Svođenje problema ispitivanja zadovoljivosti baznih EUF formula na SAT Akermanovom redukcijom

Seminarski rad u okviru kursa
Automatsko rezonovanje
Matematički fakultet

Una Stanković, Uroš Stegić
una_stankovic@yahoo.com, mi10287@alas.matf.bg.ac.rs

18. septembar 2017.

Sadržaj

1 Uvod

Projekat opisan u nastavku predstavlja seminarski rad iz predmeta Automatsko rezonovanje, na master studijama Informatike. Ovaj dokument je celina, zajedno uz kôd javno dostupan na adresi: https://github.com/uros-stegic/simple-reason, i čini njegovu dokumentaciju.

U okviru ovog projekta, razvijeno je sintaksno okruženje za rad sa iskaznom logikom i logikom prvog reda. Definicija jezika će biti predstavljena u sekciji 3. Pored samog jezika, razvijen je interpreter koji izvršava kôd.

Obim seta alata potrebnih za primenu Akermanove redukcije se vremenom povećao, tako da ceo projekat "simple reason" sada predstavlja radni okvir koji podržava sintaksu iskazne, predikatske i jednakosne logike sa neinterpretiranim funkcijama, kao i osnovne transformacije nad sintaksičkim stablima.

U nastavku, biće predstavljena Akermanova redukcija, a potom dati uputstvo za korišćenje softvera, kao i kratak pregled dizajna i arhitekture.

2 Akermanova redukcija

U ovom delu ukratko će biti predstavljene teorijske osnove vezane za Akermanovu redukciju, koje se oslanjaju na poznavanje osnovnih pojmova iz Automatskog rezonovanja, kao što su: predikatska logika, negaciona normalna forma, funkcijski simboli i ostalo gradivo pokriveno kursom.

Određeni podskup predikatske logike se može svesti na SAT problem redukcijom polazne formule zapisane u jeziku prvog reda na ekvizadovoljivu formulu iskazne logike. Jedan način da se izvede ovakvo svođenje je postupak **Akermanove redukcije**.

Podskup logike prvog reda nad kojima je moguće izvršiti Akermanovu redukciju je:

$$L = \{\neg, \wedge, \vee, =, \neq, f_i/\alpha_i, p_i/\beta_i\}$$

pri čemu su f_i neitrerpretirani funkcijski simboli arnosti α_i, p_i predikatski simboli arnosti β_i. Simbol $=$ ($\neq$) se interpretira na uobičajen način: predikati se evaluiraju kao tačni ako su objekti jednaki (nejednaki). Važno je napomenuti da se Akermanova redukcija može primeniti isključivo nad formulama koje *nisu kvantifikovane*.

Algoritam Akermanove redukcije

Ulaz: Nekvantifikovana formula jednakosne logike sa neinterpretiranim funkcijskim simbolima. **Izlaz:** Nekvantifikovana formula jednakosne logike bez neinterpretiranih funkcijskih simbola. **Algoritam**

1. Prevesti ulaznu formulu u negacionu normalnu formu

2. Zameniti termove koji predstavljaju funkcijske simbole jedinstvenim identifikatorima (počev od unutrašnjih ka spoljašnjim)
   primer:

$$f(f(x)) = 1 \vee f(x) \neq 2 \mapsto f_2 = 1 \vee f_1 \neq 2$$
$$smene : f_1 = f(x), f_2 = f(f_1)$$

3. za svaki funkcijski simbol iz smene, za svaki par argumenata dodati aksiomu funkcionalne konzistentnosi (*functional consistency axiom*):

$$(x = f_1 \implies f_2 = f_1) \implies f_2 = 1 \vee f_1 \neq 2$$

Primer

Ulaz: Formula F bez kvantifikatora u jednakosnol logici sa neinterpretiranim funkcijama:

$$x1 = x_2 \mapsto f(x_1) \neq f(x_2) \vee f(x_1) \neq f(x_3)$$

1. Transformisati F u negacionu normalnu formu
2. Zameniti funkcijske termove jedinstvenim identifikatorima iznutra-ka-spolja

$$x_1 = x_2 \mapsto f_1 \neq f_2 \vee f_1 \neq f_3$$

   Tada:

$$f_1 = f(x_1),$$
$$f_2 = f(x_2),$$
$$f_3 = f(x_3)$$

3. Dodavanjem aksiome funkcionalne konzistentnosti za svaki par argumenata od f dobijamo:

$$(x_1 = x_2 \mapsto f_1 = f_2)$$
$$\wedge(x_1 = x_3 \mapsto f_1 = f_3)$$
$$\wedge(x_2 = x_3 \mapsto f_2 = f_3)$$
$$\mapsto (x_1 = x_2 \mapsto f_1 \neq f_2 \vee f_1 \neq f_3)$$

3 Uputstvo za korišćenje

Prilikom kreiranja projekta *Simple reason*, omogućena su dva načina korišćenja programa:

- intreraktivni mod (REPL, read-evaluate-print-loop)
- izvršavanjem skript datoteke

Oba načina biće predstavljena u nastavku, ali najpre će biti prikazana sintaksa jezika, kao i naredbe okruženja, kojima vršimo različite transformacije nad zadatom formulom.

Sintaksa jezika obuhvata:

Izrazi	
$\neg F$	$\sim F$
$F_1 \wedge F_2$	$F1\&F2$
$F_1 \vee F_2$	$F1\|F2$
$F_1 \implies F_2$	$F1 => F2$
$F_1 \Leftrightarrow F_2$	$F1 <=> F2$
$\forall x. \Leftrightarrow F$	$Ax.F$
$\exists x. \Leftrightarrow F$	$Ex.F$
$pred.symbols$	$[pqr][0-9]*$
$vars$	$[xyz][0-9]*$
UF	$[fgh][0-9]*$
$const$	$[abc1-9][0-9]*$

Naredbe okruženja:

- **cnf F** - vraća kopiju formule F koja je u KNF
- **nnf F** - vraća kopiju formule F koja je u NNF
- **c_distr F** - vraća kopiju formule F kojoj su konjunkcije spuštene u AST-u.
- **d_distr F** - vraća kopiju formule F kojoj su disjunkcije spuštene u AST-u.
- **n_distr F** - vraća kopiju formule F kojoj su negacije spuštene do listova AST-a.
- **simplify F** - vraća kopiju formule F koja je pojednostavljena[1].
- **no_imps F** - vraća kopiju formule F iz koje su eliminisane implikacije i ekvivalencije
- **quant_up F** - vraća kopiju formule F u kojoj su izdignuti svi kvantifikatori do korena AST-a.
- **prenex F** - vraća kopiju formule F koja je u preneks normalnoj formi.
- **ack F** - primenjuje Akermanovu redukciju nad kopijom formule F.
- **$a := F** - deklariše novu meta promenljivu u okruženju.
- **delete $a** - uklanja meta promenljivu iz okruženja.

3.0.1 Primer upotrebe

Interaktivni mod U ovom delu će biti predstavljen primer korišćenja *Simple reason* softvera u interaktivnom modu:

[1] više o ovim transformacijama na: http://poincare.matf.bg.ac.rs/ filip/ar/ar-iskazna-logika.pdf

```
> $a := Ax.Ey.p(x) => q(y)
> $a
Ax.Ey. p(x) => q(y)
> nnf $a
Ax.Ey.~p(x) | q(y)
> quit

Quiting.
```

Skript mod Osim interaktivnog omogućen je i skript mod, najpre u svrhu mogućnosti jednostavnog izvođenja testova, kao i da bi se korisniku omogućilo da na lakši način koristi program.
Osnovna ideja je da korisnik piše skript u datoteci koji kasnije prosleđuje programu kao argument komandne linije. Pogledajmo primer ovakve upotrebe:

Datoteka "primer1.fml"

```
$a := p(x)
$b := q(x)
nnf $a => $b
```

Pokretanje iz komandne linije

```
[user@pc code]$ ./simple-reason primer1.fml
Ax.Ey.~p(x) | q(y)

Quiting.
```

4 Pregled dizajna i arhitekture

Dizajn softvera je analogan onome koji je rađen na vežbama. Razlikuje se po tome što su sve vrednosti deklarisane kao konstantne kako bismo se približili funkcionalnom aspektu jezika C++.
Sve transformacije nad formulama su izdvojene u zasebne celine u maniru uzorka projektovanja "posetilac". Za posetioca je odlučeno zato što omogućava definisanje nove operacije bez izmene klasa elemenata nad kojima se izvršava. Formule koje se posećuju su imutabilne strukture podataka i svaka transformacija formule vraća novo sintaksno stablo.

4.1 Organizacija projekta

Projekat je organizovan u nekoliko foldera:

1. include - koji sadrži .hpp fajlove i sastoji se od podfoldera:
    - formulae - u kome su deklarisani atomička formula, formula, binarne i unarna operacija, konstante, promenljive i kvantifikatori, itd.,
    - transformations - u kome su deklarisane moguće transformacije nad elementima formule, kao što su: izvlačenje kvantifikatora, distribucija konjunkcije, negacije, prebacivanje u KNF, NNF, PRENEX, akermanova redukcija, itd.,

2. src - koji sadrži .cpp fajlove i sastoji se od:
    - podfoldera formulae,
    - podfoldera transformations - u kojima su definicije funkcija,
    - fajla main.cpp,
3. parser - u kome se nalaze fajlovi parser.l i lexer.ypp,
4. tests - u kome se nalaze .fml fajlovi koji služe za testiranje projekta pri pokretanju iz komandne linije,
5. docs - u kome se nalazi projektna dokumentacija.

    Za ovakvu organizaciju strukture fajlova smo se odlučili jer nam nudi najveću preglednost i jednostavnost pri dodavanju novih fajlova i doprinosi lakšem razumevanju.

5 Zaključak

Najveći doprinos ovog rada predstavlja upravo program, koji na jasan način prikazuje šta se dobija primenom Akermanove redukcije na polaznu formulu. Osim toga, kôd može biti korišćen i za prikaz raznih drugih transformacija nad polaznom formulom (svođenje na PRENEX, NNF,itd.), što zainteresovanom korisniku može biti korisno.