Eliminácia pravidla rezu v sekventovom kalkule LK^h

Bakalárska práca

Omar Al-Shafe´i

Ing. Ján Komara, PhD.

Fakulta matematiky, fyziky a informatiky

Obsah

Úvod

Čo je to LK^h?

- Sekventový kalkulus LK^h je neštandardná verzia kalkulu LK.
- Používa Henkinove konštanty na mieste voľných premenných. (*Eigenvariable conditions* môžu byť porušené.)
- Správnosť každého odvodzovacieho pravidla LK^h závisí od jeho aktívnych formúl a nie od jeho kontextu (silná lokalita).

Cieľ práce

- Implementácia algoritmu eliminácie pravidla rezu v sekventovom kalkule LK^h bez použitia regularizácie, t. j. opravenia *eigenvariable conditions*, v paradigme deklaratívneho programovania.
- Algoritmus implementujeme na základe metódy spomenutej v článku Ján Komara. Efficient elimination of Skolem functions in LK^h. Archive for Mathematical Logic, 61(3):503–534, May 2022.

Teoretické východiská

Henkinove konštanty

- Henkinove konštanty majú tvar $c_{\exists x\, A}$ alebo $c_{\forall x\, A}$.
- Čisté formuly neobsahujú Henkinove konštanty.

Henkinove axiómy

- $\exists x\, A \to A_x\,[c_{\exists x\, A}]$
- $A_x[c_{\forall x\, A}] \to \forall x\, A$

Veta

- Čistá formula je logicky platná práve vtedy, keď je pravdivá v každej prvorádovej štruktúre, ktorá spĺňa Henkinove axiómy.

Teoretické východiská

Sekventový kalkulus LK^h

Odvodzovacie pravidlá

$$\mathrm{Ax}\ \frac{}{A,\ \Gamma \Rightarrow \Delta,\ A}\ (A \text{ je atomická}),$$

$$\mathrm{L}\exists\ \frac{A_x\left[\mathrm{c}_{\exists x\, A}\right],\ \Gamma \Rightarrow \Delta}{\exists x\, A,\ \Gamma \Rightarrow \Delta}, \qquad \mathrm{R}\forall\ \frac{\Gamma \Rightarrow \Delta,\ A_x\left[\mathrm{c}_{\forall x\, A}\right]}{\Gamma \Rightarrow \Delta,\ \forall x\, A},$$

$$\mathrm{Cut}\ \frac{\Gamma \Rightarrow \Delta,\ A \qquad A,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}.$$

Základné miery na dôkazoch

- Hĺbka $|\pi|$ dôkazu π je počet vrcholov v jeho najdlhšej vetve.
- Rezová hodnosť $\mathrm{r}\,(\pi)$ dôkazu π je suprémom hĺbok všetkých jeho rezových formúl.

Návrh riešenia

Vstupno-výstupná podmienka eliminačného algoritmu

- Vstupom je dôkaz π čistého sekventu (t. j. bez Henkinových konštánt).
- Výstupom je bezrezový dôkaz ρ toho istého sekventu s hľbkou

$$|\rho| \leq \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{|\pi|}}}}}}_{2\mathrm{r}(\pi)\text{ opakovaní } 2} .$$

Poznámka

- Štandardný eliminačný algoritmus pre LK má odhad

$$|\rho| \leq \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{|\pi|}}}}}}_{\mathrm{r}(\pi)\text{ opakovaní } 2} .$$

Návrh riešenia

Eliminačný algoritmus

- Algoritmus postupne eliminuje rezy najprv s rezovou hodnosťou $\mathrm{r}(\pi)$, a nahradí ich rezami s nižšou rezovou hodnosťou aspoň o 1, potom eliminuje rezy s rezovou hodnosťou $\mathrm{r}(\pi) - 1$, ... a nakoniec eliminuje rezy s rezovou hodnosťou 1.
- Jeden krok zníženia rezovej hodnosti dôkazu π aspoň o jedna pozostáva z dvoch krokov, pričom sa vytvorí dôkaz ρ:
  - Kváziregularizácia dôkazu π: oprava *eigenvariable condition* pre kvantifikačné rezy s najvyššou rezovou hodnosťou. Dostaneme dôkaz σ:
    $$|\sigma| < 2^{|\pi|}.$$
  - Štandardná redukcia rezovej hodnosti regulárneho dôkazu σ. Dostaneme dôkaz ρ:
    $$|\rho| < 2^{|\sigma|} < 2^{2^{|\pi|}}.$$

Implementácia

- Algoritmus implementujeme v typovej verzii multi-paradigmatického programovacieho jazyka Racket.
- Racket je dialektom programovacieho jazyka Lisp/Scheme.
- Dôkazy vizualizujeme pomocou tablovej metódy.

```
(: eliminate/all (-> Proof Integer Proof))
(define (eliminate/all p r)
    (cond
        [(> r 0)
        (eliminate/all (eliminate/rcuts
        (quasiregularization p r)  r)
        (- r 1))]
        [else p]))

(: eliminate (-> Proof Proof))
(define (eliminate p)
    (eliminate/all p (cut-rank p)))
```

Testovanie

- Vstupný dôkaz π má tvar:

```
(1) ∃y(P1(y) → P1(f(y)))*
(2) (P1(a) → P1(f(a)))* from (1) by ∃g
(3) P1(a) from (2) by →g1
(4) P1(f(a))* from (2) by →g2

    (5) ∀xP1(x)* by Cut
    (6) P1(a)* from (5) by ∀g using (a ≡ c_∀xP1(x))
    (7) □ by (6) and (3)

    (8) ∀xP1(x) by Cut
    (9) P1(f(a)) from (8) by ∀a
    (10) □ by (4) and (9)

In the place of the free variables we used:
a ≡ c_∀xP1(x)
```

Testovanie

- Výstupný dôkaz ρ má tvar:

```
(1) ∃y(P1(y) → P1(f(y)))*
(2) (P1(a) → P1(f(a)))* from (1) by ∃g
(3) P1(a) from (2) by →g1
(4) P1(f(a))* from (2) by →g2
(5) P1(f(a))* by ∀g using (a ≡ c_∀xP1(x))
(6) (P1(f(a)) → P1(f(a)))* from (1) by ∃g
(7) P1(f(a)) from (6) by →g1
(8) P1(f(a))* from (6) by →g2
(9) □ by (8) and (7)

In the place of the free variables we used:
a ≡ c_∀xP1(x)
```

Dosiahnute ciele a prínosy

Dosiahnute ciele

- Implementácia algoritmu eliminácie pravidla rezu v paradigme deklaratívneho programovania.

Prínosy práce

- Overenie správnosti metódy eliminácie pravidla rezu spomenutej v článku Algoritmus implementujeme na základe metódy spomenutej v článku Ján Komara. Efficient elimination of Skolem functions in LK^{h}. Archive for Mathematical Logic, 61(3):503–534, May 2022.
- Vysvetlenie všetkých prípadov štandardnej redukcie.

Záver

- Ďakujem za pozornosť