SEMINARSKI RAD IZ AUTOMATSKOG REZONOVANJA

- Svođenje na KNF u iskaznoj logici i logici prvog reda. Cajtinova transformacija i preimenovanje formula -

Svođenje na KNF u logici prvog reda, preimenovanje formula

U ovom odeljku biće opisan postupak svođenja formule na KNF u logici prvog reda primenom preimenovanja formula. Ekvivalentat postupak postoji u iskaznoj logici – on je čak specijalan slučaj preimenovanja - i zove se Cajtinova transformacija.

Formula je u **KNF** formi ako je oblika ∀X F, gde je X skup nekih promenljivih, a F formula bez kvantifkatora koja je sastavljena od klauza (niz literala vezanih disjunkcijama) vezanih konjunkcijama. Da bi se formula **F** pretvorila u ekvizadovoljivu formulu u **KNF** formi, potrebno je primeniti sledeće procedure:

1. eliminacija konstanti
2. preimenovanje/Cajtinova transformacija
3. izvlačenje kvantifikatora ispred formule – svođenje na **preneksnu normalnu formu**
4. skolemizacija – postupak eliminacije egzistencijalnog kvantifikatora

Eliminacija konstanti

Eliminacija konstanti je postupak koji iz polazne formule **F** pravi formulu **A** koja je logički ekvivalentna sa **F**, a ne sadrži logičke konstante **T** i **ne-T**. Pravila za eliminaciju konstanti su:

- ~T ≡ ne-T
- ~ne-T ≡ T
- A & T ≡ A
- A & ne-T ≡ ne-T
- A | T ≡ T
- A | ne-T ≡ A
- A => T ≡ T
- A & ne-T ≡ ~A
- T => A ≡ A
- ne-T => A ≡ T
- A <=> T ≡ A
- A <=> ne-T ≡ ~A
- *X T ≡ T, gde je *X kvantifikacija po promenljivoj X

Preimenovanje formule

Neka je data formula **F** u logici prvog reda. Preimenovanje formule sastoji se od rekurzivne procedure koja prolazi kroz formulu **F** i radi sledeće:

- ako je **F** atomska formula, procedura vraća tu formulu
- ako je **F**:

- ~A' – nakon primene procedure na A', dobija se neka atomska formula A. Neka je **X** skup slobodnih promenljivih formule A. **F** se menja novom atomskom formulom **s**(**X**), a zatim se proširuje formulom ∀X s(X) <=> ~A(X).
- A' * B' – nakon primene procedure na A' i B', dobijaju se neke atomske formule A i B. Neka je **X** skup slobodnih promenljivih formula A i B, tj. X je unija skupova X1 i X2 koji odgovaraju slobodnim promenljivima formula A i B redom. **F** se menja novom atomskom formulom **s(X)**, nakon čega se proširuje formulom ∀X s(X) <=> A(X1) * B(X2). Operator * je bilo koji binarni operator.
- *Y (A') – nakon primene procedure na A', dobija se neka atomska formula A. Neka je **X** skup slobodnih promenljivih formule A. Formula **F** se menja novom atomskom formulom **s(X)**, nakon čega se proširuje formulom ∀X (s(X) <=> (*Y A(X))). Operator *Y je kvantifikacija po promenljivoj Y.

Formula koja se dobija preimenovanjem od formule **F** je ekvizadovoljiva sa formulom **F**.

Svođenje na preneksnu normalnu formu

Postoji nekoliko slučajeva:

1. F ≡ ~(∀X P) ili F ≡ ~(∃X P) - formula postaje redom ∃X ~P ili ∀X ~P
2. F ≡ (+X P) * Q – ukoliko se X ne pojavljuje kao slobodna promenljiva u Q, moguće je Q staviti pod kvantifikator +. Operator * je disjunkcija ili konjunkcija. U slučaju da se X javlja kao slobodna promenljiva u Q, onda je potrebno prethodno izvršiti preimenovanje vezane promenljive X u neku promenljivu koja se ne javlja u Q.
3. F ≡ (+X P) => Q – formula se najpre transformiše u ~(+X P) ∨ Q. Sada se mogu primeniti pravila 1. i 2. da se izvuče kvantifikator ispred formule.
4. F ≡ Q => (+X P) – ukoliko se X ne pojavljuje kao slobodna promenljiva u Q, formula je ekvivalentna sa +X (Q => P). U suprotnom, vezana promenljiva X se preimenuje u neku promenljivu koja se ne javlja u Q.
5. F ≡ (+X P) <=> Q – formula se transformiše u ((+X P) => Q) ∧ (Q => (+X P))

Skolemizacija

Postupak skolemizacije podrazumeva eliminaciju egzistencijalnog kvantifikatora iz formule. Egzistencijalni kvantifikatori se eliminišu s leva na desno. Postoje dva slučaja:

1. pre egzistencijalnog kvantifikatora nema nijednog univerzalnog – tada se uvodi novi konstanti term **c**, a zatim se kvantifikovana promenljiva menja termom **c**
2. pre egzistencijalnog kvantifikatora postoji niz univerzalnih kvantifikatora po promenljivama iz skupa **X**. Tada se uvodi novi term **f**, a zatim se kvantifikovana promenljiva menja termom **f(X)**

Rezultujuća formula ne sadrži više egzistencijalni kvantifikator.

Implementacija

Implementacija postupka sprovedena je u programskom jeziku C++. Funkcija koja sprovodi prevođenje u KNF formu je:
cnf_transform(const Formula& f, std::vector<Clause>& CNF, std::list<Quant>& quants)

KNF forma se predstavlja pomoću vektora klauza i liste kvantifikatora. Argumenti CNF i quants su reference na vektore u koje će biti smešten rezultat procedure. Klauze su predstavljene kao niz vektora literala, a literali su predstavljeni strukturom Literal koja u sebi sadrži formulu (atomsku) i jednu **bool** promenljivu koja označava da li je atom negiran ili ne. Kvantifikatori su predstavljeni pomoću terma (promenljive) po kojem se kvantifikuje.

Procedura pretvaranja formule u njoj ekvizadovoljivu formulu u KNF formi sproveo sam na malo modifikovani način u odnosu na gore opisanu proceduru. Modifikacije se sastoje od sledećeg:

1. nakon jednog koraka preimenovanja, vrši se preimenovanje u novododatoj formuli tako da pod uticajem univerzalnog kvantifikatora budu skroz nove promenljive. Ovo omogućava da se kvantifikatori mogu već u tom koraku izvući ispred cele formule, budući da se u ostatku formule nikada neće javiti iste promenljive,
2. kada se vrši preimenovanje formule oblika ∃X P, u koraku preimenovanja se vrši eliminacija egzistencijalnog kvantifikatora. Pošto svaka novododata formula stvara kvantifikatore koji idu na kraj liste kvantifikatora, to znači da se u narednim koracima preimenovanja neće pojaviti neki kvantifikator pre ovog egzistencijalnog. Zahvaljujući tome, moguće je zameniti egzistencijalni kvantifikator sa onim što je u listi kvantifikatora u trenutku njegove eliminacije.

Dobijena formula biće ekvivalentna onoj koja bi se dobila primenom standardnog postupka svođenja na KNF.

U daljem tekstu opisani su detalji implementacije na nivou jezika C++.

Funkcija cnf_transform sastoji se od tri dela: eliminacija konstanti, poziv funkcije tseitin_helper i dodavanje poslednjeg literala u KNF formulu. Funkcija tseitin_helper će zapravo odraditi sav postupak svođenja formule na KNF, osim poslednjeg atoma (u primeru gore je to atom s2) koji treba dodati na kraj liste. Iz tog razloga postoji treći korak koji dodaje i taj atom u KNF.

Funkcija tseitin_helper radi rekurzivno modifikovani postupak preimenovanja i skolemizacije. Postoji nekoliko slučajeva koje funkcija obrađuje:

1. atomska formula – vraća se ta formula
2. negacija formule – nakon rekurzivnog poziva, dobija se atom **t.** Nakon izdvajanja promenljivih iz t, kreira se atomska formula **ret** sa tim promenljivama. Nakon toga se poziva funkcija set_new_vars_unary koji kreira nove promenljive koje se nikada nisu javile u formuli do tada. Sa tim novim promenljivama se kreira atom **a** i CNF se proširuje izrazima sa tim atomom. Recimo, preimenovanje formule ~p(X) bi kao rezultat dalo formulu ekvivalentnu (!X1. s1(X1) <=> p(X1)) & s1(X). **ret** odgovara izrazu s1(X), a **a** odgovara izrazu s1(X1). Na listi CNF bi se našla formula ekvivalentna sa (!X1. s1(X1) <=> p(X1)), a tseitin_helper bi vratio vrednost **ret** koju treba dodati na kraj liste
3. binarni operatori – postupa se analogno sa negacijom, samo su izrazi koji se dodaju u CNF različiti i poziva se set_new_vars_binary umesto set_new_vars_unary
4. kvantifikovana formula – nakon rekurzivnog poziva, dobija se atom **t**, a potom se poziva funkcija **resolve_quant** da obradi slučaj kvantifikovane promenljive. Ova funkcija će napraviti atomsku formulu **ret** koju treba vratiti kao povratnu vrednost funkcije, atomsku formulu **a** i atomsku formulu **t1**. Objasnićemo postupak na primeru preimenovanja univerzalnog kvantifikatora (egzistencijalni se radi slično). Formula koju ćemo preimenovati ima sledeći

oblik: !X. p(X, Y), gde su Y slobodne promenljive formule. Uvodi se nova atomska formula s i formula !Y. (!X p(X, Y)) => s(Y). Ona je ekvivalentna sa !Y ((?X. ~p(X, Y) | s(Y)) & (!X. p(X, Y) | ~s(Y)). Trenutna vrednost promenljive **t** odgovara atomu **p(X, Y)**. Mi kreiramo novi atom **t1** koji je zapravo kopija atoma **p(X, Y)**, kreiramo atom **ret** koji predstavlja s(Y) i atom **a** koji trenutno ima vrednost s(Y). Prvo izvdojimo slobodne promenljive **Y** u vektore **new_vars** i **new_vars1** i nakon toga istovremeno preimenujemo u **t** i **t1** sve promenljive iz **Y** tako da imaju imena koja se nikada nisu javila u formuli. Te nove promenljive se dodaju na kraj vektora **quants** i njima se preimenuju pojavljivanja Y u **a**. Sada u oba vektora dodajemo promenljivu **X**, ako je ona uopšte i bila deo formule **p**, i posebno obrađujemo svaku od novonastalih klauza. U slučaju klauze ?X. ~p(X, Y) | s(Y), koju prvu obrađujemo, zamenjujemo pojavljivanja **X** sa novim funkcijskim simbolom koji se nikada nije javio u formuli, i kome su promenljive iz vektora **quants** koji čuva sve univerzalne kvantifikatore koji su se do sada javili u formuli. U slučaju !X. p(X, Y) | ~s(Y), preimenovaćemo **X** novom promenljivom i dodati je na kraj vektora **quants**. Nakon poziva funkcije **resolve_quant**, **t** predstavlja **p(X_new, Y_new)**, **t1 p(F(...), Y_new)**, **a s(Y_new)**. Sve što treba je dodati nove klauze u skladu sa vraćenim vrednostima.

Pokretanje programa

Program se kompajlira tako što se u komandnoj liniji pokrene naredba "make" na sledeći način:

$ make

Nakon kompilacije programa, može se pokrenuti izvršni fajl na sledeći način, nakon čega se ulazna formula unosi u komandnu liniju:

$ ./main

Ukoliko korisnik želi da učita formulu sa nekog ulaznog fajla (recimo, "input.txt"), može to uraditi na sledeći način:

$ cat input.txt | ./main

Ulazni jezik

Ulaz programa je formula u iskaznoj ili logici prvog reda, praćena znakom ; koji se koristi kao terminator ulaza. Sintaksu formule opisana je na sledeći način:

- <symbol> - bilo koji identifikator koji počinje **malim** slovom, regularni izraz od <symbol> je : [a-z][a-zA-Z_0-9]*
- <var> - bilo koji identifikator koji počinje **velikim** slovom, regularni izraz od <var> je: [A-Z] [a-zA-Z_0-9]*
- <term> -
  1. <var> -- promenljiva
  2. <symbol> -- konstanta
  3. <symbol>(<term_seq>) -- funkcijski term
- <atomic_formula> -

1. true -- logička konstanta T
2. false – logička konstanta ne-T
3. <symbol> -- iskazna promenljiva
4. <symbol>(<term_seq>) -- složeni atom
5. <term> = <term> -- jednakost
6. <term> ~= <term> -- različitost

- <term_seq> -
  1. <term> -- jedan term, kraj niza
  2. <term>, <term_seq> -- term praćen nizom termova, suštinski niz termova
- <formula> -
  1. <formula> <=> <formula> -- ekvivalencija
  2. <formula> => <formula> -- implikacija
  3. <formula> | <formula> -- disjunkcija
  4. <formula> & <formula> -- konjunkcija
  5. ~<formula> -- negacija
  6. !<var>. <formula> -- univerazlni kvantifikator
  7. ?<var>.<formula> -- egzistencijalni kvantifikator
  8. <atomic_formula> -- atomska formula
  9. (<formula>) -- formula u zagradama

Beline se ignorišu.

Prioritet operatora od najnižeg ka najvišem jeste:

1. kvantifikatori
2. binarni operatori:
   1. <=>
   2. =>
   3. |
   4. &
3. unarni operatori:
   1. ~

Zagrade se mogu koristiti da se promeni prioritet operatora.

Operatori | i & su levo asocijativni, a operatori <=> i => su desno asocijativni.