Implementacija DP procedure za iskaznu logiku

Stefan Jevtić mi241043@alas.matf.bg.ac.rs

14. novembar 2024.

Sažetak

Ovaj dokument predstavlja seminarski rad iz predmeta ,,Automatsko rezonovanje" na master studijama na Matematičkom fakultetu u Beogradu. U okviru rada prikazana je implementacija DP procedure za iskaznu logiku u programskom jeziku C++. Korišćene su heuristike za izbor promenljive koja se eliminiše radi poboljšanja efikasnosti algoritma.

Sadržaj

1 Uvod

Raznovrsni problemi računarske inteligencije se mogu svesti na SAT problem [1] i opisati formulama iskazne logike. Davis Putnam je jedna od procedura za ispitivanje zadovoljivosti iskazne formule (pod pretpostavkom da je ona zadata u pogodnom obliku). Procedura koju su predložili Davis, Putnam, Logemann i Loveland za iskaznu logiku [2, 3], poznatija kao Davis–Putnam procedura, nekada je bila glavna praktična procedura za rešavanje SAT problema.

Ona se zasniva na propagaciji jediničnih klauza, eliminaciji "čistih" literala i eliminaciji promenljivih. Poznato je da mnogi faktori utiču na performanse procedure, kao što su struktura podataka za implementaciju klauza, izbor promenljive koja se eliminiše, način implementacije propagacije jediničnih klauza, i tako dalje [4].

Ostatak ovog rada organizovan je na sledeći način. U poglavlju 2 navodimo osnovne definicije, pojmove i notaciju koja će biti korišćena u ostatku rada.
U poglavlju 3 dajemo detaljni opis algoritma koji je predmet ovog rada.
Poglavlje 4 sadrži detalje implementacije metode. Najzad, poglavlje 5 sadrži zaključna razmatranja i osvrt na metodu opisanu u radu.

2 Osnove

2.1 Uvod u Iskaznu Logiku

Iskazna logika je grana formalne logike koja se bavi manipulacijom i analizom iskaza. Iskazi su izjave koje mogu biti tačne ili netačne, ali ne i oba istovremeno. Iskazna logika koristi simbole za predstavljanje iskaza i logičkih operatora koji omogućavaju formiranje složenih logičkih izraza.

2.2 Sintaksa iskazne logike

Sintaksa iskazne logike definiše pravila za formiranje validnih iskaznih formula:

- *Atomi*: Osnovne jedinice logike, koje ćemo označavati malim latiničnim slovima (npr. p, q, r).
- *Literali*: Literali predstavljaju atome ili njihovu negaciju (npr. p, $\neg$q). Suprotan literal od literala p je literal $\neg$p.
- *Logički operatori*: negacija ($\neg$), konjunkcija ($\wedge$), disjunkcija ($\vee$), implikacija ($\rightarrow$) i ekvivalencija ($\leftrightarrow$).
- *Prioritet operatora (opadajuće)*: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- *Asocijativnost operatora*: $\wedge$ i $\vee$ su levo, a $\rightarrow$ i $\leftrightarrow$ desno asocijativni.

2.3 Semantika iskazne logike

Semantika iskazne logike se bavi dodeljivanjem istinitosnih vrednosti iskazima:

- *Valuacija*: Funkcija koja skup svih atoma preslikava u dvočlan skup {0 (netačno), 1 (tačno)} Alternativno, valuaciju možemo posmatrati kao skup atoma koje smatramo tačnim (dok su ostali netačni).
  Ponekad je zgodno pridružiti istinitosne vrednosti samo nekim atomima, dok za ostale kažemo da su nedefinisani. Takve valuacije nazivamo *parcijalnim valuacijama.*
- *Interpretacija*: Funkcija koja skup svih iskaznih formula preslikava u dvočlan skup {0 (netačno), 1 (tačno)} za datu potpunu valuaciju.
- *Modeli*: Skup valuacija u kojima je određena formula istinita.
- Dodela istinitosnih vrednosti logičkim operatorima:
  - *Negacija* (¬, NOT): Suprotna vrednost iskaza (npr. ¬p).
  - *Konjunkcija* (∧, AND): Istinita samo ako su oba iskaza istinita (npr. $p \wedge q$).
  - *Disjunkcija* (∨, OR): Istinita ako je bar jedan od iskaza istinit (npr. $p \vee q$).
  - *Implikacija* (→, IMPLIES): Istinita ako je pretpostavka netačna ili je zaključak istinit (npr. p → q).
  - *Ekvivalencija* (↔, IFF): Istinita ako oba iskaza imaju istu vrednost istinitosti (npr. $p \leftrightarrow q$).
- *Zadovoljivost*: Formula je zadovoljiva ako postoji bar jedna valuacija koja čini formulu istinitom. Na primer, formula $p \vee q$ je zadovoljiva jer postoji valuacija (npr. p je istinito) koja je čini istinitom.
- *Tautologija*: Formula je tautologija ako je istinita u svakoj valuaciji.
  Na primer, formula $p \vee \neg p$ je tautologija jer je istinita bez obzira na valuaciju atoma p.

2.4 SAT problem

SAT (Satisfiability) problem je problem određivanja da li postoji valuacija koja zadovoljava datu iskaznu formulu. SAT problem je jedan od najvažnijih problema u računarstvu i logici jer je u pitanju prvi problem za koji je dokazano da je NP kompletan [5]. To znači da su svi postojeći algoritmi za rešavanje SAT problema eksponencijalne složenosti u najgorem slučaju. Značaj SAT problema se ogleda i u tome što se mnogi drugi NP kompletni problemi mogu efikasno svoditi na SAT.

Konjunktivna normalna forma (KNF) je oblik iskazne formule u kojoj je ona predstavljena kao konjunkcija klauza. Klauza predstavlja disjunkciju literala.

U daljem tekstu pretpostavljamo da su formule koje razmatramo u KNF-u.

3 Opis metode

DP procedura funkcioniše tako što sistematski primenjuje pravilo rezolucije na skup klauza u konjuktivnoj normalnoj formi dok ne pronađe rešenje ili utvrdi da rešenje ne postoji. Algoritam se sastoji od sledećih koraka:

- *Uklanjanje tautologičnih klauza*: Prvi korak algoritma je uklanjanje svih tautologičnih klauza iz formule. Tautologična klauza je klauza koja sadrži i literal i njegovu negaciju, čineći je automatski istinitom. Uklanjanjem ovih klauza, formula se pojednostavljuje.
- *Propagacija jediničnih klauza*: Nakon uklanjanja tautologičnih klauza, algoritam traži jedinične klauze, koje sadrže samo jedan literal. Ovi literali se postavljaju kao istiniti, a sve klauze koje ih sadrže se uklanjaju iz formule. Klauze koje sadrže negaciju ovih literala se ažuriraju uklanjanjem tog literala.
- *Eliminacija "čistih" literala*: Čist literal je literal koji se pojavljuje samo u jednom obliku (pozitivnom ili negativnom) kroz celu formulu. Algoritam pronalazi i uklanja klauze koje sadrže čiste literale, jer njihovo prisustvo ne utiče na zadovoljivost formule.
- *Eliminacija promenljive*: Eliminacija promenljive je ključni deo DP procedure. Algoritam bira promenljivu nad kojoj se primenjuje pravilo eliminacije i kombinuje klauze koje sadrže taj literal i njegovu negaciju kako bi formirao nove klauze. Ovaj proces se ponavlja dok se ne pronađe prazna klauza (što ukazuje na nezadovoljivost) ili dok se skup klauza ne isprazni (što ukazuje na zadovoljivost).

3.1 Heuristike za odabir literala

U implementaciji DP algoritma, odabir promenljive za rezoluciju može značajno uticati na efikasnost. Korišćene heuristike uključuju:

- *Heuristika maksimalne frekvencije*: Odabir promenljive koja se najčešće pojavljuje u formuli, kako bi se smanjio broj klauza što je brže moguće.
- *Nasumična heuristika*: Odabir promenljive nasumično, čime se izbegava potencijalna pristrasnost i omogućava istraživanje različitih puteva ka rešenju.

3.2 Primer

Da bismo ilustrovali rad DP algoritma, razmotrićemo jednostavan primer formule u KNF obliku:

$$(p \vee \neg q) \wedge (\neg p \vee q) \wedge (p \vee q) \tag{1}$$

Koraci algoritma:

1. Uklanjanje tautologičnih klauza: Nema tautologičnih klauza.
2. Propagacija jediničnih klauza: Nema jediničnih klauza.
3. Eliminacija "čistih" literala: Nema čistih literala.

4. Eliminacija promenljive:

   - Biramo promenjivu koju ćemo eliminisati. Krećemo na primer od p .
   - Rezolviramo svaku klauzu koja sadrži literal p sa svakom klauzom koja sadrži literal ¬p.
   - Najpre rezolviramo $(p \vee \neg q)$ sa $(\neg p \vee q)$ čime dobijamo $(q \vee \neg q)$.
   - Nakon toga, rezolviramo $(p \vee q)$ sa $(\neg p \vee q)$ čime dobijamo (q).
   - Kako više nema klauza koje sadrže literale p ili $\neg p$ možemo slobodno izbaciti promenljivu p iz skupa klauza.

Trenutno stanje formule:

$$(q \vee \neg q) \wedge (q) \tag{2}$$

Naredni rekurzivni poziv algoritma će prepoznati da se u formuli nalaze tautologična $(q \vee \neg q)$ i jedinična klauza (q) koje se eliminišu. Nakon ovih eliminacija formula postaje prazan skup klauza. Zaključujemo da je polazna formula zadovoljiva i algoritam vraća true tj. SAT.

4 Implementacija

Implementacija DP procedure smeštena je u istoimenu strukturu koja sadrži javna polja i metode navedene u nastavku.

4.1 Javna polja

- std::set< Literal > *literals* - svi prisutni literali unutar formule
- std::set< Literal > *falseLiterals* - suprotni literali od jediničnih literala

4.2 Javne metode

- *void print* (const NormalForm &f)

  Ispisivanje trenutnog stanja formule u KNF obliku

- *bool isTautologicClause* (const Clause &clause)

  Ispitivanje tautologičnosti klauze

- *void removeAllTautologyClauses* (NormalForm &f)

  Eliminacija tautologičnih klauza

- *bool isUnitClause* (const Clause &clause)

  Ispitivanje jediničnosti klauze

- *void removeFalseLiterals* (NormalForm &f)

  Eliminacija suprotnih literala

- *void removeUnitClauses* (NormalForm &f, bool &conflict)

  Eliminacija jediničnih klauza

- *bool isPureLiteral* (const Literal &literal, const NormalForm &f)

  Ispitivanje "čistoće"literala

- *void removePureClausesByLiteral* (NormalForm &f, const Literal &pure-Literal)

  Eliminacija klauza koje sadrže "čist"literal

- *void removePureClauses* (NormalForm &f)

  Eliminacija "čistih"klauza

- *std::vector< Clause > allClausesWithGivenLiteral* (const NormalForm &f, const Literal &target)

  Pronalazak klauza koje sadrže dati literal

- *Clause resolve* (const Clause &first, const Clause &second, const Literal &target)

  Kreiranje nove klauze metodom rezolucije po datom literalu

- *NormalForm parse* (std::istream &fin)

  Parsiranje formule zadate u KNF-u

- *std::map< Atom, unsigned > maximumOccurrence* (NormalForm &f)

  Izračunavanje frekvencije pojavljivanja atoma u formuli

- *std::vector< Atom > atomsRandomOrder* ()

  Kreiranje nasumičnog redosleda prisutnih atoma u formuli

- *void removeVariables* (NormalForm &f, bool &conflict)

  Eliminacija promenljivih

- *bool solve* (NormalForm &f)

  Rekurzivno ispitivanje zadovoljivosti formule

4.3 Prevođenje i pokretanje projekta

Za prevođenje i pokretanje programa koristiti g++, MSVC ili cmake.
Na *Linuxu*: U terminalu se pozicionirati u željeni direktorijum i klonirati repozitorijum preko: git clone git@github.com:StefanJevtic63/ar.git
sudo apt-get update
sudo apt-get install g++
./perform-tests.sh

Na *Windowsu*: Možete koristiti CLion ili Microsoft Visual Studio ili instalirati WSL (Windows Subsystem for Linux) tako što ćete otvoriti Powershell kao administrator i pokrenuti narednu komandu: wsl –install
Zatim je potrebno ponoviti korake navedene za operativni sistem Linux.

5 Zaključak

U ovom radu smo istražili i implementirali Davis-Putnam (DP) proceduru za ispitivanje zadovoljivosti iskaznih formula. Kroz detaljni opis algoritma, prikazali smo ključne korake kao što su uklanjanje tautologičnih klauza, propagacija jediničnih klauza, eliminacija "čistih" literala, i eliminacija promenljivih.

Implementacija DP algoritma demonstrirana je na jednostavnom primeru, a pokazali smo i različite heuristike za odabir promenljive za eliminaciju. Budući rad može se fokusirati na dalju optimizaciju algoritma i istraživanje drugih heuristika koje mogu dodatno poboljšati efikasnost i skalabilnost DP procedure.

Literatura

[1] Armin Biere, Marijn Heule, and Hans van Maaren, eds. Handbook of satisfiability. Vol. 185. IOS press, 2009.

[2] Martin Davis and Hilary Putnam, A computing procedure for quantification theory. Vol. 7, pp. 201-215. Journal of the Association for Computing Machinery, 1960.

[3] Martin Davis, George Logemann and Donald Loveland, A machine program for theorem proving. Vol. 5, 394–397. Communications of the Association for Computing Machinery, 1962.

[4] Hantao Zhang and Mark. E. Stickel, Implementing the Davis-Putnam Method. Kluwer Academic Publishers, 2000.

[5] Stephen A. Cook, The complexity of theorem-proving procedures. Communications of the Association for Computing Machinery, 1971.