиотеке увек крене од 1.

Додатни атрибути класе Node (који олакшавају имплементацију алгоритама у раду), су:

- node1count - број јединица у перли чвора
- parentsTrue - листа чворова којима је чвор Node десни син
- parentsFalse - листа чворова којима је чвор Node леви син
- level - ниво чвора (најчешће је једнак индексу променљиве, за природно уређење променљивих. Ниво завршних чворова је за један већи од броја променљивих у Буловој функцији)
- varName - име променљиве, које је једнако индексу променљиве у форми ниске, осим за завршне чворове, где је име променљиве за завршне чворове $\boxed{\top}$ односно $\boxed{\bot}$, редом 'T' односно 'F'.

Функције

Следе функције за креирање и манипулацију инстанцама класе Node.

Функција која представља конструктор класе Node је функција __init__(self, varIndex). Она креира инстанцу класе Node и поставља иницијалне вредности њеним атрибутима. Чвор се креира на основу прослеђеног индекса променљиве која га означава. Аргумент self је ознака класе чији се конструктор позива.

(Аргумент self је аргумент сваке функције која се налази у оквиру класе као једна од њених метода, које приступају њеним елементима и мењају их (видети self у референци [6]). У наставку, аргумент self се изоставља из потписа и описа функција класе).

Остале функције се деле у 3 групе; прве две групе су тзв. *гетер* и *сетер* методе (енгл. *Getter* и *Setter*), које служе редом за приступ вредностима атрибута, и мењање њихових вредности. Трећа група функција су функције за испис инстанци класе Node и њених атрибута.

1) Гетери - Функције за приступ сваком атрибуту инстанце класе Node:

- getNodeId() - функција враћа атрибут nodeId
- getVarIndex() - функција враћа атрибут varIndex
- getNodeBead() - функција враћа атрибут nodeBead
- getLeftBranch() - функција враћа атрибут leftBranch
- getRightBranch() - функција враћа атрибут rightBranch
- getNode1Count() - функција враћа атрибут node1count
- getParentsTrue() - функција враћа атрибут parentsTrue
- getParentsFalse() - функција враћа атрибут parentsFalse
- getLevel() - функција враћа атрибут level
- getVarName() - функција враћа атрибут varName

2) Сетери - Функције за промену вредности сваком атрибуту инстанце класе Node:

- setNodeBead(bead) - додељује вредност bead атрибуту nodeBead
- setLeftBranch(leftNode) - додељује leftNode атрибуту leftBranch
- setRightBranch(rightNode) - додељује rightNode атрибуту rightBranch
- setNode1Count(ck) - додељује вредност ck атрибуту node1count
- setParentsTrue(true_parent) - додељује true_parent атрибуту parentsTrue
- setParentsFalse(false_parent) - додељује false_parent атрибуту parentsFalse
- setLevel(level) - додељује вредност level атрибуту (класе) level
- setVarName(varName) - додељује вредност varName атрибуту varName

3) Функције за приказ атрибута и инстанце класе Node:

- __str__() - функција приказује инстанцу класе Node у форми:

$$(\text{nodeId}, \text{varIndex}, \text{nodeBead}) \tag{83}$$

- printNodeBranches() - функција приказује левог и десног сина чвора Node у форми:

$$(\text{leftBranch}, \text{rightBranch}) \tag{84}$$

- printNodeParents() - функција приказује листе идентификатора оца чвора Node, са вредношћу тачно и нетачно.

- printNodeData() - функција исписује све податке о чвору.

6.2.2. Класа BDD

Основни елемент (атрибут) класе BDD је структура varNodes који садржи различите инстанце класе Node (које представљају све чворове који чине BDD), а облика је:

$$\begin{aligned}&\{\ \text{varIndex\_1: [node\_1]},\\ &\ \ \text{varIndex\_2: [node\_2, node\_3, ... , node\_k]},\\ &\qquad\qquad .....\\ &\ \ \text{varIndex\_m: [node\_2, node\_i, ... , node\_k]},\\ &\qquad\qquad .....\\ &\ \ \text{varIndex\_n: [node\_i, node\_k, ... , node\_j]}\ \},\end{aligned} \tag{85}$$

где кључ речника varIndex_m представља индекс променљиве x_m, а њему одговарајући елемент је

низ чворова којима се та променљива придружује. Имплементација подразумева природно уређење променљивих Булове функције $f(x_1, \dots, x_n)$.

Уочава се у (85) да се један чвор може наћи у више низова чији су кључеви индекси променљивих - ово је због тога што један чвор може имати више очева, којима није нужно придружена иста променљива (очеви се налазе на различитим нивоима).

Атрибути

Класа BDD садржи следеће атрибуте:

- varNodes (видети (85))
- varList - листа индекса променљивих (односно кључева речника varNodes)
- nodeT - завршни чвор ⊤
- nodeF - завршни чвор ⊥

Завршни чворови су јединствени у BDD, немају синове, и не придружује им се променљива функције, већ њена истинитосна вредност. Овде представљају засебне атрибуте јер је програмски код једноставнији (за имплементацију и разумевање).

Листа индекса променљивих је засебан атрибут јер олакшава сортирање индекса променљивих (речник varNodes не подржава једноставно сортирање својих кључева).

Функције

Следе функције за креирање и манипулацију инстанцама класе BDD.

Функција која представља конструктор класе BDD је функција __init__(). Она креира инстанцу класе BDD и поставља иницијалне вредности њеним атрибутима. Иницијално, BDD се креира као празна инстанца, а атрибути добијају вредност током примене алгоритама за конструкцију BDD.

Остале функције се деле у 4 групе: гетер и сетер методе, функције специфичне намене, и функције за испис инстанци класе BDD и њених атрибута.

1) Гетери - Функције за приступ сваком атрибуту инстанце класе BDD

- getTrueNode() - враћа завршни чвор ⊤
- getFalseNode() - враћа завршни чвор ⊥
- getRoot() - враћа чвор који је корен BDD, односно чвор node_1 из речника (85), елемента varIndex_1: [node_1]
- getLevelNodes(level) - враћа листу чворова којима је придружена променљива са задатим индексом level
- getAllNodes() - враћа структуру Речник varNodes (85)

2) Сетери - Функције за промену вредности сваком атрибуту инстанце класе BDD

- setTrueNode(nodeT) - поставља прослеђен чвор nodeT као чвор ⊤
- setFalseNode(nodeF) - поставља прослеђен чвор nodeF као чвор ⊥
- addNode(node) - додаје прослеђен чвор речнику varNodes (односно, додаје чвор у BDD)

3) Функције специфичне намене - функције које обрађују класу BDD и њене елементе

- BFSgetNodeByID(node_id) - враћа чвор са задатим идентификатором претрагом BDD у ширину

- getLevel(root) - рекурзивно израчунава ниво BDD, почев од корена root
- sortVarNodes() - сортира листу индекса променљивих varList у опадајућем редоследу. Ово је потребно јер већина имплементираних алгоритама има приступ ниво по ниво, од завршних чворова према корену
- sortALLNodes() - сортира све чворове у инстанци класе BDD, тако да су чворови који се налазе на истом нивоу (односно којима је придружена иста променљива) сортирани према идентификатору, у растућем редоследу
- numberOfNodes() - враћа укупан број чворова у BDD, $B(f)$

4) Функције за приказ атрибута и инстанце класе BDD - следеће функције приказују инстанцу класе BDD у различитој форми:

- simplePrintBDD() - функција исписује идентификаторе чворова на сваком нивоу BDD:

|| m ||:
nodeId_i nodeId_j nodeId_k ... (86)

где је m ниво (односно индекс m променљиве x_m) у BDD ($1 \leq m \leq n$), a nodeId_i, nodeId_j, nodeId_k су идентификатори чворови на том нивоу (ознаке i, j, k су опште и наглашавају да број чворова на одређеном нивоу није унапред познат и зависи од Булове функције).

- prettyPrintBDDwithSons() - функција исписује идентификаторе чвора и његовог левог и десног сина, за сваки чвор у нивоу BDD, за сваки ниво у BDD:

|| m ||:
nodeId_i(nodeId_i_leftBranch, nodeId_i_rightBranch) nodeId_j(nodeId_j_leftBranch, nodeId_j_rightBranch) ... (87)

где важи објашњење као за (86). Овде су (nodeId_i_leftBranch, nodeId_i_rightBranch) идентификатори редом левог и десног сина чвора nodeId_i.

- prettyPrintBDDwithSonsBeads() - функција исписује идентификатор чвора, његову перлу, и идентификаторе његовог левог и десног сина, за сваки чвор у нивоу BDD, за сваки ниво у BDD:

|| m ||:
nodeId_i[nodeBead_i](nodeId_i_leftBranch, nodeId_i_rightBranch) nodeId_j[nodeBead_j] (nodeId_j_leftBranch, nodeId_j_rightBranch) ... (88)

где важи објашњење као за (87). Овде је nodeBead_i перла чвора nodeId_i.

Пример 26. Нека је дата истинитосна табела Булове функције 1110001011011100. Испис BDD који одговара датој функцији је дат у форми (88).

```
||1||:
|1|[1100100100001111](4, 5)

||2||:
|4|[11001001](6, 7)    |5|[00001111](3, 2)

||3||:
|6|[1100](2, 3)    |7|[1001](8, 9)

||4||:
|8|[10](2, 3)    |9|[01](3, 2)

||5||:
```

```
|2|[1]    |3|[0]
```

Испис се тумачи на следећи начин:

На нивоу ||1||, налази се корен чији је идентификатор једнак |1|, и коме је придружена променљива x_1. Перла корена је [1100100100001111] (задата истинитосна табела). Леви и десни син корена су редом чворови са идентификаторима (4, 5).

На нивоу ||2||, налазе се чворови којима је додељена променљива x_2:

- чвор чији је идентификатор једнак |4|, чија је перла једнака [11001001], и чији су леви и десни син чворови са идентификаторима редом (6, 7)
- чвор чији је идентификатор једнак |5|, чија је перла једнака [00001111], и чији су леви и десни син чворови са идентификаторима редом (3, 2)

Слично тумачење важи и за остале нивое, осим за последњи ниво BDD.

На последњем нивоу ||5||, налазе се завршни чворови са истинитосним вредностима, и то:

- чвор чији је идентификатор једнак |2| представља завршни чвор $\boxed{\top}$, чија је перла једнака [1]
- чвор чији је идентификатор једнак |3| представља завршни чвор $\boxed{\bot}$, чија је перла једнака [0].

Напомене

- Сортирање идентификатора чворова у растућем поретку на сваком нивоу је неопходно једино у алгоритму *Редукција одоздо на горе* (видети одељак 3.1.2), али не представља модификацију BDD.

- Корен, и завршни чворови $\boxed{\top}$ и $\boxed{\bot}$, имају увек идентификаторе редом 1, 2 и 3, осим у имплементацији алгоритма Редукција одоздо на горе (видети одељак 3.1.2) (јер се прво гради комплетно бинарно стабло чијим се чворовима редом придружују идентификатори, а потом се редундантни чворови уклањају.)

BDD пакети

Постоје разни доступни BDD пакети који су развијани на светским Универзитетима. Неки од таквих пакета су ([2], стр. 119):

1) BDD пакет чији су аутори *K.Brace, R.Rudell, R.Bryant*
2) Побољшана верзија 1), чији је аутор *D.Long*
3) BDD пакет CUDD, чији је аутор *F.Somenzi* [7].

Ови пакети садрже многобројне функционалности, међу којима су и имплементације модификација BDD (видети одељак 2.5). Имплементација BDD у овом раду не досеже њихову изражајност и комплексност, већ садржи минимални скуп одлика, са нагласком на практичне примене BDD у решавању разних проблема (на пример, решавање проблема линеарног Буловог програмирања, графовски алгоритми, и остале имплементације у раду).

6.1.4. Генератор случајне истинитосне табеле

Библиотека која служи за генерисање случајних истинитосних табела за задат број променљивих

зове се *random_truth_table.py.* Библиотека позива *python* уграђене библиотеке *math* и *random*.

Она садржи функцију truth_table_random_gen(var_num), која генерише случајан број и у зависности од његове вредности бира једну од истинитосних вредности 0 или 1 за случајну истинитосну табелу.

6.1.5. Функције за проверу улаза

Библиотека *checker_functions.py* садржи имплементације функција које алгоритми у поглављу 3 користе при обради BDD, а служе за одређене провере или помоћна израчунавања. Она садржи четири функције провере:

1) функцију за проверу уноса истинитосне табеле, check_truth_table
2) функцију за израчунавање броја променљивих, variables_number
3) функцију за идентификовање константне бинарне ниске, is_constant
4) функцију за проверу да ли је бинарна ниска квадратна, is_square

Провера уноса истинитосне табеле

Потпис функције: check_truth_table(truth_table)

Функција врши проверу формата уноса бинарне ниске која представља истинитосну табелу Булове функције. Услови које функција проверава су:

- дужина бинарне ниске truth_table - мора бити једнака степену броја 2,
- формат бинарне ниске truth_table - мора бити ниска карактера
- карактери бинарне ниске truth_table - морају бити или 0 или 1

Додатна провера коју функција врши је провера да ли је Булова функција која одговара бинарној ниски константно тачна, или константно нетачна, и уколико јесте, исписује резултат провере.

Уколико бинарна ниска није константно тачна или константно нетачна, функција враћа 1, што се тумачи као сигнал да се за задату истинитосну табелу може конструисати BDD.

Израчунавање броја променљивих

Потпис функције: variables_number(truth_table)

Ова функција враћа цео број n (односно $log_2(2^n)$), који представља број променљивих прослеђене истинитосне табеле truth_table дужине 2^n.

Константна бинарна ниска

Потпис функције: is_constant(bead_substring)

Функција проверава да ли је прослеђена бинарна ниска bead_substring константна (тј. облика 111…..1, или 000….0), и враћа наредне вредности:

- 1, ако је бинарна ниска константно тачна
- 0, ако је бинарна ниска константно нетачна
- (-1), ако бинарна ниска није константна.

Квадратна ниска

Потпис функције: is_square(bead_substring)

Функција проверава да ли је прослеђена бинарна ниска bead_substring квадрат, односно да ли се може записати у облику aa, где је a њена бинарна подниска. Уколико јесте квадрат, функција враћа True, а уколико није враћа False.

Следе библиотеке које реализују алгоритме у поглављу 3. Свака библиотека позива ресетовање идентификатора nextIdNum на почетну вредност бројача, позивом (82).

6.2. Конструкција BDD

Овај одељак садржи опис имплементације алгоритама из одељака 3.1.1 и 3.1.2.

6.2.1. Конструкција BDD помоћу перли

Библиотека која имплементира алгоритам Конструкција помоћу перли (видети одељак 3.1.1) назива се *bdd_from_beads.py*, и садржи две функције:

1) функцију за проверу уноса за конструкцију помоћу перли, call_bdd_beads
2) функцију за конструкцију BDD помоћу перли**,** robdd_from_beads

Провера уноса за конструкцију помоћу перли

Потпис функције: call_bdd_beads(truth_table)

Функција позива функције за проверу уноса (видети одељак 6.1.5) за:

- проверу уноса истинитосне табеле, check_truth_table, и
- израчунавање броја променљивих, variables_number

Уколико је формат уноса истинитосне табеле Булове функције одговарајућ, позива се главна функција за конструкцију BDD robdd_from_beads, која је описана у наставку.

Конструкција BDD помоћу перли

Потпис функције: robdd_from_beads(truth_table, max_level)

Функција креира корен и завршне чворове са истинитосним вредностима $\boxed{\top}$ и $\boxed{\bot}$ (јер је истинитосна табела испунила услове за креирање BDD). Корен се као чвор смешта у привремени низ tmp_level_nodes, одакле се додаје у у инстанцу класе BDD, bdd. У свакој итерацији петље, сваки први чвор у привременом низу се брише из низа и додаје у bdd, а његова перла tmp_bead се дели на леву и десну подниску, израчунава се ниво var_index за синове, и врше се провере на основу којих се постављају одговарајући леви и десни син. Уколико је током провере идентификована нова перла, за њу се креира одговарајући чвор, Node(var_index). Након доделе одговарајућем сину, тај

чвор се смешта на крај привременог низа, одакле улази у BDD у некој од наредних итерација (након што се заврше све провере за бинарне подниске његове перле).

Пример 27. Нека је дата истинитосна табела Булове функције 1100100100001111 (видети пример 8, слика 7). Поступак за покретање библиотеке је наведен у наставку.

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 199:

   bdd_test = call_bdd_beads('1100100100001111')

2. Покренути библиотеку *bdd_from_beads.py*.

Резултујући BDD се приказује у форми (88):

```
||1||:
|1|[1100100100001111](4, 5)

||2||:
|4|[11001001](6, 7)    |5|[00001111](3, 2)

||3||:
|6|[1100](2, 3)    |7|[1001](8, 9)

||4||:
|8|[10](2, 3)    |9|[01](3, 2)

||5||:
|2|[1]    |3|[0]
```

6.2.2. Комплетно уређено бинарно стабло

Библиотека која конструише комплетно уређено бинарно стабло задате истинитосне табеле Булове функције назива се *full_binary_tree.py.* Библиотека садржи три наредне функције:

1) функцију за проверу уноса истинитосне табеле за конструкцију стабла, call_full_binary_tree
2) функцију за конструкцију комплетног уређеног бинарног стабла, full_binary_tree_from_tt
3) функцију за приказ изграђеног стабла у одређеној форми, print_full_binary_tree

Провера уноса истинитосне табеле за конструкцију стабла

Потпис функције: call_full_binary_tree(truth_table)

Функција позива функције за проверу уноса (видети одељак 6.1.5) за:

- проверу уноса истинитосне табеле, check_truth_table, и
- израчунавање броја променљивих, variables_number

Ако је унос бинарне ниске у формату истинитосне табеле Булове функције, позива се главна функција за конструкцију, full_binary_tree_from_tt. Функција враћа комплетно уређено бинарно стабло, full_binary_tree, као резултат позива функције full_binary_tree_from_tt.

Конструкција комплетног уређеног бинарног стабла

Потпис функције: full_binary_tree_from_tt(truth_table, max_level)

Функција креира комплетно уређено бинарно стабло према идеји конструкције BDD помоћу перли (видети одељак 6.2.1). Функција користи класу Node за представљање чвора комплетног бинарног стабла, и класу BDD за представљање комплетног бинарног стабла, уз следеће модификације:

- За сваку подтабелу иницијалне табеле truth_table креира се чвор који улази у комплетно уређено бинарно стабло.
- У променљивој nodeBead чува се бинарна ниска која је подтабела табеле чвора који је отац чвора који се креира.
- Резултујуће комплетно уређено бинарно стабло има онолико завршних чворова колика је дужина истинитосне табеле.

Функција враћа комплетно уређено бинарно стабло full_binary_tree функцији која је позива (call_full_binary_tree).

Приказ комплетног бинарног стабла

Потпис функције: print_full_binary_tree(full_binary_tree)

Функција исписује комплетно уређено бинарно стабло у форми (88), уз модификације описане у делу *Конструкција комплетног уређеног бинарног стабла* изнад.

Пример 28. Нека је дата истинитосна табела Булове функције 10110100. Поступак за покретање библиотеке је наведен у наставку.

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_full_binary_tree у делу [*Тестирање*], линија 160:

   full_binary_tree = call_full_binary_tree('10110100')

2. Покренути библиотеку *full_binary_tree.py.*

Резултујуће комплетно уређено бинарно стабло одлучивања се приказује у форми (88):

```
||1||:
|1|[10110100](2, 3)

||2||:
|2|[1011](4, 5)    |3|[0100](6, 7)

||3||:
|4|[10](8, 9)    |5|[11](10, 11)    |6|[01](12, 13)    |7|[00](14, 15)

||4||:
|8|[1]    |9|[0]    |10|[1]    |11|[1]    |12|[0]    |13|[1]    |14|[0]    |15|[0]
```

6.2.3. Помоћне функције за редукцију бинарног стабла

Библиотека која садржи имплементацију помоћних функција које врше израчунавања у оквиру алгоритма за конструисање BDD редукцијом одоздо на горе (видети одељак 3.1.2), зове се *reduction_helper_functions.py*. Библиотека садржи три наредне функције:

1) функцију за одређивање два јединствена завршна чвора (⊤ и ⊥), determine_sink_nodes
2) функцију за елиминацију чвора, node_elimination
3) функцију за спајање чворова, node_merging

Функција под 1) се позива само једном (пре примене редукције на бинарно стабло), а функције под 2) и 3) се позивају из функције која имплементира овај алгоритам (видети наредни одељак 6.2.4).

Одређивање завршних чворова бинарног стабла

Потпис функције: determine_sink_nodes(full_binary_tree)

Ова функција врши припрему последњег нивоа са завршним чворовима прослеђеног бинарног стабла. Како у BDD постоје само два завршна чвора ⊤ и ⊥, функција бира ове чворове тако да изабран чвор ⊤ има најмањи идентификатор међу свим ⊤ чворовима; слично важи за ⊥ чвор. Сви остали чворови који су на последњем нивоу се бришу из прослеђеног комплетног бинарног стабла.

Елиминација чвора

Потпис функције: node_elimination(binary_tree, node_to_eliminate)

Ова функција имплементира прво правило алгоритма Редукција одоздо на горе, вршећи елиминацију прослеђеног чвора node_to_eliminate из бинарног стабла binary_tree. Чвор за елиминацију се уклања из низа чворова оца сваког свог сина (parentsTrue, parentsFalse), а потом се врши преспајање показивача у сваком чвору који је отац чвора који се брише, на његовог сина.

Спајање чворова

Потпис функције: node_merging(binary_tree, main_node, node_to_merge)

Ова функција имплементира друго правило алгоритма Редукција одоздо на горе, вршећи спајање прослеђених чворова у бинарном стаблу. Спајање чворова се врши тако што се чува чвор са мањим идентификатором main_node, а очеви другог чвора node_to_merge се преспајају на сачувани чвор main_node.

Ова функција се позива онолико пута колико има чворова који испуњавају услове спајања.

6.2.4. Конструкција BDD применом Редукције одоздо на горе

Библиотека која имплементира алгоритам конструкције BDD Редукција одоздо на горе (видети одељак 3.1.2), зове се *bdd_reduction_Bottom_Up.py*. Библиотека садржи једну функцију за конструкцију BDD, а позива функције из других библиотека, које се извршавају у наведеном редоследу:

1) функција за изградњу комплетног уређеног бинарног стабла одлучивања, call_full_binary_tree (видети одељак 6.2.2),
2) функција која одређује два јединствена завршна чвора ⊤ и ⊥ за BDD, determine_sink_nodes (видети одељак 6.2.3),
3) главна функција за конструкцију BDD применом редукције на комплетно бинарно стабло, reduction_bottom_up
4) функција за приказ BDD, prettyPrintBDDwithSons (видети одељак 6.1.3)

Конструкција BDD редукцијом комплетног бинарног стабла

Потпис функције: reduction_bottom_up(full_binary_tree)

Функција пролази прослеђено бинарно стабло full_binary_tree ниво по ниво, од завршног нивоа према корену. Током сваког пролаза, проверава се сваки чвор да ли је кандидат за елиминацију, за спајање на неки чвор у том нивоу, или за останак у BDD. У зависности од резултата провере, извршавају се следећи кораци:

1) позива се функција за имплементацију правила елиминације, node_elimination (видети одељак 6.2.3)
2) позива се функција за имплементацију правила спајања, node_merging (видети одељак 6.2.3) - за сваки чвор у нивоу се проверава да ли остаје у нивоу, или је кандидат за спајање са чвором који је пре њега проверен и задржан.

Функција исписује резултујући BDD у форми (88).

Пример 29. Нека је дата истинитосна табела Булове функције 10110100 из примера 28. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_full_binary_tree у делу [*Тестирање*], линија 82:

   full_binary_tree = call_full_binary_tree('10110100')

2. Покренути библиотеку *bdd_reduction_Bottom_Up.py.*

Резултујуће комплетно уређено бинарно стабло одлучивања се приказује у форми (88):

```
||1||:
|1|(2, 3)

||2||:
|2|(4, 8)    |3|(6, 9)

||3||:
|4|(8, 9)    |6|(9, 8)

||4||:
|8|    |9|
```

Комплетно бинарно стабло из примера 28 има укупно 15 чворова, а након примене редукције добијени BDD има 7 чворова.

6.3. Обрада Булових функција помоћу BDD

Овај одељак описује обраду Булових функција употребом BDD. Обрада се односи на израчунавање решења Булове једначине $f(x_1, \dots, x_n) = 1$, њене модификације и примене.

6.3.1. Број решења Булове једначине

Библиотека која израчунава број решења у сваком чвору BDD се зове *count_solutions_C.py*. Она имплементира алгоритам Број решења Булове једначине $f(x_1, \dots, x_n) = 1$ (одељак 3.2.1) помоћу функције count_solutions, а позива функције из других библиотека, које се извршавају у наведеном редоследу:

1) функција за изградњу комплетног уређеног бинарног стабла одлучивања, call_full_binary_tree (видети одељак 6.2.2),
2) функција за израчунавање броја решења, count_solutions
3) функција за приказ BDD са израчунатим бројем решења у сваком чвору, prettyPrintBDDwithOnes

Израчунавање броја решења

Потпис функције: count_solutions(bdd)

Функција пролази BDD, bdd, ниво по ниво, и на сваком нивоу (где постоје чворови) пролази чвор по чвор, рачунајући број јединица у сваком чвору, на основу броја јединица његовог левог и десног сина.

Испис броја решења у сваком чвору BDD

Потпис функције: prettyPrintBDDwithOnes(bdd)

Функција исписује BDD, bdd, са израчунатим бројем решења у форми:

|| m ||:
nodeId_i[nodeBead_i]{node1count}(nodeId_i_leftBranch, nodeId_i_rightBranch) ... (89)

где важи објашњење као у (88). Овде је nodeBead_i перла чвора nodeId_i, а node1count број решења у чвору, односно број јединица у његовој перли ($1 \leq i \leq B(f)$).

Пример 30. Нека је дата истинитосна табела Булове функције 1110001011011100. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 68:

   bdd_test = call_bdd_beads('1110001011011100')

2. Покренути библиотеку *count_solutions_C.py.*

Резултујући BDD је дат у наставку.

```
||1||:
|1|[1110001011011100]{9}(4, 5)

||2||:
|4|[11100010]{4}(6, 7)    |5|[11011100]{5}(8, 9)

||3||:
|6|[1110]{3}(2, 10)    |7|[0010]{1}(3, 10)    |8|[1101]{3}(2, 11)
|9|[1100]{2}(2, 3)

||4||:
|10|[10]{1}(2, 3)    |11|[01]{1}(3, 2)

||5||:
|2|[1]{1}    |3|[0]{0}
```

6.3.2. Сва решења Булове једначине

Библиотека која проналази сва решења једначине $f(x_1, \dots, x_n) = 1$ зове се *count_solutions_All.py.* Она служи као додатак претходном одељку (6.3.1), и садржи две функције:

1) функцију за проналажење свих решења, count_all_solutions(bdd)
2) функцију која исписује сва решења у сваком чвору у облику низа ниски, prettyPrintBDDwithSolutions(all_solutions)

Проналажење свих решења

Потпис функције: count_all_solutions(bdd)

Функција проналази сва решења једначине $f(x_1, \dots, x_n) = 1$ на сличан начин као што се рачуна укупан број решења у сваком чвору. Разлика је што се у овом алгоритму креира решење помоћу решења у левом и десном сину,тако што се на свако решење из сваког сина надовезује 1, 0 или x (видети одељак 3.2.2). Функција враћа речник

$$\{\text{nodeId\_1: ["solution1, solution2,...."], nodeId\_2: .....}\} \tag{90}$$

Овде је сваком чвору придружена листа свих решења од тог чвора до завршног чвора $\boxed{\top}$, а свако решење је ниска која се састоји из 1, 0, и / или x.

Позив функције за испис свих решења

Потпис функције: prettyPrintBDDwithSolutions(all_solutions)

Функција приказује сва решења у сваком чвору BDD, у облику

Функција исписује BDD, bdd, са израчунатим бројем решења у форми:

|| m ||:
nodeId_i['solution_1', 'solution_2', ..., 'solution_k'](nodeId_i_leftBranch, nodeId_i_rightBranch) ... (91)

где важи објашњење као у (88). Овде је 'solution_k' једно решење од чвора nodeId_i до чвора $\boxed{\top}$.

Пример 31. Нека је дат BDD из примера 30. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 108:

   bdd_test = call_bdd_beads('1110001011011100')

2. Покренути библиотеку *count_solutions_All.py*.

Резултат извршавања се приказује у форми (91):

```
||1||:
|1|['000x', '0010', '0110', '100x', '1011', '110x'](4, 5)

||2||:
|4|['00x', '010', '110'](6, 7)   |5|['00x', '011', '10x'](8, 9)

||3||:
|6|['0x', '10'](2, 10)   |7|['10'](3, 10)   |8|['0x', '11'](2, 11)
|9|['0x'](2, 3)

||4||:
|10|['0'](2, 3)   |11|['1'](3, 2)

||5||:
|2|['']   |3|[]
```

6.3.3. Случајно решење Булове једначине са униформном расподелом вероватноће

Библиотека која имплементира алгоритам Случајни избор решења једначине $f(x_1, \dots, x_n) = 1$ (одељак 3.2.3) зове се *random_solutions.py*. Она садржи три функције:

1) функцију за проналажење случајног решења са униформном расподелом вероватноће, rnd_sol. Ова функција позива уграђен *python* модул *random*.
2) функцију за креирање комплетног случајног решења, complete_random_solution
3) функцију која позива функцију под 1) задати број пута, и броји које решење се колико пута појавило, call_random_solutions
4) функцију за приказ случајног решења са униформном расподелом вероватноће, printRandomSolutions

Пре позива својих функција 1) - 4), библиотека позива функције из других библиотека у наведеном редоследу:

- функција за креирање BDD помоћу перли, call_bdd_beads (видети одељак 6.2.1),
- функција за израчунавање броја решења, count_solutions (видети одељак 6.3.1),

а након тога позива функције 1) - 4).

Генерисање случајног решења са униформном расподелом вероватноће

Потпис функције: rnd_sol(bdd)

Функција генерише случајно решење са униформном расподелом вероватноће, у смеру од корена ка завршном чвору $\boxed{1}$, крећући се чворовима који су изабрани са једнаком вероватноћом (видети одељак 3.2.3). Случајно решење се састоји из 0, 1 или карактера "x", који се уписује у случајно решење када је ниво у BDD "прескочен", односно када променљива чији индекс одговара прескоченом нивоу може подједнако вероватно бити једнака и 0 и 1 у случајном решењу.

Комплетно случајно решење

Потпис функције: complete_random_solution(random_solution)

Ова функција узима као улаз случајно решење које враћа функција rnd_sol, и уколико оно садржи карактер "x", на случајан начин бира 1 или 0 са једнаком вероватноћом 0.5, и уписује избор уместо карактера "x". На тај начин ова функција "употпуњује" случајно решење, јер сваком решењу из random_solutions које садржи "x" одговара решење ком је додељена случајно изабрана истинитосна вредност уместо x.

Број случајног појављивања сваког решења

Потпис функције: call_random_solutions(k, bdd)

Функција позива функцију rnd_sol k пута и броји колико се пута свако решење појавило као случајно. Решења се креирају на два начина, употребом променљивих:

1) random_solutions, која поред 0 и 1 може да садржи "x" на месту оне променљиве чија истинитосна вредност у решењу може бити и 0 и 1
2) random_solution_complete, садржи решење које се састоји само из 0 и 1. Обе променљиве random_solutions и random_solution_complete су речници облика

$$\text{random\_solution: num} \tag{92}$$

где је random_solution случајно решење из random_solutions, односно random_solution_complete, а num број појављивања тог решења током задатог броја итерација k функције call_random_solutions.

Ако је k=1, тада се функција за генерисање случајног решења позива само једном.

Испис униформно случајног решења

Потпис функције: printRandomSolutions(random_solutions, complete_random_solutions)

Функција приказује једно случајно решење из речника random_solutions, и њему одговарајуће комплетно решење из речника complete_random_solutions. Уколико решење из random_solutions не садржи "x", ова два исписа су једнака.

Након исписа једног решења, исписује се резултат функције call_random_solutions, у форми (92).

Пример 32. Нека је дат BDD са истинитосном табелом 1100100100001111. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 183:

```
bdd_test = call_bdd_beads('1100100100001111')
```

2. Унети број итерација (на пример 1000) функције call_random_solutions, линија 199:

$$\text{random\_solutions} = \text{call\_random\_solutions}(1000, \text{bdd\_test})$$

3. Покренути библиотеку *random_solutions.py*.

Резултат покретања ове библиотеке је:

1) Случајно решење са униформном расподелом вероватноће

```
Slucajno resenje sa uniformnom raspodelom verovatnoce:  11xx
```
(93)

У овом решењу, променљиве x_1 и x_2 имају истинитосну вредност 1, док обе променљиве x_3 и x_4 могу имати истинитосне вредности 1 и 0, са једнаком вероватноћом - због чега им је дописан "x".

```
Kompletno slucajno resenje koje odgovara ispisanom resenju:  1111
```
(94)

Ово решење је решење из (93) у коме су на случајан начин свакој променљивој x_3 и x_4 додељене истинитосне вредности, редом 1 и 1, са једнаком вероватноћом.

2) Број појављивања сваког случајног решења током 1000 итерација:

```
Sva resenja sa jednakom verovatnocom i brojem pojavljivanja:

                        0111 : 105
                        000x : 293
                        0100 : 110
                        11xx : 492
```
(95)

Постоји укупно 8 решења (што се види и из истинитосне табеле функције). Тачан број појављивања сваког решења је 1000/8 = 125. Како се униформно решење `11xx` "грана" на 4 различита решења: $1100, 1101, 1110, 1111$, то је број појављивања сваког решења 517 / 4 ~ 129, што је приближно очекиваном броју 125. Број је појављивања сваког решења које не садржи "x" близу броју 129.

Када се у сваком решењу које садржи "x" , на случајан начин изабере истинитосна вредност 0 или 1 на место "x", добија се следећи резултат.

```
Sva kompletna resenja sa jednakom verovatnocom i brojem pojavljivanja:

                        0111 : 105
                        0000 : 143
                        0001 : 150
                        0100 : 110
                        1111 : 126
                        1110 : 113
                        1100 : 137
                        1101 : 116
```
(96)

6.3.4. Израчунавање функције генератрисе решења Булове једначине

Библиотека која имплементира алгоритам за креирање генератрисе, Генератриса скупа решења са различитим бројем решења (видети одељак 3.2.4), зове се *generatrice.py*.

Библиотека позива функцију за креирање BDD (видети одељак 6.2.1), а потом позива своје две функције:

1) функцију за израчунавање генератрисе, gen_func
2) функцију за приказ BDD са генератрисом у сваком чвору, printBDDWithGeneratrise

Израчунавање генератрисе

Потпис функције: gen_func(bdd)

Функција рачуна генератрису сваког чвора прослеђеног bdd у облику листе њених коефицијената, од завршних чворова према корену. Генератриса чвора се рачуна помоћу израчунатих генератриса у левом и десном сину, што укључује сабирање и множење коефицијената обе генератрисе (видети одељак 3.2.4).

Приказ BDD са генератрисом у сваком чвору

Потпис функције: printBDDWithGeneratrise(bdd, gener_func)

Функција исписује BDD и генератрису у сваком чвору, у форми:

|| m ||:
nodeId_i[gener_func](nodeId_i_leftBranch, nodeId_i_rightBranch) ... (97)

где важи објашњење као за (86). Овде је gener_func низ a_j коефицијената генератрисе.

(Напомена. Функција користи функционалности библиотеке *Polynomial.py* ([8]), које су потребне за представљање, сабирање и множење полинома. Користи се већ постојећа библиотека, јер имплементација операција над полиномима није део овог рада).

Пример 33. Нека је дат BDD са истинитосном табелом 1110001011011100. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 142:

   bdd_test = call_bdd_beads('1110001011011100')

2. Покренути библиотеку *generatrice.py*.

BDD са резултујућим генератрисама у сваком чвору дат је у форми (97):

```
||1||:
|1|[1, 3, 3, 2, 0](4, 5)

||2||:
|4|[1, 2, 1, 0](6, 7)    |5|[1, 2, 2, 0](8, 9)
```

```
||3||:
|6|[1, 2, 0](2, 10)   |7|[0, 1, 0](3, 10)   |8|[1, 1, 1](2, 11)   |9|[1, 1,
0](2, 3)

||4||:
|10|[1, 0](2, 3)   |11|[0, 1](3, 2)

||5||:
|2|[1]   |3|[0]
```

Генератриса корена BDD је уједно и генератриса свих решења Булове функције. Овде је генератриса $[a_0, a_1, a_2, a_3, a_4]$ једнака `[1, 3, 3, 2, 0]`. Дакле, постоји тачно:

- једно решење са 0 јединица,
- три решења са 1 јединицом,
- три решења са 2 јединице,
- два решења са 3 јединице,
- нула решења са 4 јединице.

Слично се тумаче и коефицијенти a_j генератрисе унутрашњег чвора; генератриса чвора са идентификатором `|6|` је `[1, 2, 0]`, па важи да од чвора `|6|` до завршног чвора ⊤ постоји тачно:

- једно решење са 0 јединица,
- два решења са 1 јединицом,
- нула решења са 2 јединице.

6.3.5. Израчунавање полинома поузданости Булове једначине

Библиотека која имплементира алгоритам Полином поузданости Булове једначине (видети одељак 3.2.5) зове се *reliability_polinomial.py*.

Библиотека позива функцију за креирање BDD (видети одељак 6.2.1) и функцију за израчунавање броја решења у сваком чвору BDD (видети одељак 6.3.1), а потом позива своје две функције:

1) функцију за израчунавање полинома поузданости, за сваки чвор у BDD, reliability_poly
2) функцију за приказ BDD са вредношћу полинома поузданости у сваком чвору, printBDDWithPolyProb

Израчунавање полинома поузданости

Потпис функције: reliability_poly(bdd, prob)

Аргумент функције prob представља листу вероватноћа да је променљива $x_i = 1$ када је $f(x_1, \dots, x_n) = 1$, за сваку променљиву Булове функције.

Функција рачуна вредност полинома поузданости за сваки чвор, од завршних чворова према корену. Ова вредност се за сваки чвор рачуна на основу вероватноћа у листи prob и вредности полинома поузданости у левом и десном сину чвора (видети одељак 3.2.5).

Приказ BDD са полиномом поузданости у сваком чвору

Потпис функције: printBDDWithPolyProb(bdd, polynomial)

Функција исписује BDD и полином поузданости у сваком чвору, у форми:

|| m ||:
nodeId_i[polynomial](nodeId_i_leftBranch, nodeId_i_rightBranch) ... (98)

где важи објашњење као за (88). Овде је polynomial низ вредности полинома поузданости за сваки чвор у BDD.

Пример 34. Нека је дата истинитосна табела функције већинског одлучивања 00010111 (видети пример 1, одељак 1), и нека су вероватноће да је променљива једнака 1 задате листом за сваку променљиву (x_1, x_2, x_3), редом: $[0.5, 0.5, 0.5]$. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 86:

   bdd_test = call_bdd_beads('00010111')

2. Унети вероватноће за сваку променљиву $x_i = 1$ када је $f(x_1, \dots, x_n) = 1$, линија 92:

   polynomial = reliability_poly(bdd_test, [0.5, 0.5, 0.5])

3. Покренути библиотеку *reliability_polinomial.py*.

BDD са резултујућим вредностима полинома поузданости у сваком чвору дат је у форми (98):

```
||1||:
|1|[0.5](4, 5)

||2||:
|4|[0.25](3, 6)    |5|[0.75](6, 2)

||3||:
|6|[0.5](3, 2)

||4||:
|2|[1]    |3|[0]
```

Вредност полинома поузданости у корену BDD ове функције је вероватноћа да је Булове једначина једнака 1, ако је вероватноћа истинитосне вредности 1 сваке њене променљиве једнака 0.5. Ово се може преформулисати у питање: Ако променљива функције може имати истинитосну вредност 1 или 0 са једнаком вероватноћом (0.5), и ако то важи за сваку променљиву функције, која је вероватноћа да је истинитосна вредност функције једнака 1?

Одговор на питање је 0.5 - дакле, вероватноћа да је функција већинског одлучивања тачна је једнака вероватноћи да је свака њена променљива једнака 1.

6.3.6. Израчунавање решења максималне тежине Булове једначине

Библиотека која имплементира решавање проблема линеарног Буловог програмирања (видети одељак 3.2.6) назива се *linear_boolean_problem.py*.

Она позива функцију за конструкцију BDD, а потом позива своје три функције, у редоследу у ком су дате:

1) функцију за израчунавање тежине сваког чвора, у зависности од задатих тежина променљивих, compute_weights
2) фунцкију за генерисање крајњег решења максималне тежине, на основу израчунатих тежина у функцији под 1), max_weight_solution
3) функцију за приказ BDD са тежинама у сваком чвору и информацији који син је допринео тој тежини чвора.

Израчунавање тежина свих чворова

Потпис функције: compute_weights(bdd, w_weights)

Аргумент функције w_weights представља листу задатих тежина сваке променљиве. Функција рачуна тежину сваког чвора приступом од завршних чворова ка корену, тако што се упоређују тежине левог и десног сина. Бира се већа тежина и уписује у помоћни речник m_t, а у помоћни речник t_t се уписују индикатори који син је допринео коначној тежини чвора. Тежина корена је максимална тежина решења.

Генерисање решења максималне тежине

Потпис функције: max_weight_solution(bdd, w_weights, t_t)

Функција пролази кроз чворове који су означени у низу t_t, од корена до завршног чвора $\boxed{\top}$, формирајући коначно решење.

Приказ BDD са тежинама у сваком чвору

Потпис функције: printBDDWithWeights(bdd, m_t, t_t)

Функција исписује BDD и полином поузданости у сваком чвору, у форми:

|| m ||:
nodeId_i[node_weight, node_branch](nodeId_i_leftBranch, nodeId_i_rightBranch) ... (99)

где важи објашњење као за (88). Овде је node_weight израчуната тежина чвора, а node_branch има вредност 0 уколико је леви син допринео тежини тог чвора, или 1 ако је десни син допринео тежини тог чвора.

Пример 35. Нека је дата истинитосна табела функције 1100100100001111, и тежине њених променљивих (x_1, x_2, x_3, x_4) редом $[1, -2, -3, 4]$. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске као аргумент функције call_bdd_beads у делу [*Тестирање*], линија 181:

bdd_test = call_bdd_beads('1100100100001111')

2. Унети тежине за сваку променљиву x_i, линија 189:

solution = compute_weights(bdd_test, [1, -2, -3, 4])

3. Покренути библиотеку *solutions_max_weight.py.*

BDD са резултујућим тежинама у сваком чвору и ознаком који је син допринео тежини, дат је у форми (99):

```
||1||:
|1|[4, 0](4, 5)

||2||:
|4|[4, 0](6, 7)    |5|[2, 1](3, 2)

||3||:
|6|[4, 0](2, 3)    |7|[1, 1](8, 9)

||4||:
|8|[0, 0](2, 3)    |9|[4, 1](3, 2)

||5||:
|2|[1]    |3|[0]

Resenje maksimalne tezine:  0001
```

Решење максималне тежине се тумачи на следећи начин: корен BDD (који је увек први чвор решења), има тежину 4 којој је допринео његов леви син, чвор са идентификатором `|4|`. Чвор `|4|` улази у решење максималне тежине. Његова тежина износи 4, а њој је допринео његов леви син, чвор са идентификатором `|6|`. Чвор `|6|` такође има тежину 4 добијену из левог сина, а његов леви син је завршни чвор $\boxed{\top}$. Дакле,решење максималне тежине је 0001, а његова тежина износи 4.

6.4. Независни скупова циклуса са n чворова

Библиотека која имплементира алгоритам одређивања независних скупова у циклусу C_n помоћу BDD (видети одељак 3.3) зове се *independent_subsets_Cn.py*.

Библиотека садржи следеће функције:

1) Функција за обраду базних циклуса C_n, $n = 0, 1, 2$, call_ind_set. Ова функција позива главну функцију за креирање BDD независних скупова у циклусу, independent_set
2) Функција за креирање BDD независних скупова у циклусу, independent_set
3) Функција за приказ BDD независних скупова у циклусу, prettyPrintBDD_IndSet
4) Функција за верификацију креирања BDD независних скупова, verify_cn_bdd

Библиотека позива функцију за одређивање броја решења Булове једначине $f(x_1, \dots, x_n) = 1$ (видети одељак 6.3.1), која у овом случају одређује колико има независних скупова у сваком чвору BDD независних скупова циклуса C_n.

Библиотека извршава два исписа:

1) За основни приказ BDD независних скупова, библиотека позива функцију prettyPrintBDDwithSons (видети одељак 6.1.3).
2) За приказ броја независних скупова у сваком чвору BDD, библиотека позива своју функцију наведену изнад (3)).

Креирање BDD независних скупова у циклусу

Потпис функције: independent_set(n)

Функција прима број чворова n у циклусу C_n и обрађује два случаја одвојено:

- Случај "1" - случај када корен улази у независан скуп. Подграф овог случаја се мора завршити у чвору $\boxed{\perp}$, због услова да не смеју постојати два узастопна чвора са истинитосном вредношћу 1.
- Случај "0" - случај када корен не улази у независан скуп. Подграф овог случаја се може завршити у оба завршна чвора.

Током обраде оба случаја, довољно је посматрати коју истинитосну вредност има отац чвора који се креира (јер је услов да не смеју постојати два узастопна чвора са истинитосном вредношћу 1).

Функција креира BDD примењујући дефиницију независних скупова циклуса, и имплементира образац (видети одељак 3.3) да чвор чији отац има истинитосну вредност 1, левом граном се прескаче на чвор који је леви син левог сина његовог оца. Овај образац у случају "0" важи од трећег нивоа (односно, од чворова којима одговара променљива x_3), до нивоа $n-2$, а у случају "1" важи од четвртог нивоа, до нивоа $n-3$. Нивое који нису покривени обрасцем функција обрађује засебно, у складу са дефиницијом независних скупова циклуса (посебно се води рачуна о последњем чвору и како се он надовезује на корен).

Приказ BDD са бројем независних скупова циклуса у сваком чвору

Потпис функције: prettyPrintBDD_IndSet(bdd)

Функција исписује BDD и број независних скупова у сваком чвору, у форми:

$$\begin{aligned}&||\ \mathrm{m}\ ||:\\ &\mathrm{nodeId\_i[node1count](nodeId\_i\_leftBranch,\ nodeId\_i\_rightBranch)}\ \ldots\end{aligned} \qquad (100)$$

где важи објашњење као за (86). Овде је node1count број решења у сваком чвору, што у проблему независних скупова тај број означава број независних скупова у сваком чвору BDD.

Верификација BDD независних скупова циклуса

Потпис функције: verify_cn_bdd(verify_n)

Функција прима као аргумент број чворова циклуса verify_n до ког се тражи верификација. Верификација се извршава за verify_n већи или једнак 3 (јер су 1 и 2 базни случајеви).

Верификација се састоји из провере величине BDD $B(f)$ (видети одељак 3.3, (55)-(57)).

Пример 36. Нека је дат циклус C_6 (видети одељак 3.3, пример 17, слику 12). Кораци за извршавање

ове библиотеке су:

1. Унети број променљивих у циклусу као вредност променљиве n у делу [*Тестирање*], линија 287:

$$n=3$$

2. Унети број променљивих у циклусу као вредност променљиве verify_n у делу [*Тестирање*], линија 315:

$$verify\_n=30$$

2. Покренути библиотеку *independent_subsets_Cn.py*.

Резултујући BDD независних скупова циклуса је дат у форми (87):

```
||1||:
|1|(4, 5)

||2||:
|4|(6, 7)    |5|(13, 3)

||3||:
|6|(8, 9)    |7|(8, 3)    |13|(14, 15)

||4||:
|8|(10, 11)    |9|(10, 3)    |14|(12, 16)    |15|(12, 3)

||5||:
|10|(2, 12)    |11|(2, 3)    |16|(12, 3)

||6||:
|12|(2, 3)

||7||:
|2|    |3|

Ukupan broj cvorova:  16
```

Применом функције за израчунавање броја решења Булове једначине $f(x_1, \ldots, x_n) = 1$ (видети одељак 6.3.1), може се израчунати колико има независних скупова у сваком чвору. BDD са израчунатим бројем независних скупова у сваком чвору је дат у форми (97):

```
||1||:
|1|[18](4, 5)

||2||:
|4|[13](6, 7)    |5|[5](13, 3)

||3||:
|6|[8](8, 9)    |7|[5](8, 3)    |13|[5](14, 15)

||4||:
|8|[5](10, 11)    |9|[3](10, 3)    |14|[3](12, 16)    |15|[2](12, 3)

||5||:
|10|[3](2, 12)    |11|[2](2, 3)    |16|[1](12, 3)
```

```
||6||:
|12|[1](2, 3)
```

Резултат верификације резултујућег BDD независних скупова циклуса је дат у наставку.

```
Velicina ciklusa | Velicina BDD | Broj nezavisnih skupova

3 |  6 |  4
4 |  8 |  7
5 |  12 |  11
6 |  16 |  18
7 |  20 |  29
8 |  24 |  47
9 |  28 |  76
10 |  32 |  123
11 |  36 |  199
12 |  40 |  322
13 |  44 |  521
14 |  48 |  843
15 |  52 |  1364
16 |  56 |  2207
17 |  60 |  3571
18 |  64 |  5778
19 |  68 |  9349
20 |  72 |  15127
21 |  76 |  24476
22 |  80 |  39603
23 |  84 |  64079
24 |  88 |  103682
25 |  92 |  167761
26 |  96 |  271443
27 |  100 |  439204
28 |  104 |  710647
29 |  108 |  1149851
30 |  112 |  1860498
```

6.5. Одређивање оптималног редоследа променљивих

Библиотека која имплементира претрагу простора свих уређења променљивих дате истинитосне табеле Булове функције зове се *variables_order.py*. Ова библиотека илуструје проналажење оптималног редоследа променљивих приступом грубе силе. Другим речима, генеришу се сви редоследи променљивих и за сваки редослед се конструише BDD и израчуна број чворова. Једноставним упоређивањем уочава се најмањи BDD и њему одговарајућ редослед променљивих.

Библиотека се састоји из два дела:

1) Први део конструише BDD за задату истинитосну табелу и уређење променљивих (не нужно природно).
2) Други део конструише BDD за задату истинитосну табелу и могућа уређења променљивих, а потом се израчунава колико BDD има чворова у сваком редоследу променљивих, и колико има

BDD са истим бројем чворова у могућим уређењима променљивих.

Први део: Конструкција BDD за задату истинитосну табелу и уређење променљивих

Потпис функције: for_variable_order(truth_table, var_order)

Овде је truth_table истинитосна табела за коју се подразумева природан редослед променљивих, а целобројни низ var_order представља жељено уређење њених променљивих. Ова функција пермутује истинитосну табелу truth_table тако да она одговара задатом уређењу променљивих (видети одељак 4.3).

За пермутовану истинитосну табелу се позива функција за конструкцију BDD (видети одељак 6.2.1), а затим се BDD исписује позивом функције prettyPrintBDDwithSons() (видети одељак 6.1.3).

Пример 37. Нека је дата истинитосна табела Булове функције 1110001011011100, и нека је задат редослед променљивих (x_3, x_1, x_2, x_4) индексима променљивих у уређењу $[3, 1, 2, 4]$. Кораци за извршавање ове библиотеке су:

1. Унети истинитосну табелу у форми бинарне ниске и индексе променљивих у жељеном редоследу у облику низа, као аргументе функције for_variable_order у делу [*Тестирање*], линија 169:

   permuted_function = for_variable_order('1110001011011100', [3, 1, 2, 4])

2. Покренути библиотеку *variables_order.py.*

Резултујући BDD са задатим уређењем променљивих је дат у форми (87):

```
Permutacija istinitosne tabele Bulove funkcije za zadati redosled
promenljivih:  1100111110100100
Broj cvorova u BDD za zadat redosled promenljivih:  9

||1||:
|1|(4, 5)

||2||:
|4|(6, 2)    |5|(7, 8)

||3||:
|6|(2, 3)    |8|(9, 3)

||4||:
|7|(2, 3)    |9|(3, 2)

||5||:
|2|    |3|
```

Други део: Претрага простора редоследа променљивих

Део за претрагу простора редоследа променљивих садржи две функције:

1) функција за креирање свих уређења променљивих, all_variable_order
2) функција за израчунавање броја чворова BDD за сваки редослед променљивих, BDDsWithNumberofNodes

Ове две функције су објашњене у наставку.

1) Потпис функције: all_variable_order(truth_table)

Ова функција креира све пермутације уређења променљивих, и за свако уређење променљивих:

- позива функцију for_variable_order из ове библиотеке која креира пермутовану истинитосну табелу према задатом уређењу променљивих;
- позива функцију за конструисање BDD (видети одељак 6.2.1) за пермутовану истинитосну табелу из 1)
- израчунава број чворова у BDD под 2).

2) Потпис функције: BDDsWithNumberofNodes(permutation_nodes_num)

Ова функција израчунава број BDD са једнаким бројем чворова у простору уређења променљивих. Другим речима, функција израчунава колико има пемутација променљивих које генеришу BDD са истим бројем чворова.

Након претраге простора уређења променљивих, резултат се исписује помоћу функције printPermutationsAndNodes, у форми:

permuted_truth_table | permuted_variables | BDD_nodes_num (101)

Пример 38. Нека је дата истинитосна табела из примера 40, 1110001011011100. Кораци за извршавање ове библиотеке су:

1. Избрисати # испред оба трострука наводника у првом делу [*Тестирање*] (видети упутство у 6.1.1)

2. Унети истинитосну табелу у форми бинарне ниске и индексе променљивих у жељеном редоследу у облику низа, као аргументе функције all_variable_order у делу [*Тестирање*], линија 198:

```
permutation_nodes_num = all_variable_order('1110001011011100')
```

3. Покренути библиотеку *variables_order.py.*

Резултат извршавања ове библиотеке дат је у наставку.

```
 Permutovana istinitosna     Permutacija redosleda          Velicina BDD
         tabela                   promenljivih

    1111010110011000             [3, 2, 4, 1]                   9
    1110110100101100             [2, 1, 3, 4]                  11
    1011100011001110             [1, 4, 3, 2]                  10
    1101110110100100             [3, 4, 2, 1]                   8
    1011110010001110             [4, 1, 3, 2]                  10
    1110011011010100             [4, 2, 3, 1]                  11
    1100101011110100             [1, 3, 2, 4]                   9
    1110101101001010             [2, 1, 4, 3]                  10
    1110011010110010             [4, 2, 1, 3]                  11
```

```
1010110011110010            [1, 3, 4, 2]              8
1011101111000010            [3, 4, 1, 2]              8
1111001110011000            [3, 2, 1, 4]             10
1101101011010100            [4, 3, 2, 1]              9
1111100101011000            [2, 3, 4, 1]              9
1101101010001110            [4, 1, 2, 3]             10
1110110101100100            [2, 4, 3, 1]             11
1111100100111000            [2, 3, 1, 4]             10
1110010010111010            [1, 2, 4, 3]             10
1101100010101110            [1, 4, 2, 3]             10
1011110010110010            [4, 3, 1, 2]              9
1110001011011100            [1, 2, 3, 4]             11
1110101101100010            [2, 4, 1, 3]             11
1010111111000010            [3, 1, 4, 2]              8
1100111110100100            [3, 1, 2, 4]              9
```

Додатно, представља се број BDD са једнаким бројем чворова у облику речника чији је сваки елемент представљен у форми:

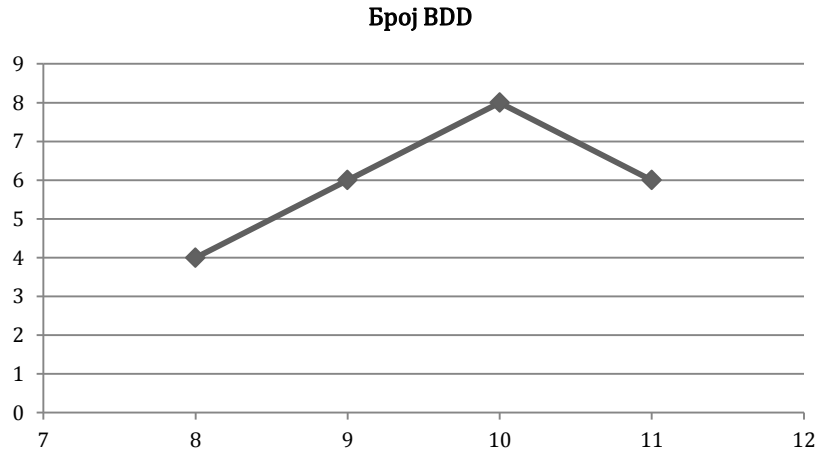
$$\text{BDD\_nodes\_num} : \text{BDD\_ num} \tag{102}$$

Резултат функције 2) је:

```
{8: 4, 9: 6, 10: 8, 11: 6}
```

што се тумачи на следећи начин: постоји укупно 4 BDD са 8 чворова, 6 са 9 чворова, 8 са 10 чворова и 6 са 11 чворова.

Слика 19 илуструје расподелу броја чворова и броја BDD из примера 38 (урађено у алату MS Excel).



Слика 19: Расподела броја чворова и броја BDD из примера 38

Ако се узме компликованија истинитосна табела са 6 променљивих,

$$1110001011011100101111001001011110111001100111110010010111001101, \tag{103}$$

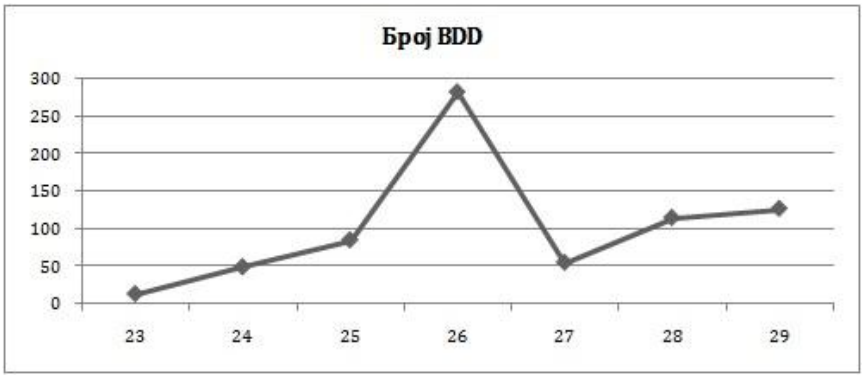
тада је резултат функције 2) једнак

```
Velicina BDD B(f): Broj BDD sa brojem cvorova B(f)

{23: 12, 24: 48, 25: 84, 26: 282, 27: 54, 28: 114, 29: 126}
```

Уочава се да минималан број чворова у BDD износи 23, и да овај број чворова има 12 BDD (односно 12 различитих уређења променљивих) од укупно 720 BDD. Другим речима, постоји више редоследа променљивих које доводе до минималног BDD. Максималну величину од 29 чворова има 126 различитих BDD. Другим речима, постоји 126 уређења променљивих које доводе до највећег броја чворова у BDD.

Слика 20 илуструје расподелу броја чворова и броја BDD истинитосне табеле (103).



Слика 20: Расподела броја чворова и броја BDD истинитосне табеле (103)

Слика 20 приказује да највећи број BDD (чак 282) имају 26 чворова, што је на средини интервала минималне и максималне величине BDD (тачније, (29+23)/2 = 26).

Слика 21 приказује расподелу броја чворова и броја BDD за две случајно генерисане истинитосне табеле, за редом $n = 7$ и $n = 8$:

$$11100100110000001101100011101101000010001001100110000101111011100111100011101111$$
$$10001011010000111100000110111000010111001100011$$

и

$$01000110100001010110000001111011101110010011011001011010011101111100011000001100$$
$$01000110111110111110000101111000010111100100110110011100000110101110001100110011$$
$$01010111100111101101101001010010010100110010101100010010110110100000110000100011$$
$$0011110100001010$$

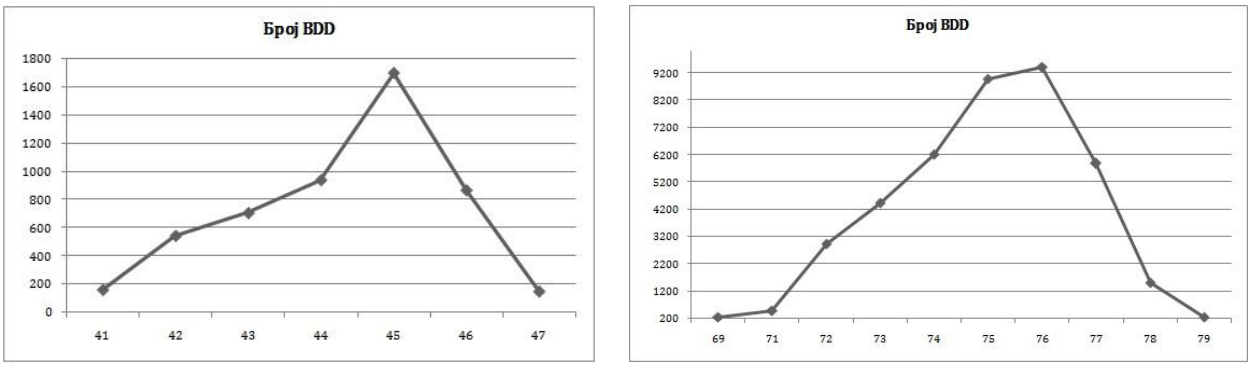
(104)

За $n = 7$:

```
Velicina BDD B(f): Broj BDD sa brojem cvorova
B(f)
{41: 156, 42: 540, 43: 702, 44: 936, 45: 1698,
46: 864, 47: 144}
```

За $n = 8$:

```
Velicina BDD B(f): Broj BDD sa brojem cvorova
B(f)
{69: 240, 71: 480, 72: 2928, 73: 4440, 74: 6216,
75: 8976, 76: 9408, 77: 5880, 78: 1512, 79: 240}
```



Слика 21: Расподела броја чворова и броја BDD за $n = 7$ и $n = 8$.

За истинитосне табеле из (104) важи да:

- за $n = 7$, минимални број чворова у BDD износи 41, и постоји укупно 156 BDD те величине. Максимални број чворова је 47, који има укупно 144 различита BDD. Највише BDD (чак 1698) имају 45 чворова ($\sim(47 + 41)/2 = 44$).
- за $n = 8$, минимални број чворова у BDD износи 69, и постоји укупно 240 BDD те величине. Максимални број чворова је 79, који има укупно 240 различита BDD. Највише BDD (чак 9408) имају 76 чворова ($\sim(69 + 79)/2 = 74$).

7. Закључак

BDD представља први избор као структура података за представљање и обраду Булових функција.

Приказана су два поступка за конструкцију BDD Булове функције на основу њене истинитосне табеле, као и алгоритми за решавање неких проблема у вези са Буловим функцијама када се зна њихов BDD. Корисност BDD за решавање неких конкретних проблема илустрована је на проблему одређивања броја независних скупова циклуса од n чворова. Алгоритми су реализовани на програмском језику *Python*.

Применљивост BDD на анализу Булових функција највише зависи од величине BDD. У вези са тим, приказане су неке фамилије Булових функција са једноставним BDD, специјално симетричне Булове функције. С друге стране, испоставља се да постоје фамилије Булових функција које за произвољан редослед променљивих имају BDD чија величина расте експоненцијално са бројем променљивих. У вези са овим проблемом, реализовано је проналажење грубом силом оптималног редоследа променљивих Булове функције, односно редоследа за које се добија најмањи BDD.

Јако интересантна тема за евентуални наставак рада у овој области свакако би могло да буде проналажење задовољавајућег редоследа променљивих применом неке од хеуристичких техника. Поред тога, постоји низ проблема који се могу решити помоћу BDD, као што су проблем осам дама, шетња витезова [9, стр. 27] и други.

8. Литература

[1] D. E. Knuth, *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*, Pearson Education 2011

[2] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design*, Springer-Verlag, Berlin Heidelberg, 1998

[3] F. Somenzi, *Binary Decision Diagrams*, http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/reading/somenzi99bdd.pdf, 1999

[4] R. E. Bryant, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on Computers, 1986

[5] M. Zivkovic , *Algoritmi*, Matematički fakultet Univerzitet u Beogradu, Beograd, 2000

[6] http://www.python.org/

[7] http://vlsi.colorado.edu/~fabio/CUDD/

[8] https://code.google.com/p/pistol/source/browse/trunk/Pistol/Polynomial.py

[9] H. R. Andersen, *An Introduction to Binary Decision Diagrams*, Department of Information Technology, Technical University of Denmark, 1997