Matematički fakultet
Univerzitet u Beogradu

DP procedura

— dokumentacija —

Studenti	Bukurov Anja	1082/2016
	Stanković Vojislav	1080/2016
Predmet	Automatsko rezonovanje	
Školska godina	2016/2017	
Nastavnik	dr Filip Marić	
Datum	17. juli 2017	

1 Opis rada

U ovom radu opsiano je kako smo implementirali DP proceduru u programskom jeziku C++. Opisali smo funckije korišćene za samu porceduru, kao i pomoćne funkcije za rukovanje literalima i klauzama.

Sama procedura implementirana je tako da redom eliminiše jednu po jednu promenljivu iz formule i ako uspe da ih sve eliminiše program vraća SAT, a u suprotom UNSAT.

2 Literali, promenljive i klauze

Literali i promenljive su predstavljeni tipom `unsigned`:

```
typedef unsigned Var;
typedef unsigned Literal;
```

Klauza je predstavljena skupom literala:

```
typedef set<Literal> Clause;
```

Formula je skup klauza:

```
typedef set<Clause> FormulaCNF;
```

Napravili smo nabrojivi tip `Polarity` koji služi za određivanje polariteta literala:

```
enum Polarity {POSITIVE, NEGATIVE};
```

2.1 Funkcije

Od funkcija za rukovanje literalima i promenljivama, implementirali smo sledeće:

Literal litFromVar(Var v, Polarity p);

- funkcija za konverziju promenljive u literal datog polariteta
- implementirano pomoću bitovskih operatora
- pozitivni literali predstavljaju se parnim brojevima, a negativni neparnim
- pozitivan literal dobija se šiftovanjem vrednosti promenljive `v` u levo
- negativan literal dobija se šiftovanjem vrednosti promenljive `v` u levo i primeniom bitovske disjunkcije na dobijenu vrednost i 1

Var varFromLit(Literal l);

- funkcija za konverziju literala proizvoljnog polariteta u promenljivu
- implementirano pomoću bitovskih operatora
- `l` se šiftuje u desno čime se odbacuje najniži bit koji je označavao polaritet literala

bool isPositive(Literal l);

- funkcija kojom se ispituje da li je literal pozitivan
- literal je pozitivan ako je broj paran, tj. ako je najniži bit 0
- provera se vrši primenom bitovske konjunkcije na `l` i 1 i negacijom dobijene vrednosti

bool isNegative(Literal l);

- funkcija kojom se ispituje da li je literal negativan
- literal je negativan ako je broj neparan, tj. ako je najniži bit 1
- provera se vrši primenom bitovske konjunkcije na `l` i 1

Literal oppositeLiteral(Literal l);

- funkcija za dobijanje suprotnog literala
- suprotan literal dobija se invertovanjem najnižeg bita tj. primenom bitovske eksluzivne disjunkcije na `l` i 1

int intFromLit(Literal l);

- funkcija za dobijanje celobrojne vrednosti od literala
- proverava se polaritet literala `l`
- ukoliko je pozitivan, na vrednost dobijenu pozivom funkcije `varFromLit(l)` dodaje se 1 i sve se kastuje u `int`
- ukoliko je negativan, na vrednost dobijenu pozivom funkcije `varFromLit(l)` dodaje se 1 i sve se kastuje u `int` a potom negira

Literal litFromInt(int i);

- funkcija za dobijanje literala od celobrojne vrednosti
- ukoliko je broj `i` pozitivan, pravi se pozitivan literal pozivom funkcije `litFromVar(i - 1, POSITIVE)`
- ukoliko je broj `i` negativan, pravi se negativan literal pozivom funkcije `-litFromVar(i - 1, NEGATIVE)`

3 Klasa DPSolve

Klasa DPSolve služi za primenu DP procedure na određenu formulu. Klasa sadrži dva privatna polja: `FormulaCNF _formula;` koje predstavlja formulu čiju zadovoljivost proveravamo i `unsigned _num;` koje predstavlja broj promenljivih u formuli.

Od javnih metoda, klasa sadrži:

DPSolve(const FormulaCNF & f, unsigned num);

- konstruktor

bool contains(const Clause & c, Literal l);

- funkcija koja proverava da li klauza c sadrzi literal l
- implementarano pomoću metode za pretragu skupa find()

bool resolution(Var v, const Clause & c1, const Clause & c2, Clause & r);

- funkcija koja primenjuje pravilo rezolucije nad klauzama `c1` i `c2` po literalu `l`
- klauza `c1` sadrži pozitivan a `c2` negativan literal `l`
- parametar `v` služi za lakše određivanje pozitivnog i negativnog literala `l`
- `r` je rezultujuća klauza koja ne sadrži `l`
- na početku, klauza `r` je jednaka klauzi `c1` a onda se iz nje briše pozitivan literal `l`
- zatim se za svaki literal `k` iz klauze `c2` proverava da li je jednak negativnom literalu `l` - njih preskačemo a za ostale proveravamo da li klauza `r` sadrži literal suprotan literalu `k`
- ako je to ispunjeno onda je klauza `r` tautologija i vraća se `false` jer ne može dalje da se primenjuje rezolucija
- ako u `r` nema suprotnog literala literalu `k` onda se on dodaje u klazu `r`

bool eliminate(Var v);

- funkcija koja uklanja iskazno slovo `v` iz formule primenom pravila rezolucije

- traže se klauze formule `_formula` koje sadrže pozitivan literal koji odgovara promenljivoj `v` i ne sadrže negativan literal koji odgovara promenljivoj `v` i takve klauze dodaje u novu formulu `nf`
- za svaku pronađenu klauzu `c1` koja sadrži pozitivan literal traži se klauza `c2` koja sadrži negativan literal koji odgovara promenljivoj `v` i nad njima se poziva funkcija `resolve(v, c1, c2, r)`
- ukoliko neki poziv funkcije `resolve(v, c1, c2, r)` vrati praznu klauzu `r` funkcija `eliminate(v)` vratiće `false` čime se označava da nema više šta da se eliminiše, u suprotnom se klauza `r` dodaje u novu formulu
- ukoliko nije dobijena nijedna prazna klauza, `eliminate(v)` vraća `true` a privatno polje `_formula` se menja u novodobijenu formulu

bool checkIfSat();

- funkcija koja proverava zadovoljivost formule
- za sve promenljive, redom, proverava se da li mogu da se eliminišu `eliminate(v)`
- ukoliko `eliminate(v)` vrati `false` to znači da je izvedena prazna klauza i formula je nezadovoljiva, a funkcija vraća `false`
- ukoliko su sve promenljive uspešno eliminisane, vraća se `true`

int skipSpaces(istream & istr);

- funkcija koja preskače beline pri učitavanju fajla u DIMACS formatu

int skipRestOfLine(istream & istr);

- funkcija koja preskače ostatak reda pri učitavanju fajla u DIMACS formatu

bool inDimacs(FormulaCNF & f, unsigned & num_of_vars, istream & istr);

- funkcija koja učitava fajl u DIMACS formatu
- prilikom čitanja se postavljaju vrednosti privatnih polja `_formula` i `_num`

4 Main funkcija

U okviru main funckije poziva se funkcija za čitanje fajla u DIMACS formatu `inDimacs(f, num, cin)`. Zatim se poziva konstruktor klase `DPSolver(f, num)`. Na kraju se poziva metoda klase `checkIfSat()` koja će vrati `true` ako je formula zadovoljiva, a u suprotnom vraća `false`. Na osnovu povratne vrednosti funkcije, na standardni izlaz se ispisuje `SAT` odnosno `UNSAT`.

5 Test primeri

U tabeli 5 prikazali smo koliko memorije program koristi za ulaze sa različitim brojem promenljivih i klauza.

Tabela 1: Primeri rada programa

promenljivih	klauza	memorija	SAT/UNSAT
729	11788	3.44 GB (Memory out)	SAT
16	60	378 KB	SAT
22	70	1.84 GB	SAT
75	325	Memory out	SAT
32	97	2.31 MB	SAT
20	44	223 KB	SAT
30	95	3.44 GB (Memory out)	UNSAT
10	28	133 KB	UNSAT
6	11	37 KB	UNSAT
24	49	338 KB	UNSAT
50	218	Memory out	UNSAT
100	430	Memory out	UNSAT