

# SEMINARSKI RAD IZ AUTOMATSKOG REZONOVANJA

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

### 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 učitava 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.  $\langle \text{formula} \rangle \Leftrightarrow \langle \text{formula} \rangle$  -- ekvivalencija
2.  $\langle \text{formula} \rangle \Rightarrow \langle \text{formula} \rangle$  -- implikacija
3.  $\langle \text{formula} \rangle \mid \langle \text{formula} \rangle$  -- disjunkcija
4.  $\langle \text{formula} \rangle \& \langle \text{formula} \rangle$  -- konjunkcija
5.  $\sim \langle \text{formula} \rangle$  -- negacija
6.  $!\langle \text{var} \rangle. \langle \text{formula} \rangle$  -- univerzalni kvantifikator
7.  $?\langle \text{var} \rangle. \langle \text{formula} \rangle$  -- egzistencijalni kvantifikator
8.  $\langle \text{atomic\_formula} \rangle$  -- atomska formula
9.  $(\langle \text{formula} \rangle)$  -- formula u zagradama

Beline se ignorišu. Važno je napomenuti da se kvantifikovana formula koja je potformula neke druge formule mora navesti u zagradama. U suprotnom, može se kao rezultat programa dobiti **Segmentation fault**.

Prioritet operatora od najnižeg ka najvišem jeste:

1. kvantifikatori
2. binarni operatori:
  1.  $\Leftrightarrow$
  2.  $\Rightarrow$
  3.  $\mid$
  4.  $\&$
3. unarni operatori:
  1.  $\sim$

Zagrade se mogu koristiti da se promeni prioritet operatora.

Operatori  $\mid$  i  $\&$  su levo asocijativni, a operatori  $\Leftrightarrow$  i  $\Rightarrow$  su desno asocijativni.

### 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  $\forall X F$ , gde je  $X$  skup nekih promenljivih, a  $F$  formula bez kvantifikatora 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 podrazumeva 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:

- $\sim T \equiv \text{ne-}T$
- $\sim \text{ne-}T \equiv T$
- $A \& T \equiv A$
- $A \& \text{ne-}T \equiv \text{ne-}T$
- $A | T \equiv T$
- $A | \text{ne-}T \equiv A$
- $A \Rightarrow T \equiv T$
- $A \& \text{ne-}T \equiv \sim A$
- $T \Rightarrow A \equiv A$
- $\text{ne-}T \Rightarrow A \equiv T$
- $A \Leftrightarrow T \equiv A$
- $A \Leftrightarrow \text{ne-}T \equiv \sim A$
- $*X T \equiv T$ , 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**:
  - $\sim A'$  – 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  $\forall X s(X) \Leftrightarrow \sim A(X)$ .
  - $A' * B'$  – 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  $X_1$  i  $X_2$  koji odgovaraju slobodnim promenljivima formula  $A$  i  $B$  redom. **F** se menja novom atomskom formulom  $s(X)$ , nakon čega se proširuje formulom  $\forall X s(X) \Leftrightarrow A(X_1) * B(X_2)$ . 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  $\forall X (s(X) \Leftrightarrow (*Y A(X)))$ . 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.  $F \equiv \sim(\forall X P)$  ili  $F \equiv \sim(\exists X P)$  - formula postaje redom  $\exists X \sim P$  ili  $\forall X \sim P$
2.  $F \equiv (+X P) * Q$  – 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.  $F \equiv (+X P) \Rightarrow Q$  – formula se najpre transformiše u  $\sim(+X P) \vee Q$ . Sada se mogu primeniti pravila 1. i 2. da se izvuče kvantifikator ispred formule.
4.  $F \equiv Q \Rightarrow (+X P)$  – ukoliko se  $X$  ne pojavljuje kao slobodna promenljiva u  $Q$ , formula je ekvivalentna sa  $+X (Q \Rightarrow P)$ . U suprotnom, vezana promenljiva  $X$  se preimenuje u neku promenljivu koja se ne javlja u  $Q$ .
5.  $F \equiv (+X P) \Leftrightarrow Q$  – formula se transformiše u  $((+X P) \Rightarrow Q) \wedge (Q \Rightarrow (+X P))$

### 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  $\exists X P$ , 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 svodenja 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 rekursivno modifikovani postupak preimenovanja i skolemizacije. Postoji nekoliko slučajeva koje funkcija obrađuje:

1. atomska formula – vraća se ta formula
2. negacija formule – nakon izdvajanja promenljivih, 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  $\sim p(X)$  bi kao rezultat dalo formulu ekvivalentnu  $(!X1. s1(X1) \Leftrightarrow p(X1)) \ \& \ s1(X)$ . **ret** odgovara izrazu  $s1(X)$ , a **a** odgovara izrazu  $s1(X1)$ . Na listi CNF bi se našla formula ekvivalentna sa  $(!X1. s1(X1) \Leftrightarrow p(X1))$ , 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 – kao i kod negacije, kreiraju se **ret** i **a** i **b**. Jedina razlika je u načinu na koji se izvlače slobodne promenljive. Prvo, potrebno je u odgovarajućim trenucima eliminisati kvantifikovanu promenljivu iz vektora. Drugo, metod `set_new_vars_quant` postavlja nove promenljive, ali radi i skolemizaciju. Metod prihvata vektor promenljivih, listu kvantifikatora, formulu koju treba preimenovati (**t**), kvantifikovanu promenljivu i jednu **bool** promenljivu koja označava da li je u pitanju egzistencijalni ili univerzalni kvantifikator. Ovaj metod treba primeniti dvaput, budući da će dve nove klauze biti kvantifikovane različitim kvantifikatorima. Naime, formula  $(!X \ P) \Leftrightarrow Q$  je ekvivalentna sa  $?X(!P \vee Q) \wedge !X(!Q \vee P)$ . Sada se u levom i desnom operandu izmene promenljive:  $!X1 \ ?X2 ((!P \vee Q) \wedge (!Q \vee P))$ . Skolemizacija menja  $X2$  u  $F1(X1)$ . Naša procedura radi upravo ovo, kada umesto  $P$  stavimo trenutnu formulu, a umesto  $Q$  stavimo  $S(Y)$  ( $Y$  su slobodne promenljive u  $\{!X \ P\}$ ). Analogno se radi i za egzistencijalni kvantifikator. U opštem slučaju, ako je  $+X \ P(X, Y)$  neka kvantifikovana formula, ona će biti zamenjena sa  $!Y \ ( \ (+X(P(X, Y)) \Leftrightarrow s(Y) )$ . Formula se dalje transformisati u konjunkciju dve implikacije. Sada možemo primetiti da će jedan od konjunkt biti oblika  $!Z \ ?Q \ F(Z, Y)$ , gde su  $Z$  vezane promenljive, a  $Y$  slobodne. Drugi konjunkt neće imati egzistencijalne kvantifikatore. Iz tog razloga je potrebno obraditi egzistencijalni kvantifikator tek kada se obrade promenljive sa univerzalnim. U implementaciji je to izvedeno tako što su u **old\_vars** i **old\_vars1** kopirane sve promenljive bez  $X$ , a zatim se  $X$  dodaje na kraj oba vektora. Zatim se vrši istovremeno preimenovanje i skolemizacija dvaput (za svaku od novih klauza). Na kraju se briše poslednji element vektora **old\_vars** i **old\_vars1**, budući da on odgovara promenljivoj  $X$  koja se neće pojaviti u  $s$ .