Implementacija DP procedure za iskaznu logiku

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

13. 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 literala po kome će se vršiti metoda rezolucije radi poboljšanja efikasnosti algoritma.

Sadržaj

1 Uvod

Raznovrsni problemi računarske inteligencije se mogu svesti na problem odlučivosti i opisati formulama iskazne logike. Davis Putnam je jenda od procedura za ispitivanje zadovoljivosti iskazne formule (pod pretpostavkom da je ona zadata u pogodnom obliku).

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 **??** 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, obično označene velikim slovima (npr. P, Q, R).
- **Literali**: Literali predstavljaju atome ili njihovu negaciju. (npr. P, ¬Q).
- **Logički operatori**:
  - **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. $P \leftrightarrow Q$).

2.3 Semantika Iskazne Logike

Semantika iskazne logike se bavi dodeljivanjem istinitosnih vrednosti iskazima:

- **Valuacija**: Proces dodeljivanja istinitosnih vrednosti atomima.

- **Istinitosne tabele**: Tablice koje prikazuju istinitosne vrednosti složenih logičkih izraza za sve moguće kombinacije atoma.
- **Modeli**: Skup valuacija u kojima je određena formula istinita.

2.4 Pojam Zadovoljivosti i Tautologičnosti

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.

2.5 SAT Problem

SAT (Satisfiability) problem je problem određivanja da li postoji valuacija koja zadovoljava datu logičku formulu. SAT problem je jedan od najvažnijih problema u računarstvu i logici jer je osnovni NP-kompletan problem.
To znači da je rešavanje SAT problema u najgorem slučaju eksponencijalno teško, ali pronalaženje rešenja za jedan SAT problem može se koristiti za rešavanje drugih NP problema.

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

Disjunktivna normalna forma (DNF) je oblik iskazne formule u kojoj je ona predstavljena kao disjunkcija konjunkcija literala.

KNF i DNF zahtevaju da se negacije spuste na nivo atoma. Dakle, nije moguće da se jedan unarni operator negacije odnosi na više literala.
Na primer, nije dozvoljeno $\neg$(P $\wedge$ Q) već se taj logički iskaz mora zapisati u obliku $\neg$P $\vee$ $\neg$Q

3 Opis metode

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

3.1 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.

3.2 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.

3.3 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.

3.4 Eliminacija promenljive

Eliminacija promenljive je ključni deo DP algoritma. Algoritam bira literal nad kojim se primenjuje pravilo rezolucije 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 sve klauze ne isprazne (što ukazuje na zadovoljivost) ili dok ne ponestane mogućnosti za izbor literala po kome se vrši pravilo rezolucije.

3.5 Heuristike za odabir literala

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

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

3.6 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 literal P za rezoluciju.
   - Formiramo novu klauzu kombinovanjem klauza $P \vee \neg Q$ i $\neg P \vee Q$, čime dobijamo $Q \vee \neg Q$, što je tautologična klauza i može se zanemariti.
   - Formula se pojednostavljuje na $(P \vee Q)$.

Rezultat: Formula je zadovoljiva jer se može zadovoljiti dodelom $P = \text{true}$ ili $Q = \text{true}$.

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**
- std::set<Literal> **falseLiterals**

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 &pureLiteral)

  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

- **bool solveRec** (NormalForm &f)

  Rekurzivno ispitivanje zadovoljivosti formule

- **bool solve** (NormalForm &f)

  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 putem rezolucije.

Implementacija DP algoritma demonstrirana je na jednostavnom primeru, a pokazali smo i primenu različitih heuristika za odabir literala. Budući rad može se fokusirati na dalju optimizaciju algoritma i istraživanje drugih heuristika koje mogu dodatno poboljšati efikasnost i skalabilnost metode.

6 Literatura

Wikipedia
Davis Putnam algoritam

Hantao Zhang, Mark. E. Stickel USA 1999.
Implementing the Davis-Putnam Method