UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ELIMINÁCIA PRAVIDLA REZU V SEKVENTOVOM KALKULE LK^H

BAKALÁRSKA PRÁCA

2022
OMAR AL-SHAFE´I

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ELIMINÁCIA PRAVIDLA REZU V SEKVENTOVOM KALKULE LK^H

BAKALÁRSKA PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Ján Komara, PhD.

Bratislava, 2022
Omar Al-Shafe´i

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Omar Al-Shafe´i
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Eliminácia pravidla rezu v sekventovom kalkule LK^h
Cut elimination in the sequent calculus LK^h

Anotácia: Implementácia algoritmu eliminácie pravidla rezu pre sekventový kalkulus LK^h v paradigme deklaratívneho programovania.

Vedúci: Ing. Ján Komara, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 20.09.2021

Dátum schválenia: 30.09.2021 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Chcel by som poďakovať môjmu školitelovi Ing. Jánovi Komarovi, PhD., za jeho cenné rady a usmernenia.

Abstrakt

Náš cieľ je implementácia eliminácie pravidla rezu v paradigme deklaratívneho programovania bez použitia regularizácie. Náš algoritmus eliminácie rezu pozostáva z kváziregularizácie striedanej so štandardnou redukciou. Výstup nášho algoritmu je dôkaz, ktorý neobsahuje žiadne pravidlá rezu. Dôkazy reprezentujeme v našej implementácii pomocou tablovej metódy, napriek tomu, že teoretickú časť našej práce vysvetľujeme v sekventovom kalkule LK^h. Algoritmus eliminácie pravidla rezu implementujeme v typovej verzii programovacieho jazyka Racket, ktorý je dialektom programovacieho jazyka Lisp/Scheme. Eliminačný algoritmus implementujeme v programovacom jazyku Racket, lebo Racket je deklaratívny programovací jazyk, ktorý obsahuje konštrukcie, ako napríklad match a cond, pomocou ktorých vieme zapísať rekurziu veľmi elegantne. Naša implementácia je konzolová aplikácia, ktorá nepoužíva žiadne grafické rozhranie.

Kľúčové slová: eliminácia pravidla rezu, match, deklaratívne programovanie

Abstract

Our aim is implementation of cut elimination in paradigm of declarative programming without using regularization. Our cut elimination algorithm consists of quasiregularization alternated with standard reduction. Outcome of our algorithm is a proof, that doesn't contain any cut rule. We represent proofs in our implementation by tableau method, despite we explain theoretical part of our bachelor thesis in sequent calculus LK^h. The cut elimination algorithm is implemented in type version of programming language Racket, which is dialect of programming language Lisp/Scheme. We implement cut elimination algoritm in programming language Racket, because Racket is declarative programming language and also contains constructions such as match or cond, by which we can write recursion very elegantly. Our implementation is a console application, which doesn't use any graphical interface.

Keywords: cut elimination, match, declarative programming

Obsah

Zoznam obrázkov

Úvod

Jeden z najpodstatnejších problémov v matematickej logike, ale aj v iných disciplínach, spočíva v tom, ako zjednodušit dôkazy. Vďaka algoritmu eliminácie pravidla rezu, ktorý je matematicky korektný, vieme tento cieľ dosiahnuť. Daný algoritmus významne zjednoduší dôkaz. Cieľom tejto bakalárskej práce je implementácia algorithmu eliminácie pravidla rezu v sekventovom kalkule LK^h v paradigme deklaratívneho programovania.

Pomocou algoritmu eliminácie pravidla rezu odstraňujeme rezy s najvyššou hĺbkou, kde používame vo veľa prípadoch rotácie stromu, ktoré sa realizujú na základe štruktúry rezovej formuly. V niektorých prípadoch sa rotácia stromu ani nepoužíva. Dôkazy implementujeme ako binárne stromy, ktoré majú vo vrcholoch jednotlivé formuly. Pracujeme v sekventovom kalkule LK^h, a preto na rozdiel od logiky prvého rádu, bude stromová reprezentácia dôkazu opačná. Zároveň sa zmení aj zápis dôkazu. Dokazovacia technika, ktorú používame, je priamim dôkazom. Priamy dôkaz je štandardom v Gentzenových systémoch, ktoré používame a naviac je ľahko pochopitelný. Programovací jazyk, ktorý používame na implementáciu sa nazýva Racket. Programovací jazyk Racket je veľmi priamočiary programovací jazyk, a preto je v ňom ideálne daný algoritmus naprogramovať, a to najmä z dôvodu, že umožnuje definovať vlastnú syntax a aj objekty potrebné pre implementáciu. Rekurziu je možné zapísať vo väčšine prípadov veľmi elegantne pomocou príkazu match alebo aj pomocou príkazu cond. [5, 3]

V prvej kapitole sa v technologických východiskách zaoberáme jednotlivými technológiami, ktoré používame na implementáciu eliminácie pravidla rezu. Pri teoretických východiskách definujeme pojmy, ktoré neskôr používame v návrhu riešenia a v implementácii.

Druhá kapitola tejto práce obsahuje návrh riešenia. V rámci tohto návrhu riešenia vysvetľujeme náš spôsob eliminácie pravidla rezu v sekventovom kalkule LK^h. V tejto kapitole sa venujeme algoritmu eliminácie pravidla rezu a tiež spomíname vstupno-vstupné podmienky eliminačného algoritmu. Na konci tejto kapitoly rozoberáme vzťah sekventového kalkulu a tabla.

V tretej kapitole sa venujeme už samotnej implementácií. Na zobrazenie typov v našej implementácii používame uml triedny diagram, na ktorom sú zobrazené jednotlivé definované typy a k nim prislúchajúce funkcie. Následne pri vysvetľovaní implementačných súborov pomenuvávame najdôležitejšie funkcie v jednotlivých súboroch, ktoré sú súčasťou našej implementacie.

V piatej kapitole sa zaoberáme vizualizáciou dôkazov a testovaním algoritmu eliminácie pravidla rezu na dôkazoch.

1 Východiská

V tejto kapitole opísujeme technológie, ktoré používame na implementáciu algoritmu eliminácie pravidla rezu. Popíšeme si ich históriu a dôvody, prečo vznikli a nakoniec aj ich vlastnosti. V teoretických východiskách rozoberáme Henkinove konštanty a predovšetkým odvodzovacie pravidlá a vlastnosti dôkazov v sekventovom kalkule LK^h, ktoré často používame v ďalších kapitolách.

1.1 Technologické východiská

V tejto sekcii analyzujeme vývin a vlastnosti programovacích jazykov Lisp, Scheme a Racket.

1.1.1 Programovací jazyk Lisp

Programovací jazyk Lisp, ktorý nazývame aj LISt Processing vyvinuli na konci 50-tych rokov 20. storočia. Zakladateľom programovacieho jazyka Lisp je John McCarthy. Základným motívom, prečo McCarthy založil Lisp bola skutočnosť, že chcel odôvodniť použitie niektorých typov logických výrazov ako model výpočtu. Neskôr začali programátori používať programovací jazyk Lisp najmä na manipuláciu symbolov, napríklad na integráciu algebrických výrazov a derivovanie. Programovací jazyk Lisp používa matematickú notáciu, pričom umožnuje zapisovať operácie v prefixovom tvare, ktorý je v zátvorke, a to napríklad, keď chceme aplikovať funkciu f na n argumentov $x_1, \ldots, x_n$, tak to zapíšeme $(f\ x_1\ \ldots\ x_n)$. Najpoužívanejšia a najdôležitejšia dátová štruktúra v Lispe sú zoznamy. Najznámejšie programovacie jazyky, ktoré sú dialektami Lispu sa nazývajú Racket a Scheme. [1]

1.1.2 Programovací jazyk Scheme

Programovací jazyk Scheme vyvinuli v 70-tych rokoch 20. storočia. Za tvorcov programovacieho jazyka Scheme sa považujú Gerald J. Sussman a Guy Lee Steele. Programovací jazyk Scheme považujeme za prvý najpoužívanejší dialekt programovacieho jazyka Lisp. Spočiatku sa programovací jazyk Scheme používal predovšetkým na výzkum a výučbu. Podporoval aj pár preddefinovaných syntaktických výrazov a procedúr. Neskôr sa stal

programovací jazyk Scheme použitelným aj v iných sférach, ako napríklad pre písanie textových editorov, pre expertné systémy a podobne. Na ukladanie používa dynamickú alokáciu, ktorá, ak už nie potrebná, sa automaticky dealokuje. Na rozdiel od programovacieho jazyk Lisp využíva lexikálny rozsah. Lexikálny rozsah umožnuje používať premenné len z bloku, v ktorom sú tieto premenné definované. [2]

1.1.3 Programovací jazyk Racket

Programovací jazyk Racket je univerzálny, multi-paradigmatický programovací jazyk, založený na programovacom jazyku Scheme a súčasne je dialektom Lispu. Bol navrhnutý ako platforma pre vytváranie a implementáciu rôznych programovacích jazykov. Používame ho napríklad aj na skriptovanie, vzdelávanie v oblasti informatiky a vedecký výzkum. Programovací jazyk Racket nám dovoľuje vytvoriť si vlastný jazyk s vlastnou syntaxou a aj ho rýchlo implementovať. To znamená, že ak je nejaký jazyk nedostupný, tak programátor si ho jednoducho môže vytvoriť. Programovací jayzk Racket používa lexikálny rozsah rovnako ako programovací jazyk Scheme. [3]

1.1.4 Typová verzia programovacieho jazyka Racket

Na implementovanie algoritmu eliminácie pravidla rezu používame typovú verziu programovacieho jazyka Racket, a preto špecifikujeme, čo má byť vstup a výstup v hlavičkách funkcií.

1.2 Teoretické východiská

V tejto kapitole na začiatku vysvetľujeme jazyk prvého rádu, potom Henkinove konštanty a axiómy, a nakoniec sekventový kalkulus LK^{h}.

1.2.1 Jazyk logiky prvého rádu

Jazyk logiky prvého rádu vysvetľujeme na základe [5, 6, 7].

1.2.1.1 Symboly. Symboly jazyka logiky prvého rádu L sú:

1. Individové konštanty, ktoré označujeme malými písmenami x, y, z s prípadnými dolnými indexami.
2. Individuové parametre, ktoré označujeme malými písmenami a, b, c s prípadnými dolnými indexami.
3. Funkčné symboly, ktoré označujeme malými písmenami f, g, h s prípadnými dolnými indexami.

4. Výrokovologické spojky $\top$ (Pravda), $\bot$ (Nepravda), $\neg$, $\wedge$, $\vee$, $\rightarrow$. Napísali sme ich v poradí od najväčšej priority po najmenšiu. Spojky $\top$ a $\bot$ sú nulárne, spojka $\neg$ je unárna a ostatné spojky sú binárne.

5. Predikátové symboly, ktoré označujeme veľkými písmenami P, Q, R s prípadnými dolnými indexami.

Hovoríme, že predikátové symboly a funkčné symboly sú mimologické symboly a výrokovologické spojky sú logické symboly. Individové konštanty a parametre súhrnne nazývame premenné.

1.2.1.2 Semitermy a termy.

Semiterm definujeme rekurzívne takto:

1. Každá premenná je semiterm.

2. Každý symbol funkčnej konštanty je semiterm.

3. Aplikácia funkčnej konštanty f na n semitermov $t_1, \ldots, t_n$, ktorú píšeme $f(t_1, \ldots, t_n)$ je semiterm.

Na označovanie semitermov používame malé písmená r, s, t. Term je uzavretý semiterm, čo znamená, že neobsahuje premenné.

1.2.1.3 Semiformuly a formuly.

Semiformulu definujeme rekurzívne ($\mathrm{b} \in \{\neg, \wedge, \vee, \rightarrow\}$):

1. $\top$ a $\bot$ sú semiformuly.

2. Aplikácia predikátovej konštanty P na n semitermov $t_1, \ldots, t_n$, ktorú označujeme $P(t_1, \ldots, t_n)$ je semiformula.

3. Ak A je semiformula, tak aj $\neg A$ je semiformula.

4. Ak A a B sú semiformuly, tak aj $(A\,\mathrm{b}\,B)$ je semiformula.

5. Ak A je semiformula, tak aj $\exists x A$ je semiformula. Analogicky to platí aj pre $\forall x A$.

Na označenie semiformúl používame veľké písmená A, B, C. Formula je uzatvorená semiformula, čo znamená, že neobsahuje voľné premenné.

1.2.1.4 Hĺbka semiformúl.

Hĺbku $\mathrm{d}(A)$ semiformuly A definujeme rekurzívne ($\mathrm{b} \in \{\wedge, \vee, \rightarrow\}$, $\mathrm{Q} \in \{\exists, \forall\}$):

$$\begin{aligned} \mathrm{d}(A) &= 1 \quad A \text{ je atomická, } \top \text{ alebo } \bot, \\ \mathrm{d}(A\,\mathrm{b}\,B) &= max(\mathrm{d}(A), \mathrm{d}(B)) + 1, \\ \mathrm{d}(\neg A) &= \mathrm{d}(\mathrm{Q}xA) = \mathrm{d}(A) + 1. \end{aligned}$$

1.2.1.5 Výrazy. Výraz v jazyku logiky prvého rádu L je:

1. Semiterm.
2. Semiformula.

Výrazy označujeme písmenom E s prípadnými dolnými indexami. Na označenie výrazov rovnakého druhu používame znak $\equiv$. Napríklad $E_1 \equiv E_2$ znamená, že výrazy E_1 a E_2 sú rovnakého druhu, z čoho vyplýva, že sú oba výrazy buď semitermy, alebo semiformuly.

1.2.1.6 Substitúcia. Nech je x premenná, ktorá je substituovateľná. Potom substitúciu premennej x za term t v každom voľnom výskyte premennej x označujeme $A_x[t]$. Substitúciu definujeme induktívne ($\mathrm{b} \in \{\wedge, \vee, \rightarrow\}$, $\mathrm{Q} \in \{\exists, \forall\}$):

1. Ak A je atomická semiformula, tak jej substitúcia je $A_x[t]$.
2. Ak semiformula je tvare $\neg A$, tak jej substitúcia je $\neg(A_x[t])$.
3. Ak semiformula je v tvare $\mathrm{A}\,\mathrm{b}\,B$, tak jej substitúcia je $(\mathrm{A_x[t]})\,\mathrm{b}\,(B_x[t])$.
4. Ak semiformula je v tvare $\mathrm{Q}xA$, tak jej substitúcia je $\mathrm{Q}xA$, lebo x nemá voľný výskyt. Ak by sme substituovali premennú y za term t, tak potom substitúcia je $\mathrm{Q}x(A_y[t])$.

1.2.2 Henkinove konštanty a Henkinove axiómy

Henkinove konštanty a axiómy vysvetľujeme pomocou článku [5].

1.2.2.1 Henkinove konštanty. Nech L je jazyk logiky prvého rádu a nech C s prípadnými dolnými indexami je množina Henkinových konštánt. Nekončnú postupnosť množín Henkinových konštánt $\mathrm{C}_0, \mathrm{C}_1, \ldots, \mathrm{C}_n, \mathrm{C}_{n+1}, \ldots$ definujeme induktívne ($\mathrm{Q} \in \{\exists, \forall\}$):

$$\mathrm{C}_n = \begin{cases} \emptyset & \text{ak } n = 0 \\ \mathrm{C}_{n-1} \cup \{\mathrm{c}_{\mathrm{Q}xA} |\ \mathrm{c}_{\mathrm{Q}xA} \notin \mathrm{C}_{n-1}\} & \text{ak } n > 0 \end{cases}$$

V druhom prípade, keď je $n > 0$, Henkinova konštanta $\mathrm{c}_{\mathrm{Q}xA}$, má v idexe formulu $\mathrm{Q}xA$. Formulu $\mathrm{Q}xA$ vytvárame pomocou jazyka prvého rádu L, ktorý obsahuje Henkinove konštanty z množiny C_{n-1}.

Henkinove konštanty sú individové parametre jazyka logiky prvého rádu L.

1.2.2.2 Množina všetkých Henkinových konštánt. Zjednotením všetkých množín nekonečnej postupnosti $\mathrm{C}_0, \mathrm{C}_1, \ldots, \mathrm{C}_n, \mathrm{C}_{n+1}, \ldots$ získame množinu všetkých Henkinových konštánt. Množinu všetkých Henkinových konštánt označujeme písmenom C.

1.2.2.3 Henkinova expanzia. Henkinovu expanziu definujeme ako prvorádový jazyk L, ktorý obsahuje Henkinove konštanty z množiny C. Henkinovu expanziu označujeme $L(\mathrm{C})$.

1.2.2.4 Čisté semiformuly a termy. Ak sú semiformula a term čisté, tak neobsahujú Henkinové konštanty z Henkinovej expanzie $L(\mathrm{C})$. Jazyk L, ktorý neobsahuje Henkinove konštanty, obsahuje len čisté semiformuly a termy.

1.2.2.5 Henkinove a kvantifikačné axiómy. Henkinov axióm je:

1. $\exists x A \to A_x[\mathrm{c}_{\exists x A}]$
2. $A_x[\mathrm{c}_{\forall x A}] \to \forall x A$

Nech t je term Henkinovej expanzie $L(\mathrm{C})$. Kvantifikačný axióm je:

1. $A_x[t] \to \exists x A$
2. $\forall x A \to A_x[t]$

1.2.3 Sekventový kalkulus LK^{h}

V tejto podsekcii definujeme odvodzovacie pravidlá, ktoré sme prevzali z článku [5], pre kalkul LK^{h} a aj, ako reprezentujeme dôkazy v tomto kalkule. Neskôr spomíname aj základne miery na dôkazoch, ktoré využívame v následujúcich kapitolách napriklád pre odhady. Sekventový kalkul LK^{h} vysvetľujeme podľa článku [5].

1.2.3.1 Sekventy. Nech Γ a Δ sú konečné multimnožiny pozostávajúce zo semiformúl. Sekvent označujeme $\Gamma \Rightarrow \Delta$, kde symbol $\Rightarrow$ oddeľuje predpoklady Γ od cieľov Δ. Sekvent v tvare $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$ môžeme zapísať pomocou výrokovologických spojok na $A_1 \wedge \ldots \wedge A_n \to B_1 \vee \ldots \vee B_m$.

Sekventy označujeme veľkými písmenom S s prípadnými dolnými indexami.

1.2.3.2 Odvodzovacie pravidlá

Axiomatické pravidlá

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

Vyrokovologické pravidlá

$$\mathrm{L}\neg\ \frac{\Gamma \Rightarrow \Delta,\ A}{\neg A,\ \Gamma \Rightarrow \Delta}, \qquad \mathrm{R}\neg\ \frac{A,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta,\ \neg A},$$

$$\mathrm{L}\wedge\,\frac{A,\,B,\,\Gamma\Rightarrow\Delta}{A\wedge B,\,\Gamma\Rightarrow\Delta}\,,\qquad \mathrm{R}\wedge\,\frac{A,\,\Gamma\Rightarrow\Delta\quad B,\,\Gamma\Rightarrow\Delta}{\Gamma\Rightarrow\Delta,\,A\wedge B}\,,$$

$$\mathrm{L}\vee\,\frac{A,\,\Gamma\Rightarrow\Delta\quad B,\,\Gamma\Rightarrow\Delta}{A\vee B,\,\Gamma\Rightarrow\Delta}\,,\qquad \mathrm{R}\vee\,\frac{\Gamma\Rightarrow\Delta,\,A,\,B}{\Gamma\Rightarrow\Delta,\,A\vee B}\,,$$

$$\mathrm{L}\to\,\frac{\Gamma\Rightarrow\Delta,\,A\quad B,\,\Gamma\Rightarrow\Delta}{A\to B,\,\Gamma\Rightarrow\Delta}\,,\qquad \mathrm{R}\to\,\frac{A,\,\Gamma\Rightarrow\Delta,\,B}{\Gamma\Rightarrow\Delta,\,A\to B}\,.$$

Kvantifikačné pravidlá

$$\mathrm{L}\exists\,\frac{A_x[\mathrm{c}_{\exists xA}],\,\Gamma\Rightarrow\Delta}{\exists xA,\,\Gamma\Rightarrow\Delta}\,,\qquad \mathrm{R}\exists\,\frac{\Gamma\Rightarrow\Delta,\,\exists xA,\,A_x[t]}{\Gamma\Rightarrow\Delta,\,\exists xA}\,,$$

$$\mathrm{L}\forall\,\frac{A_x[t],\,\forall xA,\,\Gamma\Rightarrow\Delta}{\forall xA,\,\Gamma\Rightarrow\Delta}\,,\qquad \mathrm{R}\forall\,\frac{\Gamma\Rightarrow\Delta,\,A_x[\mathrm{c}_{\forall xA}]}{\Gamma\Rightarrow\Delta,\,\forall xA}\,.$$

Štrukturálne pravidlá

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

Kvantifikačné pravidlá L$\exists$ a R$\forall$ nazývame silné kvantifikačné pravidlá. Kvantifikačné pravidlá L$\forall$ a R$\exists$ nazývame slabé kvantifikačné pravidlá. Pojmy, ako hlavná formula, pomocná formula,aktívne formuly a vedlajšie formuly sú vysvetlené v článku [5].

1.2.3.3 Dôkazy. Dôkaz reprezentujeme binárnym stromom, ktorého vrcholy sú označené semiformuly. Vrcholy stromu nazývame sekventy a vytvárame ich pomocou odvodzovacích pravidiel.

Koreň stromu nazývame konečný sekvent. Listy stromu nazývame začiatočné sekventy, ktoré vznikajú len z axiomatických odvodzovacích pravidiel.

Dôkaz označujeme gréckym písmenami π a ρ. Ak π je dôkaz sekventu $\Gamma\Rightarrow\Delta$, tak píšeme $\pi\vdash\Gamma\Rightarrow\Delta$. V prípade, ak nehovoríme o konkrétnom dôkaze ale len o tom, že dôkaz existuje, píšeme $\vdash\Gamma\Rightarrow\Delta$.

1.2.3.4 Hĺbky dôkazov. Hĺbka $|\pi|$ dôkazu π je počet sekventov v jeho najdlhšej vetve. To znamená, že ak dôkaz π obsahuje len jeden sekvent, jeho hĺbka je $|\pi|=1$. Všeobecne platí, že $|\pi|>0$.

1.2.3.5 Rezové hodnosti dôkazov. Rezová hodnosť $\mathrm{r}(\pi)$ dôkazu π je suprémom hĺbok všetkých rezových formúl v dôkaze π. Ak dôkaz π neobsahuje žiadne pravidlo rezu, tak $\mathrm{r}(\pi)=0$, lebo $\sup\emptyset=0$.

2 Návrh riešenia

V tejto kapitole je obsiahnutý celý proces eliminácie pravidla rezu v sekventovom kalkule LK^h. Rozoberáme aj jednotlivé vety a lemy, ktoré potrebujeme na matematické zdôvodnenie algoritmu elimináciu pravidla rezu. Následne vysvetľujeme, ako funguje algoritmus eliminácie pravidla rezu. Nakoniec opísujeme vzťah sekventového kalkulu a tabla. Vety, lemy a definície vysvetľujeme na základe článku [5]. Dôkazy väčšiny tvrdení v tejto sekcii sa nachádzajú v článku [5].

2.1 Základné transformácie dôkazov

Predpokladáme, že všetky dôkazy od tejto sekcie, ak nie sú bližšie špecifikované, sú v kalkule LK^h. Výrazmi od tejto sekcie sa rozumejú nielen semitermy a semiformuly, ale aj sekventy, odvodzovacie pravidlá a dôkazy. Jednotlivé lemy, vety a definície vysvetľujeme na základe [5, 4, 8].

2.1.1 Štrukturálna podobnosť dôkazov

Dva dôkazy sú štrukturálne podobné, ak medzi nimi existuje izomorfná stromová reprezentácia, v ktorej každá dvojica odvodzovacích pravidiel sa odlišuje len vo vedľajších formulách, aktívne a pomocné formuly zostávajú rovnaké. Štrukturálne podobné dôkazy majú rovnakú hĺbku a rezovú hodnosť.

2.1.2 Štrukturálna lema

Nech $\pi \vdash \Gamma \Rightarrow \Delta$ je dôkaz a nech Λ a Ω sú multimnožiny, pre ktoré platí $\Gamma \subseteq \Lambda$ a $\Delta \subseteq \Omega$. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi' \vdash \Lambda \Rightarrow \Omega$.

Dôkaz π' píšeme ako $\pi\,[\Lambda \Rightarrow \Omega]$.

2.1.3 Zoslabenie

Ako dôsledok štrukturálnej lemy dostaneme:

1. Nech $\pi \vdash \Gamma \Rightarrow \Delta$ je dôkaz a nech Λ a Ω sú ľubovoľné multimnožiny. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi\,[\Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega] \vdash \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega$.

2.1.4 Skrátenie

Ako dôsledok štrukturálnej lemy dostaneme:

1. Nech $\pi \vdash A,\ A,\ \Gamma \Rightarrow \Delta$ je dôkaz. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi\,[A,\ \Gamma \Rightarrow \Delta] \vdash A,\ \Gamma \Rightarrow \Delta$.

2. Nech $\pi \vdash \Gamma \Rightarrow \Delta,\ A,\ A$ je dôkaz. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi\,[\Gamma \Rightarrow \Delta,\ A] \vdash \Gamma \Rightarrow \Delta,\ A$.

2.1.5 Zosilnenie

Vetu o zosilnení formulujeme takto:

1. Nech $\pi \vdash A,\ \Gamma \Rightarrow \Delta$ je dôkaz a nech A nie je hlavná formula žiadneho odvodzovacieho pravidla v dôkaze π. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi' \vdash \Gamma \Rightarrow \Delta$.

2. Nech $\pi \vdash \Gamma \Rightarrow \Delta,\ A$ je dôkaz a nech A nie je hlavná formula žiadneho odvodzovacieho pravidla v dôkaze π. Potom k dôkazu π existuje štrukturálne podobný dôkaz $\pi' \vdash \Gamma \Rightarrow \Delta$.

Dôkaz tvrdenia je priamočiary s využitím indukcie na dĺžku dôkazu.

2.1.6 Definícia

Dôkaz je voľný pre Henkinovu konštantu c_{QxA}, ak neobsahuje žiadne silné kvantifikačné pravidlo, v ktorom je QxA hlavná formula $(Q \in \{\exists, \forall\})$.

2.1.7 Inverzia

1. Nech $\pi \vdash \neg A,\ \Gamma \Rightarrow \Delta$. Potom existuje dôkaz $\pi' \vdash \Gamma \Rightarrow \Delta,\ A$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

2. Nech $\pi \vdash \Gamma \Rightarrow \Delta, \neg A$. Potom existuje dôkaz $\pi' \vdash A,\ \Gamma \Rightarrow \Delta$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

3. Nech $\pi \vdash A \wedge B,\ \Gamma \Rightarrow \Delta$. Potom existuje dôkaz $\pi' \vdash A,\ B,\ \Gamma \Rightarrow \Delta$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

4. Nech $\pi \vdash \Gamma \Rightarrow \Delta,\ A \wedge B$. Potom existujú dôkazy $\pi'_1 \vdash A,\ \Gamma \Rightarrow \Delta$ a $\pi'_2 \vdash B,\ \Gamma \Rightarrow \Delta$ s rezovými hodnosťami $\mathrm{r}(\pi'_1) \leq \mathrm{r}(\pi)$, $\mathrm{r}(\pi'_2) \leq \mathrm{r}(\pi)$ a hĺbkami $|\pi'_1| \leq |\pi|$, $|\pi'_2| \leq |\pi|$.

5. Nech $\pi \vdash A \vee B,\ \Gamma \Rightarrow \Delta$. Potom existujú dôkazy $\pi'_1 \vdash A,\ \Gamma \Rightarrow \Delta$ a $\pi'_2 \vdash B,\ \Gamma \Rightarrow \Delta$ s rezovými hodnosťami $\mathrm{r}(\pi'_1) \leq \mathrm{r}(\pi)$, $\mathrm{r}(\pi'_2) \leq \mathrm{r}(\pi)$ a hĺbkami $|\pi'_1| \leq |\pi|$, $|\pi'_2| \leq |\pi|$.

6. Nech $\pi \vdash \Gamma \Rightarrow \Delta, A \vee B$. Potom existuje dôkaz $\pi' \vdash \Gamma \Rightarrow \Delta, A, B$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

7. Nech $\pi \vdash A \rightarrow B, \Gamma \Rightarrow \Delta$. Potom existujú dôkazy $\pi'_1 \vdash \Gamma \Rightarrow \Delta, A$ a $\pi'_2 \vdash \Gamma \Rightarrow \Delta, B$ s rezovými hodnosťami $\mathrm{r}(\pi'_1) \leq \mathrm{r}(\pi)$, $\mathrm{r}(\pi'_2) \leq \mathrm{r}(\pi)$ a hĺbkami $|\pi'_1| \leq |\pi|$, $|\pi'_2| \leq |\pi|$.

8. Nech $\pi \vdash \Gamma \Rightarrow \Delta, A \rightarrow B$. Potom existuje dôkaz $\pi' \vdash A, \Gamma \Rightarrow \Delta, B$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

9. Nech $\pi \vdash \exists x A, \Gamma \Rightarrow \Delta$. Potom existuje dôkaz $\pi' \vdash A_x[\mathrm{c}_{\exists xA}], \Gamma \Rightarrow \Delta$ voľný pre Henkinovu konštantu $\mathrm{c}_{\exists xA}$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

10. Nech $\pi \vdash \Gamma \Rightarrow \Delta, \forall x A$. Potom existuje dôkaz $\pi' \vdash \Gamma \Rightarrow \Delta, A_x[\mathrm{c}_{\forall xA}]$ voľný pre Henkinovu konštantu $\mathrm{c}_{\forall xA}$ s rezovou hodnosťou $\mathrm{r}(\pi') \leq \mathrm{r}(\pi)$ a hĺbkou $|\pi'| \leq |\pi|$.

2.1.8 Hlboké nahradenie termov

Hlboký výskyt termu r vo výraze E definujeme induktívne takto:

1. Ak term r má bežný výskyt vo výraze E, tak term r má hlboký výskyt vo výraze E.

2. Ak sa vo výraze E vyskytuje Henkinova konštanta v tvare $\mathrm{c}_{\mathrm{Q}xA}$, v ktorej indexe $\mathrm{Q}xA$ term r má hlboký výskyt, tak term r má hlboký výskyt aj vo výraze E.

Hlbokým nahradením termu r za term s vo výraze E rozumieme nahradenie každého hlbokého výskytu termu r vo výraze E za term s. Takéto hlboké nahradenie označujeme $E\{r/s\}$.

2.1.9 Lema o hlbokom nahradení

Ak $\pi \vdash \Gamma \Rightarrow \Delta$ je dôkaz, ktorý voľný pre Henkinovu konštantu $\mathrm{c}_{\mathrm{Q}xA}$, tak $\pi\{\mathrm{c}_{\mathrm{Q}xA}/s\} \vdash \Gamma\{\mathrm{c}_{\mathrm{Q}xA}/s\} \Rightarrow \Delta\{\mathrm{c}_{\mathrm{Q}xA}/s\}$, pričom platí, že $\mathrm{r}(\pi\{\mathrm{c}_{\mathrm{Q}xA}/s\}) = \mathrm{r}(\pi)$ a $|\pi\{\mathrm{c}_{\mathrm{Q}xA}/s\}| = |\pi|$.

2.2 Kváziregularizácia

V tejto sekcii opisujeme vlastnosť kváziregularity a vysvetľujeme celý algoritmus kváziregularizácie.

2.2.1 Kváziregularita

Rez s rezovou formulou $\mathrm{Q}xA$ v tvare $(\mathrm{Q} \in \{\exists, \forall\})$

$$\mathrm{Cut}\,\frac{\Gamma \Rightarrow \Delta, \mathrm{Q}xA \qquad \mathrm{Q}xA, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

nazývame kritický, ak sa v závere $\Gamma \Rightarrow \Delta$ pravidla rezu hlboko nachádza Henkinova konštanta c_{QxA}.

Dôkaz π je r-kváziregulárny, ak neobsahuje žiaden kritický rez stupňa r.

2.2.2 Inverzia rezu

Nech $\pi \vdash \Gamma \Rightarrow \Delta$ je dôkaz. Potom existujú dôkazy $\pi_1' \vdash \Gamma \Rightarrow \Delta, A$ a $\pi_2' \vdash A, \Gamma \Rightarrow \Delta$, ktoré neobsahujú rezy s rezovou formulou A a zároveň obsahujú všetky zvyšné rezy v dôkaze π. Dôkazy π_1' a π_2' majú hĺbky $|\pi_1'| \leq |\pi|$ a $|\pi_2'| \leq |\pi|$.

2.2.3 Algoritmus kváziregularizácie

Nech π je dôkaz čistého sekventu. Potom existuje $\mathrm{r}(\pi)$-kváziregulárny dôkaz π' rovnakého sekventu s rezovou hodnosťou $\mathrm{r}(\pi) = \mathrm{r}(\pi')$ a hĺbkou $|\pi'| < 2^{|\pi|}$.

Algoritmus kváziregularizácie sa skladá z dvoch krokov:

1. Rezové formuly kvantifikačných rezov s rezovou hodnosťou $\mathrm{r}(\pi)$ topologicky utriedime do poľa. Utriedenie v poli realizujeme pomocou vybudovanej sort funkcie tak, že pre každé dve formuly A a B skontrolujeme, či sa Henkinova konštanta priradená rezovej formule A hlboko nachádza v B. Ak áno, tak sa zotriedia v poradí B, A. Ak nie, tak sa zotriedia v poradí A, B.

2. Potom aplikujeme inverziu rezu n-krát, kde pri každej inverzii vždy vytvoríme rez z nasledujúcej rezovej formuli v poli a rez pridáme na koniec dôkazu.

2.3 Redukcia rezu

V tejto sekcii opisujeme redukciu rezu a aj jej jednotlivé prípady.

2.3.1 Algoritmus redukcie rezu

Nech $\pi \vdash \Gamma \Rightarrow \Delta$ je r-kváziregulárny dôkaz s finálnym rezom s rezovou formulou A a rezovou hodnosťou $r = \mathrm{d}(A)$, ktorá je väčšia než rezová hodnosť iných rezov v dôkaze π. Potom existuje dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s rezovou hodnosťou $\mathrm{r}(\rho) < \mathrm{r}$ a hĺbkou $|\rho| < 2|\pi|$.

2.3.2 Rezová formula je atomická

Dôkaz π má tvar

$$\left.\begin{array}{c} \vdots\,\pi_1 \qquad \vdots\,\pi_2 \\ \mathrm{Cut}\,\dfrac{\Gamma \Rightarrow \Delta, A \qquad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\cdot \end{array}\right\}\pi$$

V dôkaze π_1 hľadáme poddôkazy, ktoré končia axiomatickým odvodzovacím pravidlom Ax s hlavnou formulou A. Bez ujmy na obecnosti môžeme predpokladať, že každý taký poddôkaz je v tvare

$$\mathrm{Ax}\,\frac{}{A,\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ A}\,.$$

Poddôkaz transformujeme do tvaru

$$\dfrac{\begin{array}{c}\vdots\\ \pi_2\,[A,\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega]\end{array}}{A,\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega}\,.$$

Pomocou transformácie každého takého poddôkazu dostaneme dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s požadovanými vlastnosťami.

2.3.3 Rezová formula je výrokovologická konštanta $\top$

Nech $A \equiv \top$. Dôkaz π má tvar

$$\left.\begin{array}{c}\mathrm{Cut}\,\dfrac{\begin{array}{cc}\vdots & \vdots\\ \pi_1 & \pi_2\end{array}}{\Gamma \Rightarrow \Delta,\ \top \qquad \top,\ \Gamma \Rightarrow \Delta}\,.\\ \Gamma \Rightarrow \Delta\end{array}\right\}\pi$$

Keďže $\top$ nie je hlavná formula žiadneho odvodzovacieho pravidla v poddôkaze π_2, tak použijeme zosilnenie na poddôkaz π_2. Tým dostaneme dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s požadovanými vlastnosťami

2.3.4 Rezová formula je výrokovologická konštanta $\bot$

Nech $A \equiv \bot$. Dôkaz π má tvar

$$\left.\begin{array}{c}\mathrm{Cut}\,\dfrac{\begin{array}{cc}\vdots & \vdots\\ \pi_1 & \pi_2\end{array}}{\Gamma \Rightarrow \Delta,\ \bot \qquad \bot,\ \Gamma \Rightarrow \Delta}\,.\\ \Gamma \Rightarrow \Delta\end{array}\right\}\pi$$

Keďže $\bot$ nie je hlavná formula žiadneho odvodzovacieho pravidla v poddôkaze π_1, tak použijeme zosilnenie na poddôkaz π_1. Tým dostaneme dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s požadovanými vlastnosťami.

2.3.5 Rezová formula je negácia

Nech $A \equiv \neg B$. Dôkaz π má tvar

$$\left.\begin{array}{c} \vdots\,\pi_1 \qquad \vdots\,\pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ \neg B \qquad \neg B,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \vdots\,\pi'_1 \qquad\qquad \vdots\,\pi'_2 \\ \begin{array}{l}\text{R}\neg\\ \text{Cut}\end{array}\,\dfrac{\dfrac{B,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta,\ \neg B} \qquad \text{L}\neg\,\dfrac{\Gamma \Rightarrow \Delta,\ B}{\neg B,\ \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi'$$

Pomocou dvoch aplikácii pravidla rezu dostaneme dôkaz s požadovanými vlastnosťami

$$\left.\begin{array}{c} \vdots\,\pi'_2 \qquad \vdots\,\pi'_1 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \qquad B,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\rho$$

2.3.6 Rezová formula je konjukcia

Nech $A \equiv B \wedge C$. Dôkaz π má tvar

$$\left.\begin{array}{c} \vdots\,\pi_1 \qquad \vdots\,\pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \wedge C \qquad B \wedge C,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \vdots\,\pi'_{11} \qquad \vdots\,\pi'_{12} \qquad\qquad \vdots\,\pi'_2 \\ \begin{array}{l}\text{R}\wedge\\ \text{Cut}\end{array}\,\dfrac{\dfrac{\Gamma \Rightarrow \Delta,\ B \qquad \Gamma \Rightarrow \Delta,\ C}{\Gamma \Rightarrow \Delta,\ B \wedge C} \qquad \text{L}\wedge\,\dfrac{B,\ C,\ \Gamma \Rightarrow \Delta}{B \wedge C,\ \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi'$$

Pomocou dvoch aplikácii pravidla rezu a jedného zoslabenia dostaneme dôkaz s požadovanými vlastnosťami

$$\left.\begin{array}{c} \vdots\,\pi'_{11} \qquad\qquad \vdots\,\pi'_{12}\,[B,\ \Gamma \Rightarrow \Delta,\ C] \qquad\qquad \vdots\,\pi'_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \qquad\qquad \text{Cut}\,\dfrac{B,\ \Gamma \Rightarrow \Delta,\ C \qquad\qquad C,\ B,\ \Gamma \Rightarrow \Delta}{B,\ \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\rho$$

2.3.7 Rezová formula je disjunkcia

Nech $A \equiv B \vee C$. Dôkaz π má tvar

$$\left.\begin{array}{c} \vdots\,\pi_1 \qquad \vdots\,\pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \vee C \qquad B \vee C,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \vdots\,\pi'_1 \qquad\qquad \vdots\,\pi'_{21} \qquad \vdots\,\pi'_{22} \\ \text{R}\vee\,\dfrac{\Gamma \Rightarrow \Delta,\ B,\ C}{\Gamma \Rightarrow \Delta,\ B \vee C} \quad \text{L}\vee\,\dfrac{B,\ \Gamma \Rightarrow \Delta \qquad C,\ \Gamma \Rightarrow \Delta}{B \vee C,\ \Gamma \Rightarrow \Delta} \\ \text{Cut}\,\dfrac{\qquad\qquad\qquad\qquad}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi'$$

Pomocou dvoch aplikácii pravidla rezu a jedného zoslabenia dostaneme dôkaz s požadovanými vlastnosťami

$$\left.\begin{array}{c} \vdots\,\pi'_1 \qquad \vdots\,\pi'_{22}\,[C,\ \Gamma \Rightarrow \Delta,\ B] \qquad\qquad \vdots\,\pi'_{21} \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B,\ C \qquad C,\ \Gamma \Rightarrow \Delta,\ B}{\Gamma \Rightarrow \Delta,\ B} \qquad\qquad B,\ \Gamma \Rightarrow \Delta \\ \text{Cut}\,\dfrac{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\rho$$

2.3.8 Rezová formula je implikácia

Nech $A \equiv B \to C$. Dôkaz π má tvar

$$\left.\begin{array}{c} \vdots\,\pi_1 \qquad \vdots\,\pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \to C \qquad B \to C,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \vdots\,\pi'_1 \qquad\qquad \vdots\,\pi'_{21} \qquad \vdots\,\pi'_{22} \\ \text{R}\to\,\dfrac{B,\ \Gamma \Rightarrow \Delta,\ C}{\Gamma \Rightarrow \Delta,\ B \to C} \quad \text{L}\to\,\dfrac{\Gamma \Rightarrow \Delta,\ B \qquad C,\ \Gamma \Rightarrow \Delta}{B \to C,\ \Gamma \Rightarrow \Delta} \\ \text{Cut}\,\dfrac{\qquad\qquad\qquad\qquad}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\pi'$$

Pomocou dvoch aplikácii pravidla rezu a jedného zoslabenia dostaneme dôkaz s požadovanými vlastnosťami

$$\left.\begin{array}{c} \vdots\,\pi'_{21} \qquad\qquad \vdots\,\pi'_1 \qquad \vdots\,\pi'_{22}\,[C,\ B,\ \Gamma \Rightarrow \Delta] \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ B \qquad \text{Cut}\,\dfrac{B,\ \Gamma \Rightarrow \Delta,\ C \qquad C,\ B,\ \Gamma \Rightarrow \Delta}{B,\ \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}\,. \end{array}\right\}\rho$$

.

2.3.9 Rezová formula je existenčná

Nech $A \equiv \exists x B$. Dôkaz π má tvar

$$\left.\begin{array}{c} \pi_1 \qquad\qquad \pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ \exists x B \qquad \exists x B,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \end{array}\right\} \pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \pi_1 \qquad\qquad\qquad\qquad \pi_2' \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ \exists x B \qquad \text{L}\exists\,\dfrac{B_x\left[\mathrm{c}_{\exists x B}\right],\ \Gamma \Rightarrow \Delta}{\exists x B,\ \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}, \end{array}\right\} \pi'$$

kde poddôkaz π_2' je voľný pre Henkinovu konštantu $\mathrm{c}_{\exists x B}$.

V dôkaze π_1 hľadáme všetky poddôkazy postupne zhora nadol, ktoré končia kvantifikačným odvodzovacím pravidlom R$\exists$ s hlavnou formulou $\exists x B$. Bez ujmy na obecnosti môžeme predpokladať, že každý taký poddôkaz je v tvare

$$\text{R}\exists\,\frac{\begin{array}{c}\pi_{11}' \\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ \exists x B,\ B_x[t]\end{array}}{\Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ \exists x B}.$$

Keďže $\exists x B$ sa nevyskytuje ako hlavná formula žiadneho odvodzovacieho pravidla v poddôkaze π_{11}, aplikujeme zosilnenie na poddôkaz π_{11} a dostaneme poddôkaz $\pi_{11}' \vdash \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ B_x[t]$. Poddôkaz π_{11} transformujeme do tvaru

$$\text{Cut}\,\frac{\begin{array}{c}\pi_{11}' \\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ B_x[t]\end{array} \qquad \begin{array}{c}\pi_2'\left\{\mathrm{c}_{\exists x B}/t\right\}\left[B_x[t],\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega\right] \\ B_x[t],\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega\end{array}}{\Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega}.$$

Pomocou transformácie každého takého poddôkazu dostaneme dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s požadovanými vlastnosťami.

2.3.10 Rezová formula je všeobecná

Nech $A \equiv \forall x B$. Dôkaz π má tvar

$$\left.\begin{array}{c} \pi_1 \qquad\qquad \pi_2 \\ \text{Cut}\,\dfrac{\Gamma \Rightarrow \Delta,\ \forall x B \qquad \forall x B,\ \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}. \end{array}\right\} \pi$$

Pomocou inverzie transformujeme dôkaz do tvaru

$$\left.\begin{array}{c} \vdots\,\pi_1' \\ \dfrac{\Gamma \Rightarrow \Delta,\ B_x[\mathrm{c}_{\forall x B}]}{\Gamma \Rightarrow \Delta,\ \forall x B}\,\mathrm{R}\forall \qquad \vdots\,\pi_2 \\ \dfrac{\qquad\qquad\qquad\qquad}{\Gamma \Rightarrow \Delta}\,\mathrm{Cut} \qquad \forall x B,\ \Gamma \Rightarrow \Delta \end{array}\right\}\pi'$$

kde poddôkaz π_1' je voľný pre Henkinovu konštantu $\mathrm{c}_{\forall x B}$.

V dôkaze π_2 hľadáme všetky poddôkazy zhora nadol, ktoré končia kvantifikačným odvodzovacím pravidlom L$\forall$ s hlavnou formulou $\forall x B$. Bez ujmy na obecnosti môžeme predpokladať, že každý taký poddôkaz je v tvare

$$\mathrm{L}\forall\,\frac{\begin{array}{c}\vdots\,\pi_{21}\\ B_x[t],\ \forall x B,\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega\end{array}}{\forall x B,\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega}.$$

Keďže $\forall x B$ sa nevyskytuje ako hlavná formula žiadného odvodzovacieho v poddôkaze π_{21}, aplikujeme zosilnenie na poddôkaz π_{21} a dostaneme poddôkaz $\pi_{21}' \vdash B_x[t],\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega$. Poddôkaz π_{21} transformujeme do tvaru

$$\mathrm{Cut}\,\frac{\begin{array}{c}\vdots\,\pi_1'\{\mathrm{c}_{\forall x B}/t\}[\Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ B_x[t]]\\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega,\ B_x[t]\end{array} \qquad \begin{array}{c}\vdots\,\pi_{21}'\\ B_x[t],\ \Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega\end{array}}{\Lambda,\ \Gamma \Rightarrow \Delta,\ \Omega}.$$

Pomocou transformácie každého takého poddôkazu dostaneme dôkaz $\rho \vdash \Gamma \Rightarrow \Delta$ s požadovanými vlastnosťami.

2.4 Eliminácia pravidla rezu

Náš algoritmus nepoužíva Gentzenovský spôsob eliminácie pravidla rezu, lebo dôkazy v LK^{h} nie sú regulárne. Dôkazy, ktoré nie sú regulárne nespĺňajú štandardnú eigenvariable condition.

Vstup eliminačného algoritmu je dôkaz čistého sekventu π. Algoritmus postupne eliminuje rezy s rezovou hodnosťou $\mathrm{r}(\pi)$ a nahradí ich rezami s nižšiou rezovou hodnosťou aspoň o 1, potom eliminuje rezy s rezovou hodnosťou $\mathrm{r}(\pi)-1,\ldots$ a nakoniec eliminuje rezy s rezovou hodnosťou 1. Každé takéto zníženie rezovej hodnosti dôkazu π aspoň o 1 sa skladá z dvoch krokov:

1. Kváziregularizácia dôkazu π: opraví eigenvariable condition pre kvantifikačné rezy s rezovou hodnosťou dôkazu. Tým dostaneme dôkaz σ s hĺbkou

$$|\sigma| < 2^{|\pi|}.$$

2. Štandardná redukcia dôkazu σ: eliminuje rezy s rezovou hodnosťou $\mathrm{r}(\sigma)$ a nahradí ich rezami s nižšiou rezovou hodnosťou aspoň o 1. Tým dostaneme dôkaz ρ s hĺbkou

$$|\rho| < 2^{|\sigma|} < 2^{2^{|\pi|}}.$$

Výstup eliminačného algoritmu je bezrezový dôkaz ρ rovnakého sekventu s hĺbkou

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

Štandardný algoritmus eliminácie pravidla rezu v LK má odhad

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

Štandardný eliminačný algoritmus využíva regularizáciu. Nevýhodou regularizacie je to, že vzniknú nové rezy, ktorých rezová hodnosť sa nedá vyjadriť pomocou rezovej hodnosti originálneho dôkazu.

2.5 Vzťah medzi sekventovým kalkulom a tablom

V tejto kapitole vysvetľujeme metódu tabiel a korešpondenciu medzi tablami a sekventovým kalkulom.

Ľubovoľnú označenú formulu v tvare $A*$ nazývame cieľ a ľubovoľnú označenú formulu v tvare A nazývame predpoklad.

Smullyan spomína v [7], že tablom pre sekvent $A_1,\ldots,A_n \Rightarrow B_1,\ldots,B_n$ je v podstate tablo pre množinu $\{A_1,\ldots,A_n,\ B_1*,\ldots,B_m*\}$, a preto uzavreté tablo pre množinu $\{A_1,\ldots,A_n,\ B_1*,\ldots,B_m*\}$ nazývame tablovým dôkazom sekventu $A_1,\ldots,A_n \Rightarrow B_1,\ldots,B_n$.

2.5.1 Metóda tabiel

Odvodzovacie pravidlá tablovej metódy, na rozdiel od odvodzovacích pravidiel nášho kalkulu, neobsahujú bočné formuly. Odvodzovacie pravidlá pre tablovej metódy sú:

Axiomatické pravidlá

$$\Box\ \frac{\begin{array}{c}A*\\ A\end{array}}{\Box} \qquad \Box\bot\ \frac{\bot}{\Box} \qquad \Box\top\ \frac{\top*}{\Box}$$

Axiomatické pravidlo $\Box$ sa dá aplikovať, ak sa na jednej vetve dôkazu nachádzajú formuly A a $A*$ bez ohľadu na ich poradie.

Unárne výrokovologické pravidlá

$$\neg a\ \frac{\neg A}{A*} \qquad \neg g\ \frac{\neg A*}{A}$$

$$\wedge a1\ \frac{A \wedge B}{A} \qquad \wedge a2\ \frac{A \wedge B}{B}$$

$$\vee g1\ \frac{A \vee B*}{A*} \qquad \vee g2\ \frac{A \vee B*}{B*}$$

$$\rightarrow g1\ \frac{A \rightarrow B*}{A} \qquad \rightarrow g2\ \frac{A \rightarrow B*}{B*}$$

Binárne výrokovologické pravidlá

$$\wedge g\ \frac{A \wedge B*}{A* \quad B*} \qquad \vee a\ \frac{A \vee B}{A \quad B} \qquad \rightarrow a\ \frac{A \rightarrow B}{A* \quad B}$$

Kvantifikačné pravidlá

$$\exists a\ \frac{\exists x A}{A_x[\mathrm{c}_{\exists x A}]} \qquad \exists g\ \frac{\exists x A*}{A_x[t]*}$$

$$\forall a\ \frac{\forall x A}{A_x[t]} \qquad \forall g\ \frac{\forall x A*}{A_x[\mathrm{c}_{\forall x A}]*}$$

Štrukturálne pravidlá

$$Cut\ \frac{}{A* \quad A}$$

2.5.2 Korešpondencia medzi sekventovým kalkulom a tablom

V našom variante Gentzenových systémov používame sekventový kalkul. Gentzenove systémy používaju aj tablovú metódu.

Ak chceme dôkaz v tablovej metóde previesť do sekventového kalkulu, tak bod tabla nahradíme množinou obsahujúcou tento bod a všetky body nad týmto bodom, na ktorých sa odvodzovacie pravidlo ešte nepoužilo. V prípade unárnych výrokovologických pravidiel dva body tabla nahradíme jedným sekventom, keď použijeme postupne pravidlo, ktoré sa odlišuje len v indexe. Následne treba dôkaz prevrátiť, lebo sekventový dôkaz sa číta zdola nahor.

Najprv uvádzame príklad dôkazu sekventu $\Rightarrow (\neg P \vee \neg Q) \vee Q \wedge P$ v stromovej podobe sekventového kalkulu LK^{h}

$$\dfrac{\mathrm{R}\vee\dfrac{\mathrm{R}\wedge\dfrac{\mathrm{R}\vee\dfrac{\mathrm{R}\neg\dfrac{\mathrm{Ax}\dfrac{}{Q \Rightarrow Q, \neg P}}{\Rightarrow Q,\ \neg P,\ \neg Q}}{\Rightarrow (\neg P \vee \neg Q),\ Q} \qquad \mathrm{R}\vee\dfrac{\mathrm{R}\neg\dfrac{\mathrm{Ax}\dfrac{}{P \Rightarrow \neg Q,\ P}}{\Rightarrow P,\ \neg P,\ \neg Q}}{\Rightarrow (\neg P \vee \neg Q),\ P}}{\Rightarrow (\neg P \vee \neg Q),\ Q \wedge P}}{\Rightarrow (\neg P \vee \neg Q) \vee Q \wedge P}$$

a následne rovnaký dôkaz v tablovej metóde.

Tablový dôkaz $(\neg P \vee \neg Q) \vee Q \wedge P*$.

$(\neg P \vee \neg Q) \vee Q \wedge P*$ (1)	Treba dokázať
$(\neg P \vee \neg Q)*$ (2)	Z (1) pomocou $\vee g1$
$Q \wedge P*$ (3)	Z (1) pomocou $\vee g2$
$Q*$ (4) $\quad$ $P*$ (7)	Z (3) pomocou $\wedge g$
$\neg Q*$ (5) $\quad$ $\neg P*$ (8)	(5) z (2) pomocou $\vee g2$; (8) z (2) pomocou $\vee g1$
Q (6) $\quad$ P (9)	(6) z (5) pomocou $\neg g$; (9) z (8) pomocou $\neg g$
$\square$ $\quad$ $\square$	
(4),(6) $\quad$ (7), (9)	

3 Implementácia

V tejto kapitole vizualizujeme typy, ktoré sme vytvorili v programovacom jazyku Racket a ich vzťahy v rámci našej implementácie pomocou uml triedneho diagramu. Hovoríme aj o jednotlivých súboroch a funkciách, ktoré v implementácii používame.

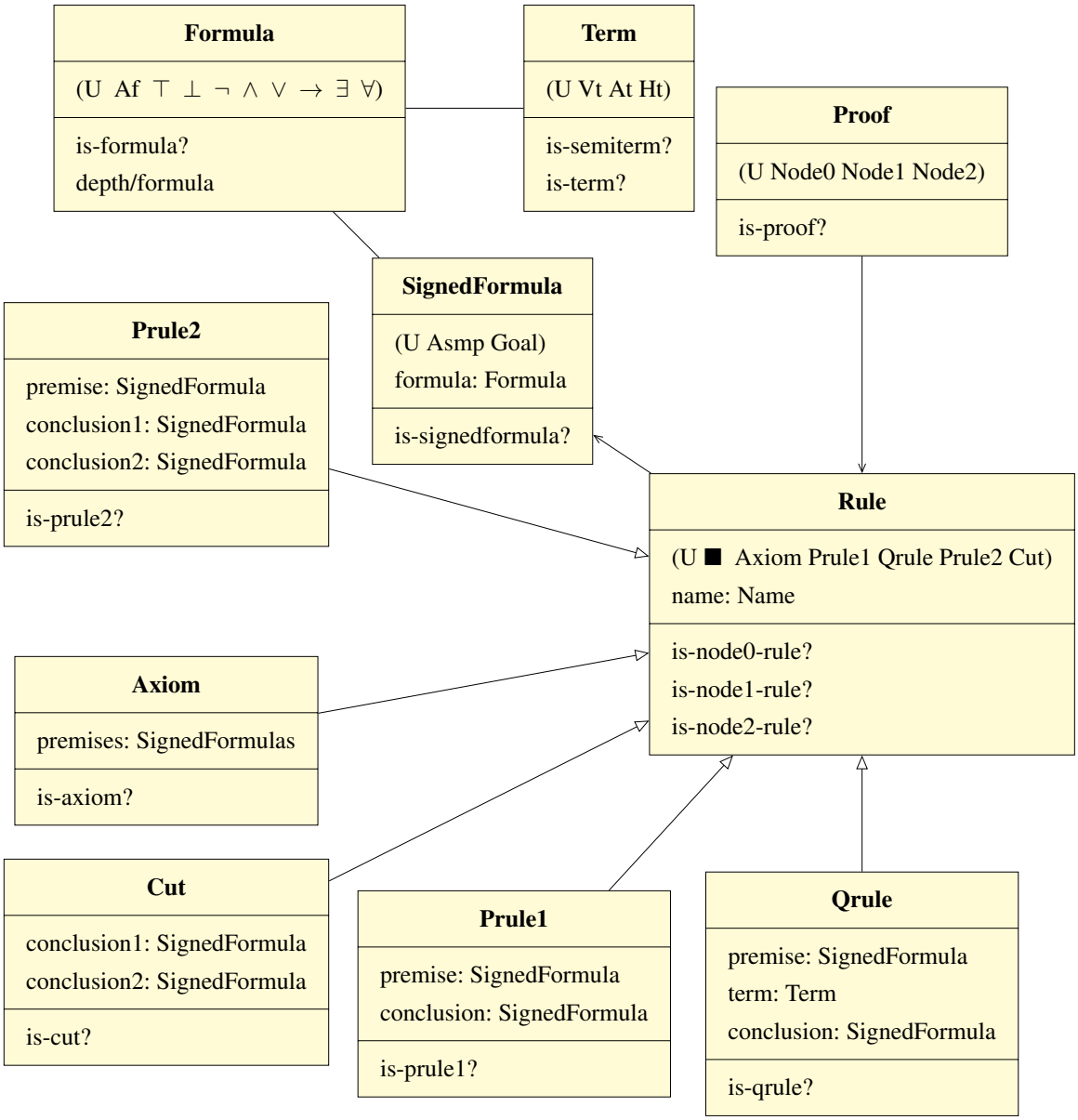
Keď spomíname jednotlivé funkcie v súboroch, tak sú usporiadané v takom poradí ako ich máme v implementácii.

3.1 UML triedny diagram

Typy, ktoré používame v implementácii reprezentujeme v našom uml triednom diagrame pomocou tried. Hodnota typu vyjadreného pomocou zjednotenia U môže nadobúdať jednu z hodnôt typov, ktoré sú v zjednotení U. Napríklad typ Term môže nadobúdať hodnotu typu Vt, At alebo Ht. Typ Vt reprezentuje individuové konštanty, At reprezentuje semitermy (aj funkčné symboly) a Ht reprezentuje Henkinove konštanty.

Tablovú verziu dôkazov reprezentujeme pomocou typu Proof. Na vytvorenie typu Node0 potrebujeme typ Axiom alebo typ ■. Typ ■ je pravidlo bez formuly a označuje neuzavretú vetvu dôkazu. Na vytvorenie typu Node1 potrebujeme typ Prule1 alebo Qrule a jeden typ zo zjednotenia U typu Proof. Typ Qrule reprezentuje kvantifikačné pravidlá. Na vytvorenie typu Node2 potrebujeme Cut alebo Prule2 a dva typy zo zjednotenia U typu Proof.

Typom Af rozumieme atomickú formulu, ktorá potrebuje na vytvorenie 2 argumenty, symbol a pole termov. Na vytvorenie typov $\exists$ a $\forall$, ktoré sú kvantifikačné formuly, potrebujeme 2 argumenty Vt a typ Formula. Typ Sexp, ktorý používame vo funkciách, je skratka pre symbolický výraz. Typom Asmp označujeme formulu, ktorá je predpoklad a typom Goal označujeme formulu, ktorá je cieľ.



Obr. 3.1: UML triedny diagram.

3.2 Súbory

Naša implementácia pozostáva z nasledujúcich súborov:

1. expr.rkt
2. proof.rkt
3. write.rkt
4. read.rkt

5. display.rkt
6. elim.rkt
7. debug.rkt

3.3 Súbor expr.rkt

Súbor expr.rkt definuje typy Formula a Term. Súbor expr.rkt obsahuje aj funkcie, ktoré overujú správnosť vytvorenia týchto typov. Dôležité funkcie, ktoré sme v tomto súbore použili sú:

(: is-semiformula? (-> Formula Boolean)) Funkcia overuje, či sme správne vytvorili semiformulu. Pojem semiformula popisujeme v pododseku 1.2.1.3.

(: is-formula? (-> Formula Boolean)) Funkcia overuje, či sme správne vytvorili formulu. Pojem formula vysvetľujeme na konci pododseku 1.2.1.3.

(: is-semiterm? (-> Term Boolean)) Funkcia overuje, či sme správne vytvorili semiterm. Bližšie pojem semiterm popisujeme v pododseku 1.2.1.2.

```
1 (: is-semiterm? (-> Term Boolean))
2 (define (is-semiterm? t)
3   (match t
4     [(Vt _) #t]
5     [(At f ts)
6     (define n (length ts))
7     (and (= (arity f) n) (are-semiterms? ts))]
8     [(Ht a) (and (is-quantifier-formula? a) (is-formula? a))]
9     [_ (error "is-semiterm?")]))
```

Obr. 3.2: Kód funkcie is-semiterm?.

(: is-term? (-> Term Boolean)) Funkcia overuje, či sme správne vytvorili term. Bližšie pojem term popisujeme v pododseku 1.2.1.2.

(: depth/formula (-> Formula Natural)) Funkcia vráti hĺbku formuly. Hĺbku formuly počítame na základe pododseku 1.2.1.4.

```
(: depth/formula (-> Formula Natural))
(define (depth/formula a)
  (match a
    [(Af _ _) 1]
    [(⊤) 1]
    [(⊥) 1]
    [(¬ b) (+ (depth/formula b) 1)]
    [(∧ b c) (max (+ (depth/formula b) (depth/formula c)) 1)]
    [(∨ b c) (max (+ (depth/formula b) (depth/formula c)) 1)]
    [(→ b c) (max (+ (depth/formula b) (depth/formula c)) 1)]
    [(∃ x b) (+ (depth/formula b) 1)]
    [(∀ x b) (+ (depth/formula b) 1)]
    [_ (error "depth/formula")]))
```

Obr. 3.3: Kód funkcie depth/formula.

(: substitute/term-in-formula (-> Formula Term Term Formula)) Funkcia substituuje term vo formule za iný term. Substitúciu bližšie vysvetľujeme v pododseku 1.2.1.6.

3.4 Súbor proof.rkt

Súbor proof.rkt definuje typ dôkaz a obsahuje funkcie na overenie jeho správnosti a iné funkcie, ktoré pracujú s dôkazmi.

3.4.1 Hlboké nahradenie

Tento odsek obsahuje funkcie, ktoré slúžia na aplikáciu lemy o hlbokom nahradení 2.1.9.

(: dr/formula (-> Formula Term Term Formula)) Funkcia hlboko nahradí term za iný term vo formule a vráti novú formulu.

(: dr/proof (-> Proof Term Term Proof)) Funkcia hlboko nahradí term za iný term v dôkaze a vráti nový dôkaz.

3.4.2 Základne miery na dôkazoch

Funkcie, ktoré v tomto odseku spomíname, používame na výpočet základnych mier na dôkazoch, ktoré sme použili viackrát pri odhadoch v návrhu riešenia.

(: depth/proof (-> Proof Integer)) Funkcia vráti hĺbku dôkazu. Výpočet hĺbky dôkazu bližšie popisujeme v pododseku 1.2.3.4.

```
1 (: depth/proof (-> Proof Integer))
2   (define (depth/proof p)
3     (match p
4     [(Node0 rule) 1]
5     [(Node1 rule p1) (+ (depth/proof p1) 1)]
6     [(Node2 rule p1 p2) (+ (max (depth/proof p1)
7                                 (depth/proof p2)) 1)]))
```

Obr. 3.4: Kód funkcie depth/proof.

(: cut-rank (-> Proof Integer)) Funkcia vráti rezovú hodnosť dôkazu, ktorú počítame na základe pododseku 1.2.3.5.

3.4.3 Overovanie dôkazov

V tomto odseku spomíname, vďaka akým funkciam vieme overovať správnosť vytvorenia dôkazov.

(: is-proof? (-> Proof SignedFormulas Boolean)) Funkcia overuje, či sme správne vytvorili dôkaz. Funkcia is-proof? je najpodstatnejšia funkcia, pomocou ktorej v súbore debug.rkt testujeme správnosť vytvorených dôkazov.

3.5 Súbor write.rkt

Súbor write.rkt obsahuje funkcie určené na transformáciu dôkazov, označené formúl, termov a ďalších typov do symbolických výrazov. Najdôležitejšie funkcie v tomto súbore sú:

(: term->sexp (-> Term Sexp)) Funkcia transformuje term do symbolického výrazu.

(: formula->sexp (-> Formula Sexp)) Funkcia transformuje formulu do symbolického výrazu.

(: proof->sexp (-> Proof Sexp)) Funkcia transformuje dôkaz do symbolického výrazu.

3.6 Súbor read.rkt

Súbor read.rkt používame na čítanie dôkazov a našich definovaných typov zo symbolického výrazu. Analogicky obsahuje funkcie aj pre čítanie iných typov. V podstate tento súbor slúži na opačné funkcie k súboru write.rkt. Na overenie správnosti funkcií sme použili aj porovnanie vstupu funkcie v súbore write.rkt s výstupom príslušnej funkcie v súbore read.rkt. Najdôležitejšie funkcie v tomto súbore sú:

(: sexp->term (-> Sexp Term)) Funkcia načíta term zo symbolického výrazu.

(: sexp->formula (-> Sexp Formula)) Funkcia načíta formulu zo symbolického výrazu.

(: sexp-w/rules->proof (-> Sexp SignedFormulas Rules (Listof (Pairof String Ht)) (Values Proof (Listof (Pairof String Ht))))) Funkcia načítana odvodzovacie pravidlo zo symbolického výrazu.

(: sexp->proof (-> Sexp SignedFormulas (Values Proof (Listof (Pairof String Ht))))) Funkcia načíta dôkaz zo symbolického výrazu. Vstupný symbolický výraz nemusí obsahovať všetky údaje o dôkaze, lebo niektoré sa dajú dopočítať.

3.7 Súbor display.rkt

Súbor display.rkt používame na čitatelné zobrazovanie dôkazov, označených formúl, termov a iných typov. Dôkazy zobrazujeme pomocou tablovej metódy. Najdôležitejšie funkcie v tomto súbore sú:

(: transform/proof->lines (-> Proof (Listof (Pairof Index SignedFormula)) Index Natural (Listof (Pairof String Ht)) (Values Index Lines (Listof (Pairof String Ht))))) Funkcia transformuje typ Proof do typu riadky.

(: display/lines (-> Lines Void)) Funkcia zobrazí označené formuly v štruktúre Lines do riadkov.

(: display/proof (-> Proof SignedFormulas (Listof (Pairof String Ht)) Void)) Funkcia zobrazí dôkaz prostredníctvom tabla. Druhý argument funkcie sú označené formuly čistého sekventu. Nakoniec tretí argument sú jednotlivé pomenovania Henkinových konštánt v dôkaze, ktoré si vieme týmto spôsobom určiť.

3.8 Súbor elim.rkt

V súbore elim.rkt máme implementovanú elimináciu pravidla rezu. Všetky procesy, ktoré s elimináciou pravidla rezu priamo súvisia, sa nachádzajú v tomto súbore.

3.8.1 Inverzia

Inverziu sme implementovali pomocou následujúcich funkcií na základe odseku 2.1.7.

(: invert-unary (-> SignedFormula Proof Proof)) Funkcia aplikuje inverziu na dôkaz a vráti nový dôkaz.

(: invert-binary (-> SignedFormula Proof (Values Proof Proof))) Funkcia aplikuje inverziu na vstupný dôkaz, ktorá vráti dva nové dôkazy.

3.8.2 Redukcia rezu

Funkcie, ktoré v tomto odseku spomíname, používame na redukciu rezu. Redukciu rezu spomíname v sekcii 2.3.

(: replace/axiom (-> Proof Proof SignedFormulas Proof)) Funkcia nahradí poddôkazy v dôkaze, ktoré sa končia axiomatickým pravidlom Ax a nahradia sa novým dôkazom. Redukcia atomickej rezovej formuly je konkrétny prípad 2.3.2 redukcie, v ktorej funkciu replace-/axiom používame.

(: replace/quantifier (-> Proof Proof SignedFormula Name Proof)) Funkcia nahradí postupne zhora nadol v dôkaze poddôkazy, ktoré končia kvantifikačným odvodzovacím pravidlom, ktorého názov píšeme v argumente Name. Funkciu replace/quantifier používame na implementovanie prípadov, ktoré spomíname v odsekoch 2.3.9 a 2.3.10.

(: reduce (-> SignedFormula Proof Proof Proof)) Funkcia aplikuje typ redukcie na základe tvaru rezovej formuly.

```
(: reduce (-> SignedFormula Proof Proof Proof))
(define (reduce c p1 p2)
  (match c
    [(Asmp (Af P ts)) (reduce/Af p1 p2 (Af P ts))]
    [(Asmp (⊤)) (reduce/⊤ p1 p2)]
    [(Asmp (⊥)) (reduce/⊥ p1 p2)]
    [(Asmp (¬ a)) (reduce/¬ p1 p2 a)]
    [(Asmp (∧ a b)) (reduce/∧ p1 p2 a b)]
    [(Asmp (∨ a b)) (reduce/∨ p1 p2 a b)]
    [(Asmp (→ a b)) (reduce/→ p1 p2 a b)]
    [(Asmp (∃ x b)) (reduce/∃ p1 p2 (∃ x b))]
    [(Asmp (∀ x b))  (reduce/∀ p1 p2 (∀ x b))]
    [_ (error "choose-reduce/" c)]))
```

Obr. 3.5: Kód funkcie reduce.

3.8.3 Kváziregularizácia

Následujúce funkcie, ktoré spomíname, používame na algoritmus kváziregularizácie. Algoritmus kváziregularizácie popisujeme v odseku 2.2.3.

(: get-rqcuts (-> Proof Natural (Listof Formula))) Funkcia vráti pole všetkých rezových formúl kvantifikačných rezov s rezovou hodnosťou r.

(: topological-sort (-> (Listof Formula) (Listof Formula) (Listof Formula))) Funkcia vráti pole, ktorý má topologicky utriedené rezové formuly. Spôsob, ako sme rezové formuly utriedili spomíname v odseku 2.2.3.

(: cut-inversion (-> Proof Formula (Values Proof Proof))) Funkcia aplikuje inverziu rezu na dôkaz a vráti dva nové dôkazy. Inverziu rezu popisujeme v odseku 2.2.2.

(: quasiregularization (-> Proof Natural Proof)) Funkcia aplikuje kváziregularizáciu na dôkaz a vráti r-kváziregulárny dôkaz. Algoritmus a vstupné podmienky kváziregularizácie spomíname v odseku 2.2.3.

3.8.4 Eliminácia pravidla rezu

Tento odsek obsahuje funkcie, ktoré slúžia na samotnú elimináciu pravidla rezu. Celý algoritmus eliminácie pravidla rezu vysvetľujeme v sekcii 2.4.

(: eliminate/rcuts (-> Proof Integer Proof)) Funkcia eliminuje všetky rezy s rezovou hodnosťou r.

(: eliminate/all (-> Proof Integer Proof)) Funkcia na vstupe dostane rezovú hodnosť dôkazu a samotný dôkaz. Funkcia postupne eliminuje všetky rezy.

```
1 (: eliminate/all (-> Proof Integer Proof))
2 (define (eliminate/all p r)
3   (cond
4     [(> r 0)
5     (eliminate/all
6     (eliminate/rcuts
7     (quasiregularization p r) r) (- r 1))]
8     [else p]))
```

Obr. 3.6: Kód funckie eliminate/all.

(: eliminate (-> Proof Proof)) Funkcia na vstupe dostane dôkaz čistého sekventu a na výstupe dostane bezrezový dôkaz toho istého sekventu. Funkcia eliminate je centrálna funkcia, na ktorej algoritmus eliminácie pravidla rezu testujeme.

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

Obr. 3.7: Kód funkcie eliminate.

3.9 Súbor debug.rkt

V tomto súbore testujeme správnosť našich funkcií pomocou výpisu do konzoly. Dôkazy, formuly, termy a iné typy, ktoré sme vytvorili, sa nachádzajú v tomto súbore.

4 Testovanie programu

V tejto kapitole opisujeme jednotlivé testy v súbore debug.rkt, ktoré názorne ukazujú vizualizáciu našich dôkazov v tablovej metóde a správnosť našej implementácie algoritmu eliminácie pravidla rezu.

4.1 Vytváranie a vizualizácia dôkazov

Tablová verzia dôkazu π, ktorý neskôr vizualizujeme má tvar
$(a \equiv \mathrm{c}_{\forall x \exists y P(x,y)},\ b \equiv \mathrm{c}_{\exists y \forall x P(x,y)})$:

$$
\begin{array}{rl}
\exists y \forall x P(x,y) \rightarrow \forall x \exists y P(x,y) * (1) & \\
\exists y \forall x P(x,y) \quad (2) & (2) \text{ z } (1) \text{ pomocou } \rightarrow g1 \\
\forall x \exists y P(x,y) * (3) & (3) \text{ z } (1) \text{ pomocou } \rightarrow g2 \\
\exists y P(a,y) * (4) & (4) \text{ z } (3) \text{ pomocou } \forall g \\
\forall x P(x,b) \quad (5) & (5) \text{ z } (2) \text{ pomocou } \exists a \\
P(a,b) \quad (6) & (6) \text{ z } (5) \text{ pomocou } \forall a \\
P(a,b) * (7) & (7) \text{ z } (4) \text{ pomocou } \exists g \\
\square & \\
{\scriptstyle (6),\ (7)} &
\end{array}
$$

Dôkazy v našej implementácii vytvárame pomocou funkcie sexp->proof, lebo je to najjednodušší spôsob ako vytvoriť dôkaz. K danému tablovému dôkazu vyššie vytvoríme rovnaký dôkaz v našej implementácii.

```
(: π Proof)
(define-values (π table/π)
(sexp->proof '(→g1
(→g2
(∀g
(∃a
(∀a (∀ x (∃ y (P^2 x y)))
(∃g (∃ y (∀ x (P^2 x y)))
(□))))))))  (list sf)))
```

Obr. 4.1: Vytváranie dôkazu.

Jednotlivé formuly číslujeme v preorderovom poradí. Na vizualizáciu Henkinových konštánt používame písmená *a*, *b* a *c* s prípadnými dolnými indexami. Vetvy oddeľujeme prázdnym riadkom a súčasne, keď aplikujeme pravidlo, ktoré vytvorí dve vetvy, tak vynecháme riadok. Na konci uvádzame aj informácie o voľných premenných, ktoré sme v dôkaze použili. Dôkaz π vizualizujeme v našej implementácii takto (nad prerušovanou čiarou sú príkazy a pod prerušovanou čiarou je výstup):

```
(display/proof π (list sf) table/π)
(is-proof? π (list sf))

-------------------------------------------
(1) (∃y∀xP(x,y) → ∀x∃yP(x,y))*
(2) ∃y∀xP(x,y) from (1) by →g1
(3) ∀x∃yP(x,y)* from (1) by →g2
(4) ∃yP(a,y)* from (3) by ∀g using (a ≡ c_∀x∃yP(x,y))
(5) ∀xP(x,b) from (2) by ∃a using (b ≡ c_∃y∀xP(x,y))
(6) P(a,b) from (5) by ∀a
(7) P(a,b)* from (4) by ∃g
(8) □ by (7) and (6)

In the place of the free variables we used:
a ≡ c_∀x∃yP(x,y)
b ≡ c_∃y∀xP(x,y)
#t
```

Obr. 4.2: Príklad vizualizácie dôkazu.

4.2 Testovanie implementovaného algoritmu

V tejto sekcii testujeme správnosť našej implementácie algoritmu eliminácie pravidla rezu. Pre lepšiu čitateľnosť zobrazujeme dôkazy pomocou tablovej metódy.

4.2.1 Príklad 1

Predpokladáme, že vstupný dôkaz π má tvar $\left(a \equiv \mathrm{c}_{\forall x P(x)}\right)$:

$$\exists y(P(y) \rightarrow \forall x P(x)) * (1)$$

$P(a) * (2)$	$P(a)$ (5)	Pomocou Cut
$P(a) \rightarrow \forall x P(x) * (3)$	$P(a) \rightarrow \forall x P(x) * (6)$	(3) z (1) pomocou $\exists g$; (6) z (1) pomocou $\exists g$
$P(a)$ (4)	$P(a)$ (7)	(4) z (3) pomocou $\rightarrow g1$; (7) z (6) pomocou $\rightarrow g1$
$\square$	$\forall x P(x) * (8)$	(8) z (6) pomocou $\rightarrow g2$
(2), (4)	$P(a) * (9)$	(9) z (8) pomocou $\forall g$
	$\square$	
	(9), (7)	

Aplikujeme redukcie rezu v prípade, keď rezová formula je atomická. Nahradíme poddôkaz obsahujúci $\square$ v ľavej vetve dôkazu π za poddôkaz (6-9) vrátane $\square$ a dostaneme výsledný dôkaz ρ, ktorý je v tvare $\left(a \equiv \mathrm{c}_{\forall x P(x)}\right)$:

$\exists y(P(y) \rightarrow \forall x P(x)) * (1)$	
$P(a) \rightarrow \forall x P(x) * (2)$	(2) z (1) pomocou $\exists g$
$P(a)$ (3)	(3) z (2) pomocou $\rightarrow g1$
$P(a) \rightarrow \forall x P(x) * (4)$	(4) z (1) pomocou $\exists g$
$P(a)$ (5)	(5) z (4) pomocou $\rightarrow g1$
$\forall x P(x) * (6)$	(6) z (4) pomocou $\rightarrow g2$
$P(a) * (7)$	(7) z (6) pomocou $\forall g$
$\square$	
(7), (5)	

4.2.2 Príklad 2

Predpokladáme, že vstupný dôkaz π má tvar $\left(a \equiv \mathrm{c}_{\forall x P(x)}\right)$:

$\exists y(P(y) \to P(f(y))) * (1)$

$P(a) \to P(f(a)) * (2)$ (2) z (1) pomocou $\exists g$

$P(a)$ (3) (3) z (2) pomocou $\to g1$

$P(f(a)) * (4)$ (4) z (2) pomocou $\to g2$

$\forall x P(x) * (5)$ $\forall x P(x)$ (7) pomocou Cut

$P(a) * (6)$ $P(f(a))$ (8) (6) z (5) pomocou $\forall g$; (8) z (7) pomocou $\forall a$

$\square$ $\square$

(3),(6) (9),(6)

Aplikujeme kváziregularizáciu a dostaneme nový dôkaz:

$\exists y(P(y) \to P(f(y))) * (1)$

$\forall x P(x) * (2)$ $\forall x P(x)$ (7) pomocou Cut

$P(a) \to P(f(a)) * (3)$ $P(a) \to P(f(a)) * (8)$ (3) z (1) pomocou $\exists g$; (8) z (1) pomocou $\exists g$

$P(a)$ (4) $P(a)$ (9) (4) z (3) pomocou $\to g1$; (9) z (8) pomocou $\to g1$

$P(f(a)) * (5)$ $P(f(a)) * (10)$ (5) z (3) pomocou $\to g2$; (10) z (8) pomocou $\to g2$

$P(a) * (6)$ $P(f(a))$ (11) (6) z (2) pomocou $\forall g$; (11) z (7) pomocou $\forall a$

$\square$ $\square$

(4),(6) (11),(10)

Následne aplikujeme unárnu inverziu na poddôkaz π_1:

$\exists y(P(y) \to P(f(y))) * (1)$

$\forall x P(x) * (2)$ $\forall x P(x)$ (7) pomocou Cut

$P(a) * (3)$ $P(a) \to P(f(a)) * (8)$ (3) z (2) pomocou $\forall g$; (8) z (1) pomocou $\exists g$

$P(a) \to P(f(a)) * (4)$ $P(a)$ (9) (4) z (1) pomocou $\exists g$; (9) z (8) pomocou $\to g1$

$P(a)$ (5) $P(f(a)) * (10)$ (5) z (4) pomocou $\to g1$; (10) z (8) pomocou $\to g2$

$P(f(a)) * (6)$ $P(f(a))$ (11) (6) z (4) pomocou $\to g2$; (11) z (7) pomocou $\forall a$

$\square$ $\square$

(3),(5) (11),(10)

Eliminujeme kvantifikačný rez:

$$\exists y(P(y) \to P(f(y))) * (1)$$
$$P(a) \to P(f(a)) * (2) \qquad \text{(2) z (1) pomocou } \exists g$$
$$P(a) \quad (3) \qquad \text{(3) z (2) pomocou } \to g1$$
$$P(f(a)) * (4) \qquad \text{(4) z (2) pomocou } \to g2$$

$$P(f(a)) * (5) \qquad P(f(a)) \quad (10) \qquad \text{pomocou Cut}$$
$$P(f(a)) * (6) \qquad \square$$
$$P(f(a)) \to P(f(a)) * (7) \qquad {\scriptstyle (4),(10)}$$
$$P(f(a)) \quad (8) \qquad \text{(8) z (7) pomocou } \to g1$$
$$P(f(a)) * (9) \qquad \text{(9) z (7) pomocou } \to g2$$
$$\square$$
$${\scriptstyle (9),(8)}$$

Eliminujeme atomický rez a tým dostaneme výsledny bezrezový dôkaz ρ:

$$\exists y(P(y) \to P(f(y))) * (1)$$
$$P(a) \to P(f(a)) * (2) \qquad \text{(2) z (1) pomocou } \exists g$$
$$P(a) \quad (3) \qquad \text{(3) z (2) pomocou } \to g1$$
$$P(f(a)) * (4) \qquad \text{(4) z (2) pomocou } \to g2$$
$$P(a) * (5)$$
$$P(f(a)) \to P(f(a)) * (6)$$
$$P(f(a)) \quad (7) \qquad \text{(7) z (6) pomocou } \to g1$$
$$P(f(a)) * (8) \qquad \text{(8) z (6) pomocou } \to g2$$
$$\square$$
$${\scriptstyle (5),(3)}$$

Záver

Cieľ tejto práce bola implementácia algoritmu eliminácie pravidla rezu v sekventovom kalkule LK^h. Algoritmus eliminácie pravidla rezu sme implementovali pomocou kváziregularizácie striedanou so štandardnou redukciou. Dôkazy sme implementovali v tablovej metóde pre jednodušiu čitatelnosť.

Náš algoritmus eliminácie pravidla rezu dostane na vstupe dôkaz π čistého sekventu a vráti bezrezový dôkaz ρ toho isté sekventu.

Celú implementáciu sme naprogramovali v programovacom jazyku Racket, ktorý je známy dialekt programovacieho jazyka Lisp, lebo sa v ňom ľahko vytvoril vlastný jazyk, a naviac je multi-paradigmický. Spočiatku sme implementovali redukciu pre rezy, ktoré nie sú kvantifikačné. Následne sme zrealizovali aj elimináciu kvantifikačných rezov, ktoré sme najprv testovali na regulárnych dôkazoch. Po implementovaní kváziregularizácie sme začali testovať algoritmus eliminácie pravidla rezu aj pre neregulárne dôkazy. Tým, že sme overili správnosť nášho algoritmu eliminácie pravidla rezu na netriviálnych dôkazoch v testovacej kapitole, sme splnili cieľ tejto bakalárskej práce.

Henkinové konštanty sme v dôkazoch zobrazovali dlhý čas nečitateľným spôsobom pomocou c_{QxA} ale nakoniec sme doimplementovali iný spôsob, ako zobrazovať Henkinove konštanty pomocou písmen a, b, c s prípadnými dolnými indexami. Používateľ môže dokonca sám zadať na vstupe, aké písmená patria ktorým Henkinovým konštantám.

Zjednodušili sme spôsob, ako zadávať vstupy v súbore debug.rkt. Spočiatku sme implementovali spôsob, ako načítať typy zo symbolického výrazu. Na vytvorenie dôkazov netreba zadávať všetky údaje, lebo sa môžu dopočítať.

Námetom na nové práce je implementácia rovno v sekventovom kalkule LK^h. Napríklad pomocou grafického rozhrania namiesto jednoduchého výpisu do konzoly, ktorý sme používali, aby sme dosiahli čitateľnejšiu vizualizáciu dôkazov. Taktiež je priestor na vylepšenia v oblasti inteligentného zobrazovania Henkinových konštánt na vstupe. Ďalšia oblasť, ktorá by sa mohla zlepšiť, je spracovanie chýb, aby nebolo potrebné vždy vyhodiť výnimku pri zadaní chybového vstupu.

Literatúra

[1] Harold Abelson and Gerald Jay Sussman. *Structure and interpretation of computer programs*. The MIT Press, 1996.

[2] R Kent Dybvig. *The Scheme programming language*. Mit Press, 2009.

[3] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, Eli Barzilay, Jay McCarthy, and Sam Tobin-Hochstadt. The racket manifesto. In *1st Summit on Advances in Programming Languages (SNAPL 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[4] J. Komara and P. J. Voda. Syntactic reduction of predicate tableaux to propositional tableaux. In *Proceedings of TABLEAUX '95*, number 918 in LNAI, pages 231–246. Springer Verlag, 1995.

[5] Ján Komara. Efficient elimination of Skolem functions in LK^{h}. *Archive for Mathematical Logic*, 61(3):503–534, May 2022.

[6] Maroš Malý. Štrukturálna vnútorná skolemizácia. Bakalárska práca, Fakulta matematiky, fyziky a informatiky, 2020.

[7] R. M. Smullyan. *First Order Logic*. Springer Verlag, 1968.

[8] Vítězslav Švejdar. *Logika: neúplnost, složitost a nutnost*. Academia-nakladatelství Akademie věd ČR, 2002.