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

U okviru ovog projekta, razvijeno je sintaksno okruženje za rad sa iskaznom logikom i logikom prvog reda. Definicija jezika će biti predstavljena u sekciji 2. Pored samog jezika, razvijen je interpreter koji izvršava kôd. Obim seta alata potrebnih za primenu same 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.

1.1 Akermanova redukcija

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 siboli 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*):

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

2 Uputstvo za korišćenje

Simple reason je moguće koristiti na dva načina:

- intreraktivni mod (REPL, read-evaluate-print-loop)

- izvršavanjem skript datoteke

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.

Primer upotrebe

Prikaz jednog primera korišćenja simple reason softvera u interaktivnom modu:

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

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

```
Ax.Ey.~p(x) | q(y)
> quit

Quiting.
```

U skript modu, 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.
```

3 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". Formule koje se posećuju su imutabilne strukture podataka i svaka transformacija formule vraća novo sintaksno stablo.