UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

WEBOVÝ EDITOR REZOLVENČNÝCH DOKAZOV

Bakalárska práca

2020 **Norbert Jurík**

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

WEBOVÝ EDITOR REZOLVENČNÝCH DOKAZOV

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	FMFI KAI – Katedra aplikovanej informatiky
Školiteľ/Školiteľka:	Mgr. Ján Kľuka, PhD.
Konzultant:	RNDr. Jozef Šiška, PhD.

Bratislava 2020 **Norbert Jurík**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Norbert Jurík
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: Webový editor rezolvenčných dôkazov
Web-Based Editor of Resolution Proofs

Anotácia: V rámci predmetu Matematika (4) – Logika pre informatikov vyučujeme okrem iných aj rezolvenčný kalkul pre logiku prvého rádu a s ním súvisiace úpravy formúl do klauzálnej formy a skolemizáciu. Existujúce nástroje na kontrolu a poskytovanie spätnej väzby k formálnym dôkazom, či už vytvorené pre náš ale aj iné podobné kurzy, buď nepodporujú tento formálny systém alebo automatizujú niektoré jeho časti, čo zabraňuje študentom v získaní konkrétnej skúsenosti s nimi.

Cieľ: Implementovať webový nástroj na kontrolu rezolvenčných dôkazov v logike prvého rádu.

Literatúra: Genesereth, M., Kao, E.: Introduction to Logic. Third Edition. Morgan & Claypool, 2017.
Nyitraiová, A.: Educational tools for first order logic. Bakalárska práca – Univerzita Komenského, Bratislava, 2018.
Onódy, Z.: A proof assistant for first-order logic. Bakalárska práca – Univerzita Komenského, Bratislava, 2018.

Kľúčové slová: rezolvencia, unifikácia, skolemizácia, webová aplikácia na strane klienta, nástroj na podporu výučby

Vedúci: Mgr. Ján Kľuka, PhD.
Konzultant: RNDr. Jozef Šiška, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 04.10.2019

Dátum schválenia: 07.10.2019

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Chcem sa poďakovať svojmu školiteľovi Mgr. Jánovi Kľukovi, PhD. za cennú pomoc, rady, konzultácie a čas, ktorú mi venoval počas písania bakalárskej práce.

Abstrakt

Práca sa zaoberá tvorbou webovej aplikácie - editora, ktorý ponúka študentom precvičovanie pravidiel rezolvencie a faktorizácie v logike prvého rádu, ktoré sú aplikovateľné v rezolvenčnom dôkaze nesplniteľnosti. Na základe používateľom definovaného prvorádového jazyka webový editor overuje syntax klauzúl v dôkaze a validuje správnosť krokov rezolvenčného dôkazu. Dôkaz nie je odvodzovaný automaticky, ale je zadávaný používateľom, ktorý tak po každom kroku získava spätnú väzbu o jeho správnosti. Dôkaz vrátane jazyka je možné exportovať do textového súboru a v prípade potreby opätovne importovať naspäť do aplikácie. Aplikácia je naprogramovaná ako JavaScript aplikácia s využitím knižníc React, Redux a Bootstrap. Editor bol úspešne otestovaný v rámci cvičení a domácej úlohy na predmete Matematika (4) – Logika pre informatikov.

Kľúčové slová: rezolvencia, unifikácia, skolemizácia, webová aplikácia na strane klienta, nástroj na podporu výučby

Abstract

This thesis deals with the creation of a web application - an editor that offers students to practice the rules of resolution and factoring in the first order logic, which are applicable in the resolution proof of unsatisfiability. Based on a user-defined first order language, the web editor verifies the syntax of the clauses in the proof and validates the correctness of the steps of the resolution proof. The proof is not derived automatically, but is entered by the user, who thus receives feedback on its correctness after each step. Proof, including language, can be exported to a text file and imported back into the application if necessary. The application was developed in JavaScript using the React, Redux and Bootstrap libraries. The editor was successfully tested in the subject of Mathematics 4 - Logic for Informatics in the form of exercise and homework.

Key words: resolution, unification, skolemization, client-side web application, support tool for teaching

Obsah

Slovník termínov

Disjunkcia	Logická disjunkcia (používa sa pre ňu symbol ∨) je výroková spojka, ktorej hodnota je pravda práve vtedy, keď aspoň jeden zo vstupných operandov je pravda. Dva výroky spojené disjunkciou tvoria výrok (nazývaný aj disjunkciou týchto dvoch výrokov), ktorý je pravdivý práve vtedy, ak je pravdivý aspoň jeden zo spájaných výrokov. V hovorovom jazyku sa väčšinou vyjadruje spojkou alebo.
Konjunkcia	Logická konjunkcia (používa sa pre ňu symbol ∧) je výroková spojka. Dva výroky spojené konjunkciou tvoria výrok (nazývaný aj konjunkciou týchto dvoch výrokov), ktorý je pravdivý práve vtedy, ak sú pravdivé obidva spájané výroky. V hovorovom jazyku sa väčšinou vyjadruje spojkou a.
Negácia	Negácia je logické prevrátenie hodnoty alebo výroku. Označuje sa znakom ¬ pred výrokom. Ak máme výrok A, potom negácia výroku A môže vyzerať takto ¬A.
DNF formula	Formula je zadaná v disjunktívnej normálnej forme, ak má tvar disjunkcie konečného počtu konjunkcií, ktoré obsahujú konečný počet výrokových premenných alebo ich negácií.
CNF formula	Formula je zadaná v konjunktívnej normálnej forme, ak má tvar konjunkcie konečného počtu disjunkcií, ktoré obsahujú konečný počet výrokových premenných alebo ich negácií.
NNF formula	Formula je v negačnom normálnom tvare (NNF) vtedy a len vtedy, keď neobsahuje implikáciu a pre každú jej podformulu ¬A platí, že A je atomická formula.
Komplementárny literál	Dva literály sú komplementárne, ak majú tvar p a ¬p.

Úvod

Novodobé informačné a komunikačné technológie prinášajú zmeny do vzdelávacieho systému. Umožňujú študentom vnímať poznatky viacerými zmyslami a tým, v porovnaní s tradičnými formami vzdelávania, umožňujú dosiahnuť vyšší efekt vo vzdelávaní. V súčasnej praxi sa v súvislosti s modernými formami vzdelávania často stretávame s webovými stránkami či aplikáciami, ktoré nahrádzajú klasické formy študijnej literatúry. Webové aplikácie sú schopné implementovať aj zložité aplikačné logiky, ktoré nie je nutné inštalovať, ale je možné ich využívať priamo vo webovom prehliadači. Táto časť vzdelávacieho procesu sa stáva dôležitým prezentačným prvkom a významným komunikačným prostriedkom medzi pedagógom a študentom.

V rámci štúdia predmetu Matematika (4) – Logika pre informatikov sme sa stretli s rôznymi výučbovými nástrojmi, ktoré sa venovali rôznym témam logiky prvého rádu. Zatiaľ sa však žiaden z nich nevenoval problematike rezolvenčného kalkulu, a práve táto problematika je cieľom tejto bakalárskej práce. Vytvoriť webový editor na kontrolu tvorby rezolvenčných dôkazov v logike prvého rádu, ktorý poskytne študentom okamžitú spätnú väzbu pri riešení úloh.

Webový editor nebude poskytovať výsledné riešenie, ale postupne overovať správnosť používateľom zadávaných formúl s ohľadom na definovaný jazyk a predchádzajúce kroky dôkazu. Samotné vyhodnocovanie klauzúl prebieha na pozadí aplikácie po každom relevantnom vstupe od používateľa. Priebežné vyhodnocovanie dôkazov podporuje u študentov osvojenie si správneho uvažovania pri riešení úloh typu rezolvenčného dokazovania, nakoľko každý akcia používateľa je sprevádzaná spätnou väzbou z aplikácie formou validačných správ. Aplikácia tiež bude kontrolovať syntaktickú správnosť zadaných klauzúl, symbolov jazyka, či ich prípadné duplicity. Prehodnotenie už zadaných prvkov jazyka, či krokov dôkazu, je umožnené používateľovi v ľubovoľnej fáze riešenia úlohy rezolvenčného kalkulu. Riešenie bude možné vyexportovať a následne naimportovať opätovne do aplikácie, čo môže uľahčiť prácu pedagógom pri kontrole prípadných domácich zadaní študentov, ktoré sú odovzdané v elektronickej podobe.

Práca je členená do nasledujúcich častí. Prvá kapitola je venovaná teoretickým poznatkom logiky prvého rádu so zameraním sa na syntax a sémantiku logiky prvého rádu, formalizáciou v logike a princípy tvorby rezolvenčných dôkazov. V druhej kapitole sú popísané súčasné možnosti použitia webových technológií. Prácam s podobnou tematikou je venovaná tretia kapitola. Štvrtá kapitola detailne rozoberá požiadavky na webový editor a celkový návrh aplikácie s ohľadom na dostupné technológie. S konkrétnou implementáciou sa čitateľ oboznámi v piatej kapitole, kde sú uvedené detaily, týkajúce sa použitých implementačných tried, aplikačnej logiky a používateľského rozhrania. Posledná, šiesta kapitola ponúka čitateľovi výsledky testovania webového editora študentami FMFI, ktorí zaslali spätnú väzbu cez publikovaný dotazník k webovému editoru.

1 Logika prvého rádu a rezolvencia

Logika [1] je vedná disciplína, ktorá študuje formy usudzovania. Je to veda o správnom usudzovaní [3]. V logike sa študujú také schémy usudzovania, ktoré sú správne (korektné) bez ohľadu na pravdivosť alebo nepravdivosť ich zložiek. Z podobných, či diametrálne odlišných tvrdení, inak povedané od seba závislých či nezávislých výrokov, usudzovaním sa získavajú nové tvrdenia, a teda odvodzujú sa nové výroky. Výroky sú vety hovorovej reči, o pravdivosti ktorých má zmysel uvažovať. Tieto atomické výroky konštatujú všeobecné stavy, hovoria o vlastnostiach alebo vzťahoch objektov. Hovorová reč je v logike formalizovaná na symbolickej úrovni, pri ktorej sa ignoruje konkrétny obsah jednotlivých výrokov. Z týchto dôvodov sa logika môže označiť ako formálna alebo matematická logika.

Pre logiku sú zaujímavé predovšetkým zloženiny výrokov, nie iba výroky samotné. Základným princípom v logike je používanie symbolov a ich zoskupovanie pomocou logických spojok (jazykových prostriedkov typu „...a...“, „...alebo...“ “, „ak..., potom...“, ...) do väčších celkov nazývaných formuly, ale aj formalizácia procesu transformácie danej formuly na inú formulu matematickými metódami. Používa sa striktný matematický systém odvodzovania nových formúl pomocou povolených operácií z jednoduchších formúl (axióm).

Aj logika ako vedná disciplína sa vyvíja a existujú viaceré logiky, od výrokovej, ktorá veľmi zjednodušuje prirodzený jazyk len na vlastnosti a vzťahy objektov, cez predikátovú, ktorá kvantifikuje nielen objekty, ale aj vzťahy, či neklasické logiky, ako Lukasiewiczoca logika a intuicionistická logika, či fuzzy logika.

Logika prvého rádu [1] je rozšírením výrokovej logiky, ktorá sa zaoberá vlastnosťami výrokových spojok, kvantifikátorov a predikátových symbolov popisujúcich vzťahy (relácie). Predikátový symbol je prísudok (alebo celá prísudková časť), ktorý vyjadruje činnosť, stav alebo vlastnosť popisovaného objektu.

1.1 Syntax logiky prvého rádu

Syntax [1] sú pravidlá budovania viet v jazyku. Jazyk logiky prvého rádu jazyk obsahuje prostriedky na spájanie základných výrokov o vzťahoch a vlastnostiach (výrokové spojky) a prostriedky na vyjadrovanie toho, koľko objektov má vlastnosti alebo je vo vzťahoch (kvantifikátory). Obsahuje vyjadrovacie prostriedky, pomocou ktorých sme schopní rozlišovať jednotlivé objekty (indivídua), ich vlastnosti a vzťahy medzi nimi.

Symbolmi jazyka logiky prvého rádu $\mathcal{L}$ sú [1] *symboly (indivíduových) premenných* z nejakej nekonečnej spočítateľnej množiny $V_{\mathcal{L}}$ (označujeme ich x, y, ...), *mimologické symboly,* konkrétne *symboly konštánt* z nejakej spočítateľnej množiny $C_{\mathcal{L}}$ (a, b, ...), *funkčné symboly* z nejakej spočítateľnej množiny $F_{\mathcal{L}}$ (f , g, ...), *predikátové symboly* z nejakej spočítateľnej množiny $P_{\mathcal{L}}$ (P, R, ...), ďalej *logické symboly*, ako *logické spojky: unárna* ¬, *binárne* ∧, ∨, →, *symbol rovnosti* ≐, *existenčný kvantifikátor* ∃ a *všeobecný kvantifikátor* ∀ a nakoniec *pomocné symboly* (,) a , (ľavá, pravá zátvorka a čiarka).

Množiny $V_{\mathcal{L}}$, $C_{\mathcal{L}}$, $F_{\mathcal{L}}$, $P_{\mathcal{L}}$sú vzájomne disjunktné.

Logické a pomocné symboly sa nevyskytujú v symboloch z $V_{\mathcal{L}}$, $C_{\mathcal{L}}$, $F_{\mathcal{L}}$, $P_{\mathcal{L}}$.

Každému symbolu $S \in P_{\mathcal{L}}$ je priradená *arita* ar(S) ∈ N^+.

Jazyk predikátovej logiky [3] $\mathcal{L}$ je definovaný nad symbolmi jazyka nasledovne:

Termy sú postupnosti symbolov označujúce objekty*:*

1. Indivíduové premenné a indivíduové konštanty sú termy;
2. ak f je n-árny symbol funkcie a $t_1, t_2, ..., t_n$ sú termy, potom výraz $f(t_1,t_2, ...,t_n)$ je term;
3. žiadne iné postupnosti symbolov nie sú termy.

Atomické formuly:

1. Ak P je n-árny predikátový symbol, $t_1, t_2, ..., t_n$ sú termy, potom výraz $P(t_1,t_2, ...,t_n)$ je atomická formula;
2. žiadne iné symboly nie sú atomické formuly.

Formuly:

1. Každá atomická formula je formula;
2. Ak A je formula, tak aj $\neg A$ je formula (negácia A);
3. Ak A a B sú formuly, tak aj $(A \wedge B)$,$(A \vee B)$,$(A \rightarrow B)$ sú formuly (konjunkcia, disjunkcia, implikácia A a B;
4. Ak x je indivíduová premenná a A je formula, tak aj $\exists xA$ a $\forall xA$ sú formuly (existenčná a všeobecná kvantifikácia formuly A vzhľadom na x);
5. Nič iné nie je formula.

Základnou entitou jazyka logiky prvého rádu je *formula* (označovaná písmenami *A, B, C,...*).

Teória je množina formúl.

Za predpokladu [1], že $A_1, A_2, ..., A_n$ je konečná postupnosť formúl, *konjunkciu* postupnosti formúl $A_1, A_2, ..., A_n$, teda $(((A_1 \wedge A_2) \wedge A_3) \wedge \cdots \wedge A_n)$, skrátene zapisujeme:

$$(A_1 \wedge A_2 \wedge A_3 \wedge \cdots \wedge A_n), \text{ prípadne } \bigwedge_{i=1}^{n} A_i.$$

Konjunkciu prázdnej postupnosti formúl (n = 0) označujeme $\top$ a chápeme ju ako ľubovoľnú tautológiu, napríklad $(p_1 \vee \neg p_1)$.

Disjunkciu postupnosti formúl $A_1, A_2, ..., A_n$, teda $(((A_1 \vee A_2) \vee A_3) \vee \cdots \vee A_n)$, skrátene zapisujeme:

$$(A_1 \vee A_2 \vee A_3 \vee \cdots \vee A_n), \text{ prípadne } \bigvee_{i=1}^{n} A_i.$$

Disjunkciu prázdnej postupnosti formúl označujeme $\bot$ alebo $\square$ a chápeme ju ako ľubovoľnú nesplniteľnú formulu, napríklad $(p_1 \wedge \neg p_1)$.

Ak n = 1, samotná formula A_1 je konjunkciou aj disjunkciou jednoprvkovej postupnosti formúl A_1 zároveň.

Táto bakalárska práca je zameraná na problematiku rezolvencie, čo je metóda dokazovania nad klauzálnymi teórami. Preto je potrebné definovať špeciálne typy formúl: literál, klauzula a klauzálna teória.

Nech $\mathcal{L}$ je jazyk logiky prvého rádu [1].

Literál je atomická formula $P(t_1, ..., t_m)$ jazyka $\mathcal{L}$ alebo jej negácia $\neg P(t_1, \ldots, t_m)$.

Klauzula je všeobecný uzáver disjunkcie literálov, teda uzavretá formula jazyka $\mathcal{L}$ v tvare $\forall x_1 \cdots \forall x_k(L_1 \vee \cdots \vee Ln)$ (skrátene$(\forall \vec{x} \bigvee_{i=1}^{n} L_1)$), kde $L_1,...,L_n$ sú literály a $x_1,...,x_k$ sú všetky voľné premenné formuly $L_1 \vee \cdots \vee L_n$.

Klauzula môže byť aj jednotková $(\forall \vec{x}\ L_1)$ alebo prázdna ($\square$).

Klauzálna teória je množina klauzúl $\{C_1, ..., C_n\}$. Môže byť tvorená aj jedinou klauzulou alebo byť prázdna. Všeobecné kvantifikátory v zápise klauzúl budeme zanedbávať. Teda namiesto $\forall x1 \cdots \forall x_n(L_1 \vee \cdots \vee L_m)$ píšeme iba $L_1 \vee \cdots \vee L_m$.

1.2 Formalizácia v logike prvého rádu

Jednoduchú formalizáciu si predstavme ako nahradenie všeobecných a konkrétnych objektov zo sveta symbolmi premenných a konštánt.

Príklad:

Máme k dispozícii tieto informácie, tvrdenie z reálneho sveta:

1. Neexistuje taký človek, ktorý by sa každému páčil.

Riešenie:

$V_{\mathcal{L}}$ = {x, y}, kde *x, y* sú dvaja rôzni ľudia všeobecne

$C_{\mathcal{L}}$ = {}, tvrdenie neobsahuje žiadnu konštantu, žiadneho konkrétneho človeka

$P_{\mathcal{L}} = \{P^2\}$, kde predikátový symbol *P* vyjadruje vzťah medzi 2 ľuďmi (predikátový symbol s aritou 2) a znamená, že človek *x* sa páči človeku *y*

Pomocou logických symbolov, ktoré viažeme na konštanty a premenné, definujeme riešenie nasledovne:

1. $\neg \exists x \forall y\, P(x, y)$

1.3 Sémantika logiky prvého rádu

Sémantika jazyka logiky prvého rádu sa zaoberá interpretáciou jazyka $\mathcal{L}$ a pravdivostným hodnotením formúl.

Nech $\mathcal{L}$ je jazyk relačnej logiky prvého rádu [1].

Štruktúrou pre jazyk $\mathcal{L}$ nazývame dvojicu $\mathcal{M} = (M, i)$, kde M je neprázdna množina, nazývaná *doména* štruktúry $\mathcal{M}$, *i* je zobrazenie, nazývané *interpretačná funkcia* štruktúry $\mathcal{M}$, ktoré každému symbolu konštanty *c* jazyka $\mathcal{L}$ priraďuje prvok $i(c) \in M$, každému funkčnému symbolu *f* jazyka $\mathcal{L}$ s aritou n priraďuje funkciu $i(f): M^n \rightarrow M$ a každému predikátovému symbolu *P* jazyka $\mathcal{L}$ s aritou n priraďuje množinu $i(P) \subseteq M^n$.

Nech $\mathcal{M} = (D, i)$ je štruktúra pre jazyk $\mathcal{L}$. *Ohodnotenie indivíduových premenných* je ľubovoľná funkcia $e : V_{\mathcal{L}} \rightarrow D$ (priraďuje premenným prvky domény). Nech ďalej *x* je indivíduová premenná z $\mathcal{L}$ a *v* je prvok D. Zápisom $e(x/v)$ označíme ohodnotenie indivíduových premenných, pre ktoré platí $e(x/v)(x) = v$ a súčasne $e(x/v)(y) = e(y)$, ak *y* je iná premenná ako *x*.

Termy s funkčnými symbolmi môžu byť vnorené, vyhodnocujeme ich rekurzívne [1]:

Nech $\mathcal{M} = (M, i)$, je štruktúra pre jazyk logiky prvého rádu $\mathcal{L}$, nech *e* je ohodnotenie premenných. *Hodnotou termu t* v štruktúre $\mathcal{M}$ pri ohodnotení premenných *e* je prvok z *M* označovaný $t^{\mathcal{M}}[e]$ a zadefinovaný induktívne pre všetky premenné *x*, konštanty *a*, každú aritu *n*, všetky funkčné symboly *f* s aritou *n*, a všetky termy $t_1, ..., t_n$ nasledovne:

- $x^{\mathcal{M}}[e] = e(x)$,

- $a^{\mathcal{M}}[e] = i(a)$,
- $(f(t_1, \dots, t_n))^{\mathcal{M}}[e] = e(f)(t_1^{\mathcal{M}}[e], \dots, t_n^{\mathcal{M}}[e])$.

Nech $\mathcal{M}$ = *(D, i)* je štruktúra pre jazyk $\mathcal{L}$, *e* je ohodnotenie premenných. Relácia štruktúra $\mathcal{M}$ spĺňa formulu *X* pri ohodnotení *e* (skrátene $\mathcal{M} \vDash X[e]$) má nasledovnú definíciu [1]:

- $\mathcal{M} \vDash t_1 \doteq t_2[e] vtt\ t_1^{\mathcal{M}}[e] = t_2^{\mathcal{M}}[e]$,
- $\mathcal{M} \vDash \mathrm{P}(t_1, \dots, t_n)[e] vtt\ (t_1^{\mathcal{M}}[e], \dots, t_n^{\mathcal{M}}[e] \in i(P)$,
- $\mathcal{M} \vDash \neg A[e] vtt\ \mathcal{M} \nvDash \neg A[e]$,
- $\mathcal{M} \vDash (A \wedge B)[e] vtt\ \mathcal{M} \vDash A[e]\ a\ zároveň\ \mathcal{M} \vDash B[e]$,
- $\mathcal{M} \vDash (A \vee B)[e] vtt\ \mathcal{M} \vDash A[e]\ alebo\ \mathcal{M} \vDash B[e]$,
- $\mathcal{M} \vDash (A \rightarrow B)[e] vtt\ \mathcal{M} \nvDash A[e]\ alebo\ \mathcal{M} \vDash B[e]$,
- $\mathcal{M} \vDash \exists \mathrm{x}\, \mathrm{A}[e] vtt\ pre\ nejaký\ prvok\ m \in D\ máme\ \mathcal{M} \vDash A[e(x/m)]$,
- $\mathcal{M} \vDash \forall \mathrm{x}\, \mathrm{A}[e] vtt\ pre\ nejaký\ prvok\ m \in D\ máme\ \mathcal{M} \vDash A[e(x/m)]$,

pre všetky arity $n > 0$, všetky predikátové symboly P s aritou n, všetky termy $t_1, \dots, t_n$, všetky premenné *x* a všetky formuly *A*, *B* jazyka $\mathcal{L}$.

Súhlasne s autorom [1], nech *X* je formula jazyka $\mathcal{L}$ a nech *S* je množina formúl jazyka $\mathcal{L}$. Formula *X* je splniteľná vtt aspoň jedna štruktúra $\mathcal{M}$ pre $\mathcal{L}$ spĺňa *X* pri aspoň jednom ohodnotení *e*. Množina formúl *S* je splniteľná vtt aspoň jedna štruktúra $\mathcal{M}$ pre $\mathcal{L}$ spĺňa *S* pri aspoň jednom ohodnotení *e*. Formula *X* (množina formúl *S*) je nesplniteľná vtt nie je splniteľná.

Nech *X* je formula v jazyku $\mathcal{L}$, nech *S* je množina formúl v jazyku $\mathcal{L}$. Formula X (prvorádovo) vyplýva z *S* (skrátene *S* |= *X*) vtt pre každú štruktúru $\mathcal{M}$ pre $\mathcal{L}$ a každé ohodnotenie *e* platí, že ak $\mathcal{M}$ spĺňa *S* pri *e*, tak $\mathcal{M}$ spĺňa *X* pri *e*.[1]

Nech *X* je formula a *S* je množina formúl v spoločnom jazyku $\mathcal{L}$. Potom z S vyplýva X vtt S ∪ {¬X} je nesplniteľná.[1]

1.4 Rezolvenčné dôkazy

Pri konštrukcii sémantickej interpretácie logiky prvého rádu sa používajú rôzne prístupy, napríklad hilbertovský kalkul, sémantické tablá, rezolvencia. Nakoľko cieľom bakalárskej práce je implementácia úpravy formule do klauzálnych teórií a kontrola tvorby rezolvenčných dôkazov, priblížime si bližšie teoretické poznatky v tejto oblasti. Súhlasne

s autorom [13] metóda rezolvencie je dokazovací systém, ktorý umožňuje dokázať nesplniteľnosť klauzálnej teórie postupným odvodzovaním nových klauzúl pravidlami rezolvencie a faktorizácie, kým nedospejeme k prázdnej klauzule. Táto metóda sa dá aplikovať aj mechanicky a bola prvou prakticky úspešnou metódou automatického dokazovania

Nech *A, B* sú postupnosti symbolov, *σ* je *substitúcia*, teda zobrazenie premenných na termy. *Aσ* je výsledok aplikácie substitúcie na formulu *A*, teda formula, ktorá vznikne súčasným nahradením každého voľného výskytu premennej *x* vo formule *A* termom *σ(x).*

Substitúcia σ je *unifikátorom A* a *B* vtedy a len vtedy, keď $A\sigma = B\sigma$.

Nech *C* a *D* sú prvorádové klauzuly, nech *A* a *B* sú atómy, nech *L* a *K* sú literály.

Rezolvencia je odvodzovacie pravidlo:

$$\frac{A \vee C \qquad \neg B \vee D}{(C\theta \vee D)\sigma},$$

kde *σ* je unifikátor $A\theta$ a *B* a θ je premenovanie premenných.

Ďalším odvodzovací pravidlom je *faktorizácia*:

$$\frac{L \vee K \vee C}{(L \vee C)\sigma},$$

kde σ je unifikátor *L* a *K*.

Nech *T* je klauzálna teória. Zamietnutím *T* (angl. refutation) je každá konečná postupnosť klauzúl $Z = (C_1, C_2, \ldots, C_n)$, kde C_n = a každá klauzula C_i , $1 \leq i \leq n$, je [1]:

• prvkom *T*, alebo

• odvodený pravidlom rezolvencie z klauzúl C_j a C_k, ktoré sa v Z nachádzajú pred C_i , alebo

• odvodený pravidlom faktorizácie z klauzuly C_j , ktorá sa v Z nachádza pred C_i.

Rezolvencia je korektná a úplná v nasledujúcom zmysle [1]:

Nech *T* je klauzálna teória. Potom existuje zamietnutie *T* vtt *T* je nesplniteľná.

1.5 Úprava do klauzálnej teórie a skolemizácia

Klauzálnymi teóriami [1] sa dajú formalizovať mnohé tvrdenia. Implikácie sa dajú vyjadriť disjunkciami a negáciami, konjunkciu v konzekvente, konjunkciu v antecedente viacerými literálmi v klauzule. Namiesto existenčného kvantifikátora môžeme pomenovať objekt konštantou alebo funkciou, ktorej dáme ako argumenty súvisiace objekty.

Na použitie rezolvencie na akúkoľvek teóriu je potrebné ju najprv previesť na rovnako splniteľnú klauzálnu teóriu. To je možné dosiahnuť nasledujúcim algoritmom:

TI: Implikácie nahradíme disjunkciami.

TN: Negačný normálny tvar (NNF): Presunieme negácie k atómom.

TV: Premenujeme premenné tak, aby každý kvantifikátor viazal inú premennú ako ostatné kvantifikátory.

TS: Skolemizácia: Existenčné kvantifikátory nahradíme substitúciou nimi viazaných premenných za Skolemove konštanty/aplikácie Skolemovych funkcií na všeobecne príslušné kvantifikované premenné.

TP: Prenexný normálny tvar (PNF): presunieme všeobecné kvantifikátory na začiatok formuly.

TD: Konjunktívny normálny tvar (CNF): distribuujeme disjunkcie do konjunkcií.

TK: Odstránime konjunkcie rozdelením konjunktov do samostatne kvantifikovaných klauzúl.

Popíšme si jednotlivé kroky detailnejšie. Krok TI znamená nahradenie implikácie disjunkciou. Používame nasledovné pravidlo:

$$(A \rightarrow B) \Leftrightarrow (\neg A \vee B)$$

Formula X je v negačnom normálnom tvare (NNF) vtedy a len vtedy, keď neobsahuje implikáciu a pre každú jej podformulu $\neg A$ platí, že A je atomická formula [1].

Formulu bez implikácií do NNF upravíme pomocou

- de Morganovych zákonov:

$$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$$

$$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$$

- pravidla dvojitej negácie:

$$\neg \neg A \Leftrightarrow A$$

- pravidiel pre negáciu kvantifikátorov:

$$\neg \exists x A \Leftrightarrow \forall x \neg A$$

$$\neg \forall x A \Leftrightarrow \exists x \neg A$$

Významnú rolu v procese rezolvenčého dokazovania hrá skolemizácia. Skolemizácia [1] vytvorí ekvisplniteľnú teóriu, ostatné úpravy sú ekvivalentné.

Skolemizácia [1] je úprava formuly X v NNF, pri ktorej:

• každý výskyt podformuly $\exists y\, A$, ktorý sa nachádza v X mimo všetkých oblastí platnosti všeobecných kvantifikátorov nahradíme formulou

$$A\{y \to c\}$$

pre nový symbol konštanty c, nazývaný *Skolemova konštanta*;

• každý výskyt podformuly $\exists y\, A$, ktorý sa nachádza v X v oblasti platnosti všeobecných kvantifikátorov premenných $x_1, ..., x_n$

$X = \cdots \forall x_1(\cdots \forall x_2(\cdots \forall x_n(\cdots \exists y\, A \cdots) \cdots) \cdots) \cdots$

nahradíme formulou

$$A\{y \to f(x_1, x_2, ..., x_n)\}$$

pre nový funkčný symbol f, nazývaný *Skolemova funkcia*.

Skolemove konštanty a funkcie pomenúvajú objekty, ktorých existenciu formula požaduje.

Formula predikátovej logiky je v Skolemovej forme, ak je sentenciou (má len viazané premenné), neobsahuje existenčné kvantifikátory a jej jadro je konjunkcia klauzúl.

Pokračujeme konverziou formuly do PNF.

Formula X je v prenexnom normálnom tvare (PNF) vtedy a len vtedy, ak má tvar $Q_1x_1\, Q_2x_2 \cdots Q_nx_n\, A$, kde $Q_i \in \{\forall, \exists\}$, x_i je premenná a A je formula bez kvantifikátorov [1].

Skolemizovanú formulu v NNF upravíme do PNF opakovanou aplikáciou nasledujúcich transformácií [1]:

- ak x nemá voľný výskyt v B,

  $\forall x\, A \wedge B \Leftrightarrow \forall x\, (A \wedge B)$ $\qquad$ $B \wedge \forall x\, A \Leftrightarrow \forall x\, (A \wedge B)$

  $\forall x\, A \vee B \Leftrightarrow \forall x\, (A \vee B)$ $\qquad$ $B \vee \forall x\, A \Leftrightarrow \forall x\, (B \vee A)$

- ak x má voľný výskyt v B a y je nová premenná, platí:

  $\forall x\, A \wedge B \Leftrightarrow \forall y\, A\{x \to y\} \wedge B$ $\qquad$ $B \wedge \forall x\, A \Leftrightarrow B \wedge \forall y\, A\{x \to y\}$

  $\forall x\, A \vee B \Leftrightarrow \forall y\, A\{x \to y\} \vee B$ $\qquad$ $B \vee \forall x\, A \Leftrightarrow B \vee \forall y\, A\{x \to y\}$

Na záver distribuujeme disjunkcie do konjunkcií a odstránime konjunkcie rozdelením konjunktov do samostatne kvantifikovaných klauzúl.

2 Použité technológie

Pri tvorbe informačných systémov je dôležitá voľba technológii, od ktorých sa následne odvíja vývoj samotnej aplikácie. Nejde len o výber vhodných nástrojov pre riešenie konkrétnej problematiky, ale aj možnosť ich efektívneho prepojenia a spolupráce.

2.1 Webové technológie

Aktuálnym trendom pri tvorbe informačných systémov dostupných širokej verejnosti sú webové technológie. Pomocou webového prehliadača môžeme ľahko pristupovať k webovým aplikáciám. Súčasné webové aplikácie zabezpečujú značnú časť funkcionality vykonávaním programov priamo v prehliadači a označujú sa ako front-endové. Pri tvorbe takýchto webových aplikácií vývojári pracujú primárne s HTML, CSS a JavaScript-om. Pre tieto technológie existuje množstvo nástrojov a frameworkov, ktoré im uľahčujú prácu a je možné rozširovať ich o ďalšie voľne dostupné komponenty a knižnice. Medzi najpoužívanejšie JavaScript frameworky a knižnice patria: React, Angular, Vue.js, Ember.js a mnohé iné.

2.1.1 React

React (tiež známy ako React.js) [6] je technológia vyvinutá a používaná spoločnosťami Instagram a neskôr Facebook s ambicióznym cieľom: umožniť programátorovi vytvárať frontend webovej aplikácie deklaratívnym spôsobom; programátor definuje, čo sa má na základe daných dát (tzv. aplikačného stavu) zobraziť a o synchronizáciu a prechody medzi klientom a serverom sa stará React. Hlavnou ideou tejto knižnice je vytvorenie komponentov, ktoré sa dajú použiť na viacerých miestach v aplikácii. Ak užívateľ alebo systém vykoná nejakú akciu a zmení sa obsah stránky - prehliadač prekreslí len konkrétny komponent. Komponent je objekt pozostávajúci z elementov, ako sú tlačidlá, textové návestia, či používateľom zadané textové reťazce

Napríklad jednoduchý komponent – tlačidlo s dvomi atribútmi (reakciou na kliknutie `increment` a počítadlom `counter` pre hodnotu) by mohol byť repezentovaný nasledovne:

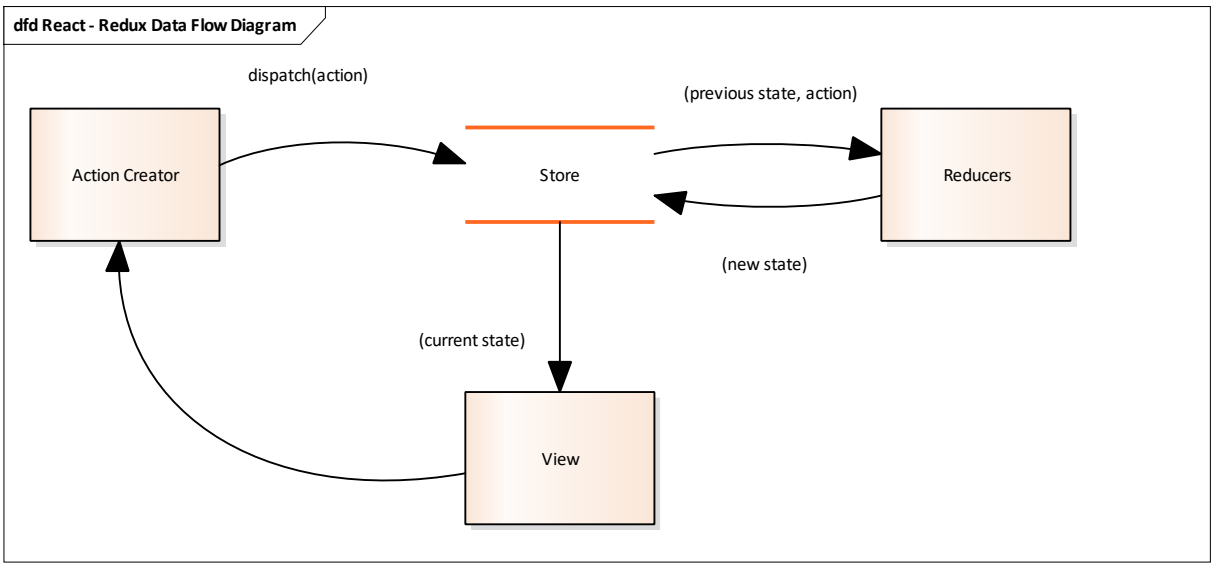
```
const Counter = ({ counter, increment }) => {
   return (<div>
      <div> {counter} </div>
      <button onClick={increment}> +1 </button>
   </div>);
};
```

2.1.2 Redux

Redux [8] je stavový kontajner, určený pre tvorbu dynamických webových aplikácií. Jeho použitie je vhodné napríklad pre spojenie s knižnicou React, ktorá slúži len na tvorbu používateľského rozhrania, ktoré sa generuje práve na základe stavu. Stav aplikácie je objekt, do ktorého si aplikácia ukladá dáta, ako sú napríklad vstupy od používateľa, číselné hodnoty, či zložitejšie dátové štruktúry.

Akcie používateľa, ako napríklad pridanie nového kroku, či zadanie vstupnej textovej hodnoty, sú odchytávané v aplikácii. Akcie popisujú iba to, čo sa v aplikácii stalo, ale neopisujú, ako sa mení stav aplikácie. Na tento účel sú definované reduktory. Reduktory určujú, ako sa zmení stav aplikácie v reakcii na akcie odoslané z komponentov. Prevezmú predchádzajúci stav a akciu a vrátia nasledujúci stav. Stav je zapuzdrený v reduxovom úložisku, do ktorého komponenty posielajú akcie a kontajnerové komponenty z neho dostavajú dáta.

Tok akcií a dát je znázornený na *Obrázok 1*.



Obrázok 1 Tok akcií a dát v React – Redux aplikácii [15]

V nadväznosti na vyššie definovaný príklad tlačidla, by číselný prírastok na príslušnom tlačidle mohol znamenať zmenu stavu, resp. uloženie novej výslednej hodnoty. Funkcia `increment()` reprezentuje akciu, `mapStateToProps` previaže stav s vlastnosťami, `mapDispatchToProps` obslúži definovanú akciu a funkcia `connect()` ich spojí s komponentom. Po odchytení akcie, by došlo k zvýšeniu hodnoty a zmene stavu cez `reducer`.

```
function increment() {
   return { type: 'INCREMENT' };
};

const mapStateToProps = (state) => {
   return { counter: state };
};

const mapDispatchToProps = (dispatch) => {
   return {
      increment: () => dispatch(increment())
   };
};

connect(mapStateToProps, mapDispatchToProps)(Counter);

const reducer = (state = 0, action) => {
   switch (action.type) {
      case 'INCREMENT': return state + 1
   };
};
```

2.1.3 Bootstrap

Bootstrap [7] patrí k najznámejším CSS frameworkom, ktorý poskytuje sadu komponentov pre GUI s dôrazom na ich responzivitu na rôznych zariadeniach. Bootstrap ponúka responzívne prvky, ako tlačidlá, nadpisy, a pod. a je možné ich škálovať podľa toho, čo na konkrétnej webovej stránke potrebujeme.

2.2 Parser formúl

Vstupom pre našu aplikáciu sú formuly a symboly jazyka logiky prvého rádu, ktoré je potrebné syntakticky analyzovať a vytvárať ich objektové reprezentácie. Na to nám poslúži balík js-fol-parser [9], sada syntaktických analyzátorov (parserov) pre jazyky prvého rádu s mnohými spôsobmi písania spojok a niekoľkými alternatívnymi gramatikami. Parsery zanalizujú používateľom zadané reťazce a na základe definovaných tried vytvoria požadované objekty, s ktorými je možné ďalej pracovať. Parsery sú generované z gramatík pomocou generátora peg.js. Pôvodná verzia vznikla v rámci bakalárskej práce Milana Cifru [12] a následne boli parsery oddelené do samostatného balíka [9] a rozšírené vedúcim tejto práce.

2.3 Platforma pre vývoj aplikácie

Na vývoj informačných systémov, ako aj webových stránok, webových aplikácií, webových služieb a mobilných aplikácií je vhodné integrované vývojové prostredie Microsoft Visual Studio [10]. Obsahuje editor kódu Visual Studio Code, ktorý má zabudovaný debugger a množstvo ďalších vstavaných nástrojov. Podporuje rôzne programovacie jazyky, vrátane HTML, CSS a JavaScript-u. Plne vyhovuje pre potreby vývoja webovej aplikácie.

3 Práce s podobnou tematikou

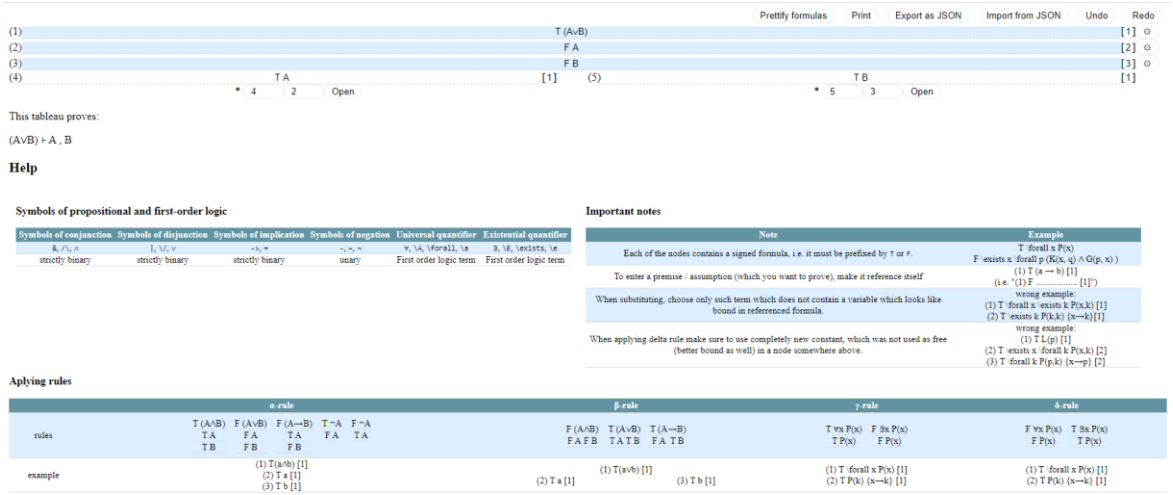
Problematike tvorby rôznych typov dôkazov v logike prvého rádu sa venovali aj iné študentské práce. Najpríbuznejšie dostupné práce (študentské ale aj iné aplikácie) boli podrobené preskúmaniu a analýze za účelom identifikácie overených foriem správania sa a ich použitia. To pozitívne bude následne použité pri tvorbe webového editora rezolvenčných dôkazov.

3.1 Editor tablových dôkazov

Výsledkom študentskej práce Educational tools for first order logic [4] je webová aplikácia na strane klienta, ktorá poskytuje používateľom, študentom, spätnú väzbu pri dôkazoch analytickým tablom pri úprave formúl logiky prvého rádu. Dôkaz je vizualizovaný ako strom a každý vrchol obsahuje nejakú formulu označenú značkou pravdivosti T alebo nepravdivosti F. Formuly sú odvodzované aplikovaním pravidiel: α / Alfa, β / Beta, γ / Gama a δ / Delta. Pri prechode stromom cez jednotlivé vrcholy sa vyhodnocujú formuly podľa pravidiel. Ak sa identifikuje nejaké pravidlo Alfa, tak sa vygeneruje jedna vetva ako nasledovník so zoznamom formúl. Ak sa identifikuje pravidlo Beta, tak sa vygenerujú 2 vetvy ohodnotené formulami β1 a β2. Alfa a Beta pravidlá neboli predmetom tejto práce. Autorka sa zamerala na implementáciu Gama a Delta pravidiel, ktoré slúžia na odstraňovanie kvantifikátorov. Pravidlá Gama riešia kvantifikátory $\forall$ a $\neg\exists$. Pravidlá Delta upravujú vo formule kvantifikátory $\exists$ a $\neg\forall$.

Aplikácia nevytvára dôkaz automaticky, ale validuje každý krok dôkazu cez vstup zadaný používateľom. Aplikácia graficky zvýrazňuje chyby a každú chybu vypíše pri príslušnej formule.

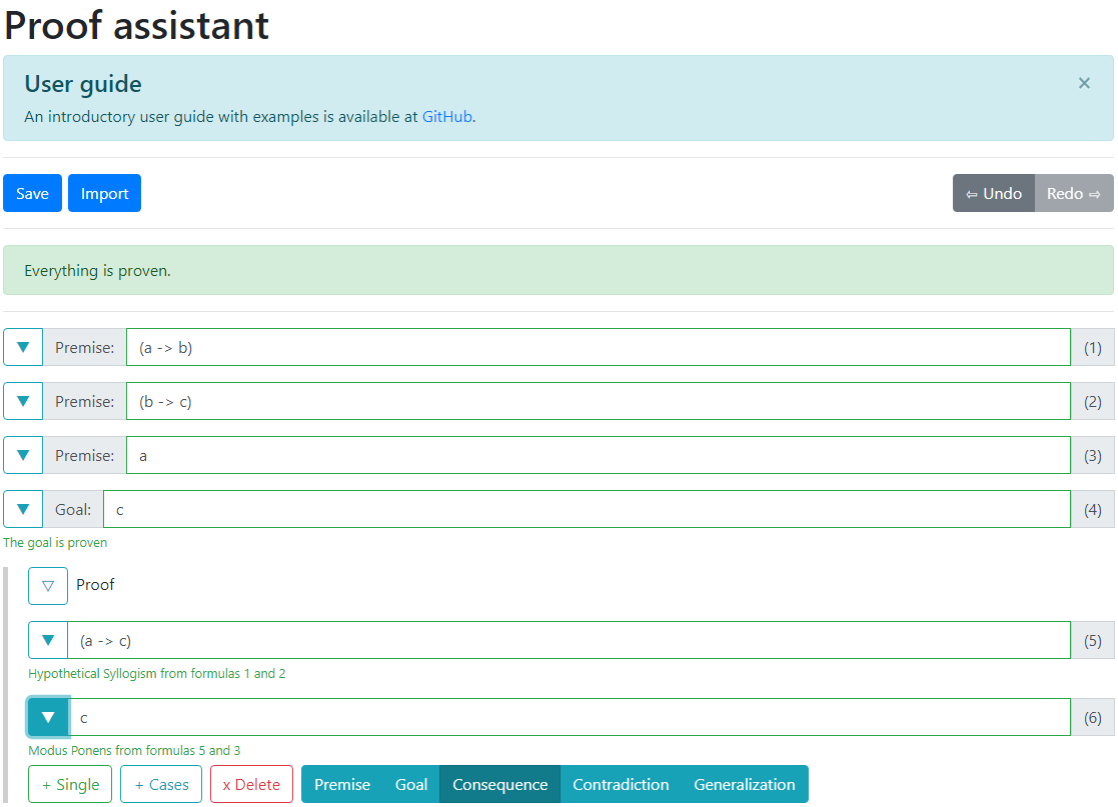
V práci sa autorka zamerala aj na substitúciu formúl a vylepšenie editora o undo a redo, kroky vzad a vpred, pri validácii formúl. Aplikácia je vhodná pre naučenie sa dokazovania podľa metódy analytického tabla. Grafické rozhranie nie je veľmi používateľsky prívetivé, ale aplikácia ako editor formúl s postupným vyhodnocovaním krokov dôkazu podľa analytického tabla spĺňa svoju edukatívnu funkciu.



Obrázok 2 Ukážka študentskej práce Educational tools for first order logic [4]

3.2 Dokazovací asistent pre logiku prvého rádu

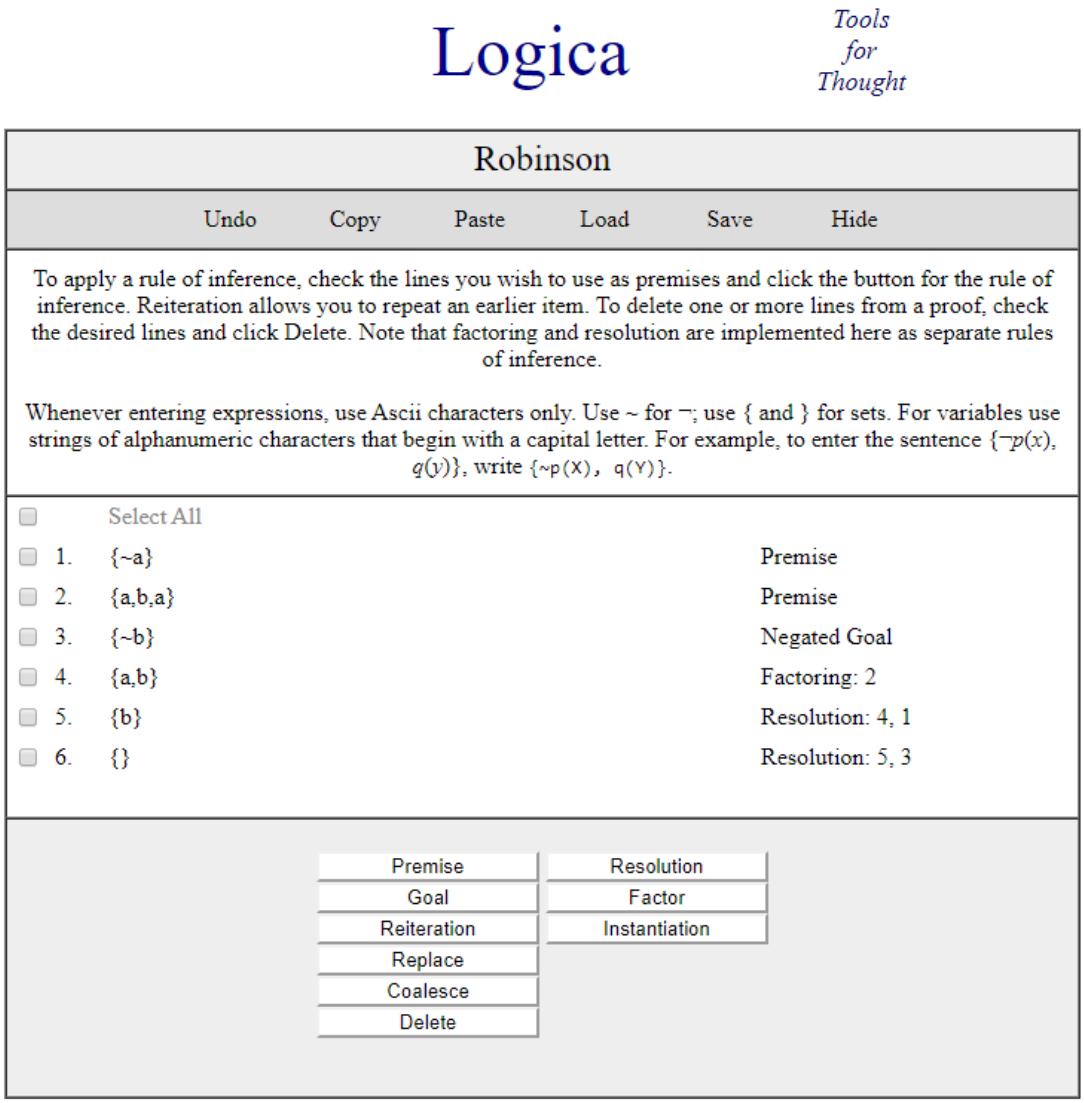
V študentskej práci A proof assistant for first-order logic [5] sa autor zameral na vytvorenie webovej aplikácie pre posilnenie matematického štýlu uvažovania študentov pri dokazovaní tvrdení. Implementovaný dokazovací asistent podporuje tri typy dôkazov, a to priamy dôkaz, dôkaz analýzou prípadov a dôkaz sporom. Obsahuje pevnú sadu pravidiel, bez možnosti konfigurácie. Určenie pravidla, podľa ktorého sa dá formula odvodiť z doterajších, je automatizované. Dokazovač pozná veľa pravidiel a ekvivalentných úprav. Z dôvodu automatizácie sa môže ľahšie stať, že používateľ dospeje nesprávnou úvahou k správnemu záveru, ktorý mu dokazovací asistent potvrdí na základe iného, korektného pravidla. Používateľ si to nemusí všimnúť a utvrdí sa v nesprávnej úvahe. Tiež sa môže stať, že používateľ uvažuje správne, ale napriek širokému repertoáru pravidiel dokazovač jeho domnienku nevie potvrdiť. Používateľ potom môže nadobudnúť dojem, že je jeho úvaha nesprávna, aj keď je iba nedostatočne podrobná. Na druhej strane je tento prístup pohodlnejší a rýchlejší ako explicitne uvádzanie pravidiel.



Obrázok 3 Ukážka študentskej práce A proof assistant for first-order logic [5]

3.3 Rezolvenčný editor Robinson

Rezolvenčný editor Robinson [11], vytvorený na Stanford University, sa taktiež venuje rezolvenčým dôkazom. Obsahuje základné kroky ako predpoklad, cieľ, opakovanie, výmenu, zlúčenie, rezolvenciu, faktorizáciu a substitúciu, avšak priveľa častí je zautomatizovaných. Celú úpravu do CNF aj so skolemizáciou, či počítanie unifikátorov vyhodnocuje samotný nástroj. Ako pomôcka na dosiahnutie výsledku je síce efektívny, no študent si pri tom neprecvičí všetky potrebné kroky rezolvencie.



Obrázok 4 Ukážka z aplikácie Robinson [11]

3.4 Zhrnutie

Vyššie uvedené práce majú svoje silné a slabé stránky. Ich analýzou sme dospeli k záveru, že webový editor, ktorý je predmetom tejto bakalárskej práce, bude implementovaný bez automatizovaných krokov, ako editor vstupov zadaných používateľom, aby bola podporená edukačná funkcia editora ako učebnej pomôcky a viedla používateľa k osvojeniu si teoretických vedomostí a k samostatnej práci, obdobne ako v práci Educational tools for first order logic [4]. V práci A proof assistant for first-order logic [5] a Robinson [6] sú niektoré kroky dokazovania vynechané alebo automatizované, čo môže viesť študentov k nesprávnemu osvojeniu si teórie. V navrhovanej práci nebudú žiadne kroky rezolvenčného dôkazu vynechané a postup bude v súlade s teoretickými znalosťami v danej oblasti. Dôraz je potrebné klásť aj na grafické rozhranie s prijateľnou nápovedou pre používateľa.

4 Analýza a návrh systému

4.1 Požiadavky na systém

Základom celkového návrhu implementácie informačného systému sú požiadavky. Cieľom je zostaviť základný rámec požadovanej funkcionality, definovať funkčné požiadavky na webový editor na tvorbu a kontrolu rezolvenčných dôkazov v logike prvého rádu.

Táto webová aplikácia má študentom umožniť získanie konkrétnej skúsenosti s rezolvenciou a faktorizáciou, negenerovať kroky dôkazu automaticky, práve naopak, rozvíjať samostatné myslenie používateľa, vyhnúť sa automatizácii dôkazu a poskytnúť relevantnú spätnú väzbu s označením chyby.

Webový editor je dostupný pre ľubovoľného používateľa, ktorý má prístup na server, na ktorom je aplikácia publikovaná. Práca s webovým editorom je bez nutnosti prihlásenia sa alebo registrácie. Rozhranie taktiež umožňuje vkladanie aplikácie ako komponentu do iných aplikácií.

Editor poskytuje používateľovi možnosť definovať vlastný jazyk logiky prvého rádu, bližšie popísaný v kap. 1, v rozsahu definovania konštánt, funkcií a predikátov a ich arity. Vďaka tomu je výklad zrozumiteľný aj pre ďalších posudzovateľov správnosti dôkazu.

Používateľ zadáva postupne predpoklady a jednotlivé kroky dôkazu, definované v kap. 1.4. Zadanie kroku znamená napísanie formuly ako predpokladu, či uvedenie výslednej formuly po aplikovaní pravidiel faktorizácie alebo rezolvencie, vrátane uvedenia premisy referenciou na príslušné poradové číslo kroku. Nepovinne je možné zadať podmienky premenovania (premenných na premenné) a/alebo podmienok unifikácie (premenných na termy). Po každej vstupnej aktivite používateľa nasleduje prehodnotenie krokov na základe už zadaných krokov dôkazu, ktoré sú späté s krokom, v ktorom nastala aktivita. Pri zmene už existujúceho pravidla, premenovaní, unifikácii, referencie na iný krok, či ľubovoľnej inej zmene kroku, aplikácia opätovne prehodnotí súvisiace kroky dôkazu čo najefektívnejšie podľa toho, aká zmena nastala. V prípade, že aplikácia identifikuje chybu v použití pravidiel, používateľovi sa zobrazí relevantná chybová hláška.

Používateľ má možnosť vrátenia sa späť do predchádzajúceho stavu (undo), až do úplného začiatku a následne sa posunúť zase vpred (redo), za predpokladu, že nedôjde k zmene niektorého existujúceho stavu.

Editor spĺňa požiadavky na prácu, ako vloženie nového kroku na koniec dôkazu, editácia existujúceho, či kroky vpred a vzad (undo, redo), ale aj vymazanie, pridanie nového kroku medzi existujúce, či zmena poradia už definovaných krokov.

Vytvorené alebo rozpracované zadanie si môže používateľ vyexportovať z aplikácie a uložiť v súbore JSON, vo formáte ktorý je nezávislý od počítačovej platformy a používa sa na ukladanie a prenos textu a dátových objektov. V prípade potreby je možné opätovne naimportovať vytvorené zadanie do webového editora a pokračovať v práci s webovým editorom.

Používateľské grafické rozhranie bude riešené ako jedna ucelená webová stránka, ktorá bude funkčne spôsobilá pre zobrazenie a interpretáciu v súčasnosti najpoužívanejšími webovými prehliadačmi. Bude to dynamická stránka, ktorá sa automaticky prispôsobí svojím vzhľadom rozsahu riešenej úlohy, bez obmedzenia na počet symbolov jazyka, či krokov dôkazu. Grafická podoba stránky bude rešpektovať účel a zameranie celého webového editora a bude rozdelená do logických celkov, ako sú editor jazyka, editor dôkazu a sekcia pre funkčné tlačidlá.

4.2 Návrh štruktúry aplikácie

Z pohľadu implementácie aplikácia rozlišuje dva základné prvky:

- Komponenty – poskytované knižnicou React a slúžia na tvorbu používateľského rozhrania, ktoré sa geneneruje, prekresľuje práve na základe stavu;
- Stav – udržiavaný cez stavový kontajner Redux a vhodný pre dynamickú prácu s webovou stránkou.

Na najvyššej vrstve sú komponenty členené na:

Editor jazyka (Language): Prostredníctvom tohto komponentu používateľ zadáva symboly jazyka – konštanty, funkcie a predikáty. Komponent poskytuje vstupné elementy na definovanie týchto symbolov, ako aj na zobrazovanie chybových hlášok.

Editor dôkazu (Proof): V komponente editora dôkazu je možné pridávať kroky dôkazu, definovať typ kroku, či ide o predpoklad, rezolvenciu alebo faktorizáciu. Obsahuje komponenty na zadanie formuly v závislosti od zvoleného typu kroku v textovom tvare a poskytuje používateľovi spätnú väzbu. V prípade potreby zobrazí chybovú hlášku. Nechýba ani komponent v zmysle tlačidla na pridanie nového kroku dôkazu, zmazania kroku, či preusporiadanie už zadaných krokov.

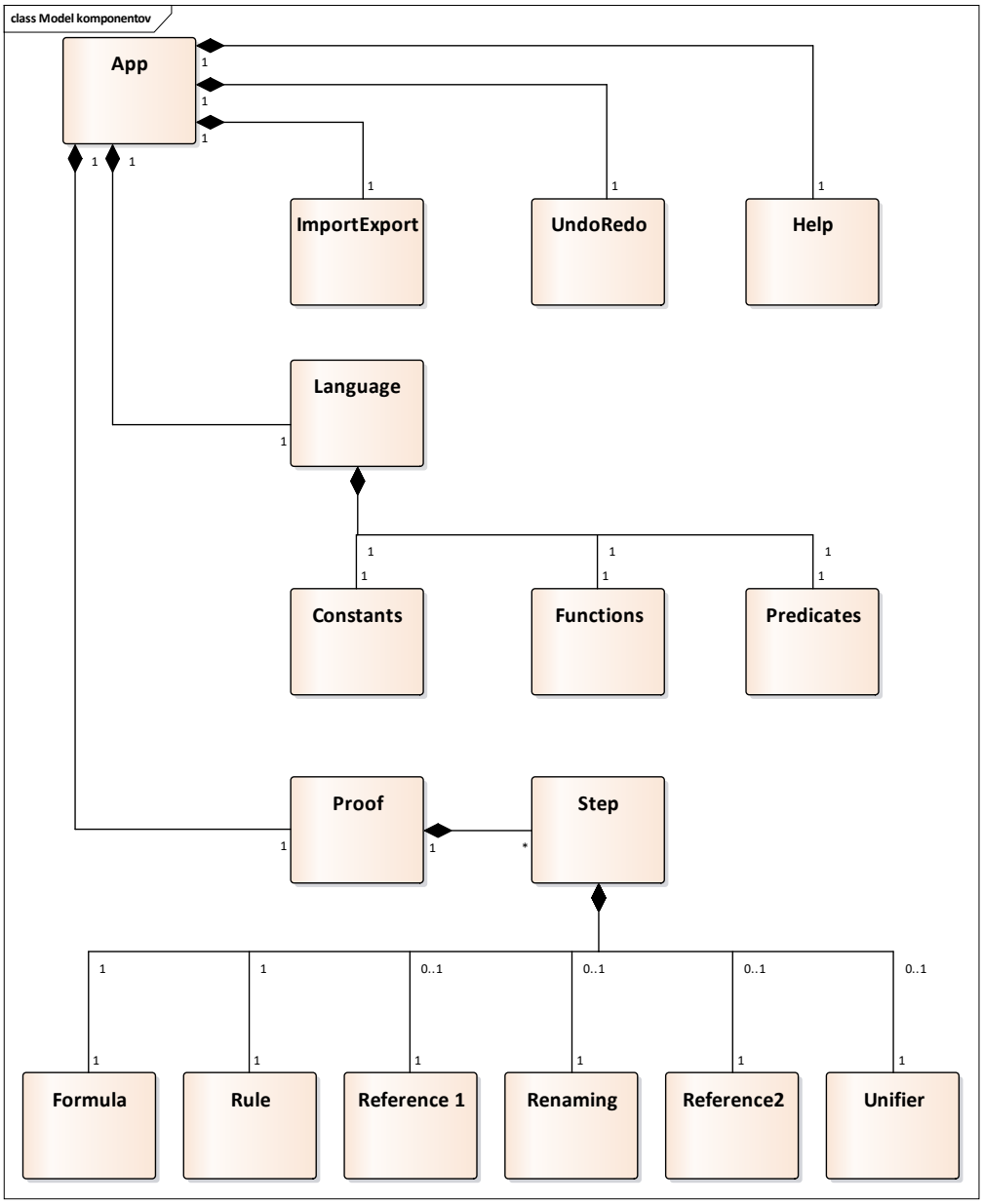
UndoRedo komponent: Tlačidlá pre vrátenie sa do predchádzajúceho stavu aplikácie, čiže posun vzad a následne posun vpred, za predpokladu, že používateľ nerealizoval aktivitu, ktorá by zmenila stav aplikácie, sú používateľovi k dispozícii.

ImportExport komponent: Elementy tohto komponentu prezentujú používateľovi možnosť exportovať a importovať stav aplikácie. Definovaný jazyk, štruktúru a formuly sa budú dať uložiť (exportovať) do počítača používateľa, a naspäť importovať.

Help komponent: Sekcia s nápovedami pre požívateľa, ako správne definovať a oddeľovať symboly jazyka, či aké alternatívne zápisy logických spojok možno použiť.

Po zadefinovaní komponentov, teda toho, ako bude vyzerať stránka (grafika, chybové hlášky a pod.), je potrebné definovať správanie sa aplikácie cez akcie tak, aby boli zastrešené všetky požiadavky na systém, definované v kap. 4.1.

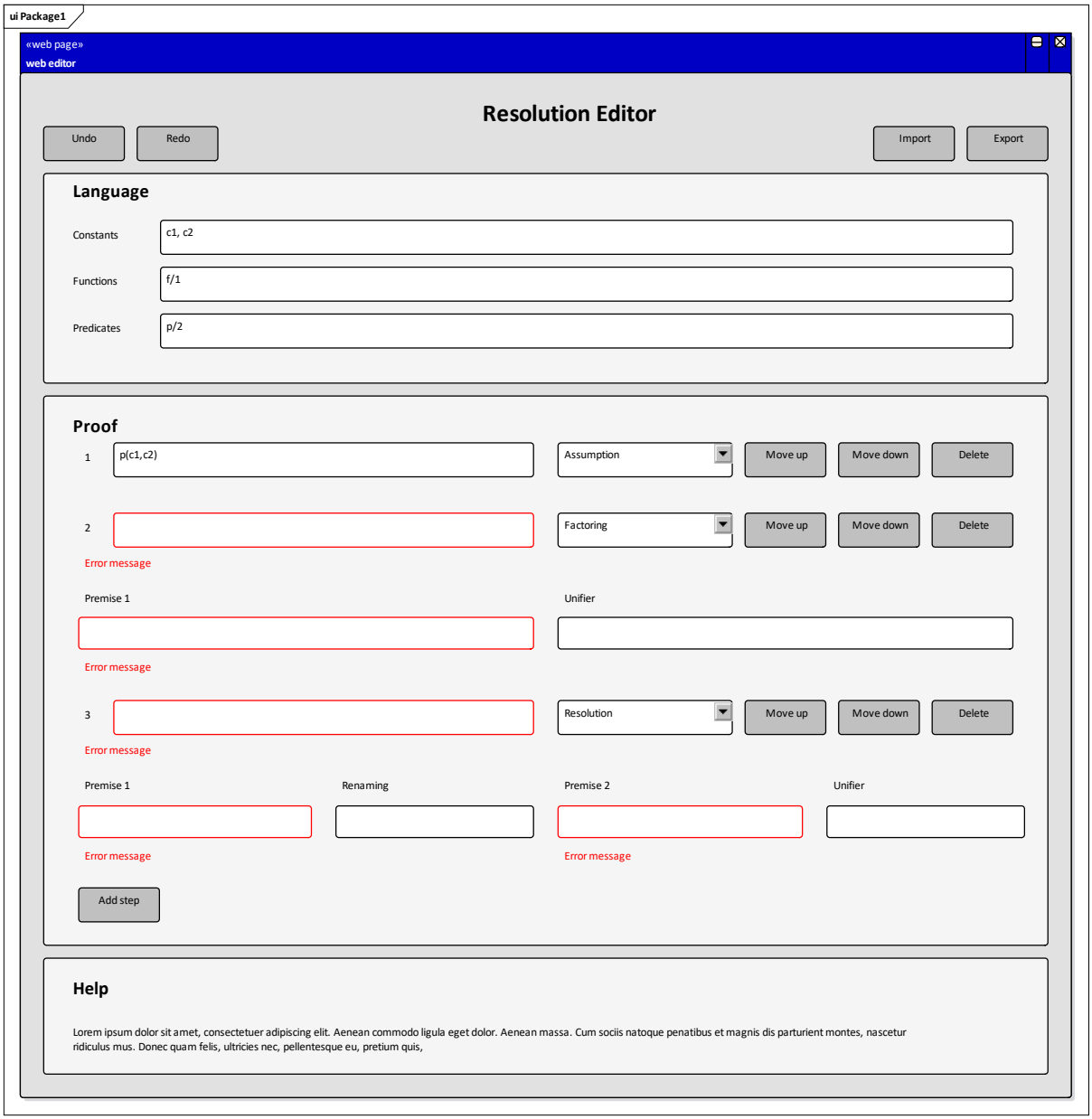
Obrázok 5 zobrazuje návrh komponentov a ich vzájomných väzieb v aplikácii. Komponent `Language` bude ukladať symboly konštánt, predikátové symboly a funkčné symboly. Komponent `Proof` môže obsahovať 0..n krokov, elementov `Step`. `Step` obsahuje práve jednu formulu, zloženú z 0..n literálov, predpis pravidla `Rule` a v závislosti od zvoleného pravidla elementy `Reference1`, `Renaming`, `Reference2` a `Unifier`.



Obrázok 5 Model komponentov

4.3 Návrh používateľského rozhrania

V súvislosti s navrhovanou štruktúrou je používateľské rozhranie (*Obrázok 6*) členené do 3 základných komponentov, a to: editor jazyka / Language, editor dôkazu / Proof a sekcia nápovedy / Help komponent. Editor dôkazu je doplnený o funkčné tlačidlo na pridanie nového kroku Add step a na vymazanie kroku je navrhnuté tlačidlo Delete. Preusporiadanie poradia krokov je navrhnuté tiež prostredníctvom tlačidiel Move up a Move Down, ktoré budú súčasťou každého jedného kroku. Samotné presúvanie krokov je možné len výmenou dvoch susediacich krokov. UndoRedo komponent a ImportExport komponent sú graficky vizualizované prostredníctvom funkčných tlačidiel Undo, Redo, Import a Export.



Obrázok 6 Návrh používateľského rozhrania

5 Implementácia editora rezolvenčných dôkazov

Editor na overenie správnosti rezolvenčných dôkazov je implementovaný ako JavaScript aplikácia s využitím knižníc React, Redux, Bootstrap a js-fol-parser [9]. Aplikácia je programovaná v platforme Microsoft Visual Studio a použitý bol editor kódu Visual Studio Code.

Ide o dvojvrstvovú aplikáciu, kde používateľská aj aplikačná vrstva bežia v prehliadači na strane klienta. Aplikácia sa síce dodáva cez webový server, ale ten neplní žiadnu funkciu počas jej behu.

Zdrojový kód je členený do nasledovných častí:

- `actions` - zoznam akcií, ktoré sú v aplikácii spracovávané;
- `components` - tu sú definované prvky používateľského rozhrania;
- `containers` - definované prepojenia akcií s komponentami a prepojenia stavu s vlastnosťami komponentov;
- `model` - definovanie štruktúr a pravidiel pre prácu s formulami a termami;
- `reducers` - tu sú uložené používané reduktory na prechody medzi stavmi na základe akcií.

5.1 Aplikačná logika

Aplikačná logika je oddelená od prezentačnej časti. Entity aplikačnej logiky sa používajú na reprezentáciu stavu rezolvenčného editora a na služby pre získavanie dát a reakcií na akcie používateľa. V nasledujúcich podkapitolách budú priblížené z pohľadu implementácie formuly a termy, stav aplikácie, akcie a reduktory.

5.1.1 Model pre rezolvenčný kalkul

Definovanie dátových štruktúr a pravidiel pre prácu s formulami a termami je umiestnené v priečinku `model`.

Formula je reprezentovaná ako trieda `Formula` a rozšírením tejto základnej triedy sú definované všetky ďalšie potrebné triedy pre prácu s formulami. Základná trieda Formula má definované metódy `toString()`, ktorá slúži na výpis formuly vo formáte reťazca, `equals()`, ktorá slúži na zistenie rovnosti typu a hodnoty objektov, vrátane vnorených

objektov, ak existujú, a `substitute()`, ktorá rieši aplikovanie substitúcie na podformuly a ich termy. Uvedené metódy môžu byť preťažené v triedach, ktoré sú rozšírením základnej triedy `Formula`, ako sú napr. triedy klauzula (`Clause`), či literál (`Literal`).

Obdobne je riešená aj práca s termami. Základná trieda `Term` je definovaná s atribútom `name`, ktorý reprezentuje názov termu a metódami `toString()`, `substitute()` a `equals()` s rovnakým významom, ako je uvedené pre formuly.

Aby sa v aplikácii ľahšie pracovalo s definovanými triedami, sú všetky triedy reexportované v súbore `index.js`, v priečinku `model`.

Operácie rezolvencie a faktorizácie sú implementované ako metódy `isResolventOf()` a `isFactorOf()` triedy `Clause`. V triede je tiež vytvorená pomocná multimnožina, ktorá ukladá literály a počet ich výskytov pre zefektívnenie algoritmu. Ten pozostáva z postupného generovania možných výsledných rezolventov po aplikovaní premenovania a následne unifikácie alebo faktorov po aplikovaní unifikácie. Postupným porovnávaním literálov sa odvodzujú nové výsledné klauzuly, ktoré sa porovnávajú s klauzulou zadanou používateľom. Využitie generátorov `getResolvents()` a `getFactors()` optimalizuje proces kontroly dôkazu.

5.1.2 Stav

Po spustení webového editora sa aplikácia nastavuje do iniciálneho stavu `initialCombinedState`. Stav je reprezentovaný ako štruktúrovaný objekt, ktorý pozostáva z 3 častí: jazyk `language`, dôkaz `steps` a pomocný atribút `inputChange`. Inak povedané, stav je zložený z 2 samostatných reduktorov `language` pre prácu s jazykom a `steps` pre prácu s krokmi dôkazu. Pole `inputChange` slúži na pomoc pri filtrovaní akcií, ktoré majú dopad na undo/redo operácie. `Language` je ďalej štruktúrovaný na objekty `const`, `funs` a `preds`, teda konštanty, funkcie a predikáty definované používateľom. `Steps` majú v sebe uložené informácie o poradí krokov, implementované ako pole `order`, mapu krokov `allSteps` s detailmi formúl, ktoré sú inštanciami vlastných objektov tried, pomocnú mapu umiestnenia krokov `rank`, ktorá uchováva vzťah medzi krokom a jeho poradím v dôkaze a posledné použité číslo kroku `id`, ktoré slúži na ich identifikáciu.

Štruktúra stavu vyzerá nasledovne:

```
state = {
    language: {
        consts: {
            input: "",
```

```
            object: new Set(),
            error: "",
            symbols: []
        },
        funs: {
            input: "",
            object: new Map(),
            error: "",
            symbols: []
        },
        preds: {
            input: "",
            object: new Map(),
            error: "",
            symbols: []
        }
    },
    steps: {
            order: [],
            allSteps: new Map(),
            rank: new Map(),
            id: 0
    },
    inputChange: {
        "originValue": ""
    }
};
```

Objekt nového prázdneho kroku, ktorý sa vkladá do mapy `allSteps`, vyzerá takto:

```
newStep = {
    formula: {
        input: "",
        object: undefined,
        error: ""
    },
    rule: "Assumption",
    reference1: {
        input: "",
        object: undefined,
        error: ""
    },
    renaming: {
        input: "",
        object: undefined,
        error: ""
    },
    reference2: {
        input: "",
        object: undefined,
        error: ""
    },
    unifier: {
        input: "",
        object: undefined,
        error: ""
    },
    valid: false
};
```

5.1.3 Akcie

Ďalšou časťou aplikačnej logiky, ako bolo uvedené v kap. 5, sú akcie. Umiestnené sú v priečinku `actions`. Pri práci s editorom jazyka sú implementované akcie pre editáciu množiny konštánt `CHANGE_CONST`, editáciu funkcií `CHANGE_FUN` a editáciu predikátov `CHANGE_PRED`.

Webový editor reaguje na akcie týkajúce sa pridania kroku na koniec `ADD_STEP` alebo medzi už existujúce kroky `INSERT_STEP`. Editáciu kroku reprezentuje akcia `CHANGE_STEP`. Pre krok dôkazu je možné vybrať typ z 3 možných hodnôt: Assumption, Resolution, Factoring (`CHANGE_RULE`). Samozrejmosťou sú akcie ako zmena unifikátora pri rezolvencii aj faktorizácii `CHANGE_UNIFIER`, premenovanie pri rezolvencii `CHANGE_RENAMING` a referencovanie sa na predchádzajúce kroky. Na zmenu číselných odkazov na premisy pravidla rezolvencie a faktorizácie reagujú akcie `CHANGE_REFERENCE1` a `CHANGE_REFERENCE2`. Vymazanie kroku zastrešuje akcia `DELETE_STEP` a na presúvanie poradia krokov sú určené akcie `STEP_UP` a `STEP_DOWN`.

Akcie `INPUT_FOCUS` a `INPUT_BLUR` slúžia ako pomocné akcie potrebné pre správne filtrovanie akcií a zaznamenanie stavu pri undo/redo operáciách a zmene hodnôt vstupných polí.

Komponenty používateľského rozhrania so stavom a akciami prepájajú na to špecializované komponenty nazývané kontajnery, ktoré vykresľujú stránku používateľovi.

5.1.4 Reduktory

Reduktor `language` sa stará o zaznamenávanie zmien konštánt, funkcií a predikátov a ich validáciu. Okrem reakcií na akcie `CHANGE_CONST, CHANGE_FUN, CHANGE_PRED` rieši aj odchytávanie syntaktických chýb a chýb v zmysle duplicitne zadaných symbolov pre konštanty, funkcie a predikáty.

Samotné zadávanie krokov dôkazu neznamená iba pridávanie nového kroku na koniec, a validáciu kroku ako takého, ale potrebné je riešiť súlad so zadanými symbolmi jazyka, prípadne prehodnotiť celý dôkaz, či jeho relevantnú časť. Reduktor `steps` reaguje na akcie `CHANGE_STEP`, `CHANGE_RULE`, `CHANGE_RENAMING`, `CHANGE_REFERENCE1`, `CHANGE_REFERENCE2`, `CHANGE_UNIFIER` a validuje syntax jednotlivých krokov dôkazu. Je previazaný aj s reduktorom `language`. Ak sa zmení jazyk, potrebné je prehodnotiť celý dôkaz. Po každej zmene kroku, ako aj poradia krokov nasleduje prehodnotenie krokov, ktoré

nasledujú za ním alebo s ním súvisia. Vyhodnocuje logické chyby a tým poskytuje spätnú väzbu používateľovi, ak klauzula nevyhovuje rezolvencii, či faktorizácii.

Pomocný reduktor `step` zabezpečuje iba zmenu hodnôt inputov. Pri identifikovaní novej hodnoty sa vždy vytvorí nová kópia objektov. Neprepisujú sa aktuálne objekty.

5.2 Používateľské rozhranie

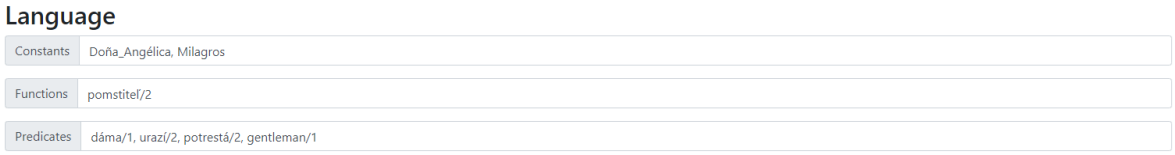
Prístup používateľa k aplikácii je prostredníctvom webovej stránky. Jednotlivé elementy, ktoré sa na stránke vyskytujú sú definované ako komponenty stránky a ich implementácia je umiestnená v priečinku `components`.

Na vzhľad jednotlivých komponentov v celej aplikácii sa používa knižnica Bootstrap, ktorá už má v sebe zakomponované návrhové šablóny založené na CSS štýloch. Prilinkovaním knižnice s využitím jej tried tak jednotlivé komponenty rozhrania získavajú preddefinované štýly.

Hlavný komponent, prezentovaný triedou `App`, vykresľuje jednotlivé zvyšné komponenty a má väzbu na všetky aplikačné kontajnery (`ActualLanguage`, `ActualProof`, `AddStep`, `ImportExport` a `UndoRedo`).

Komponenty sú rozdelené do logických celkov tak, aby vyhovovali procesu riešenia úloh na rezolvenčný kalkul pre logiku prvého rádu.

Komponent `Language` slúži pre zadanie symbolov jazyka, konštánt, funkcií a predikátov, ktoré používateľ zadáva ako postupne za sebou idúce symboly oddelené čiarkou. Ukážka možného vykreslenia komponentu `Language` je na *Obrázok 7*.



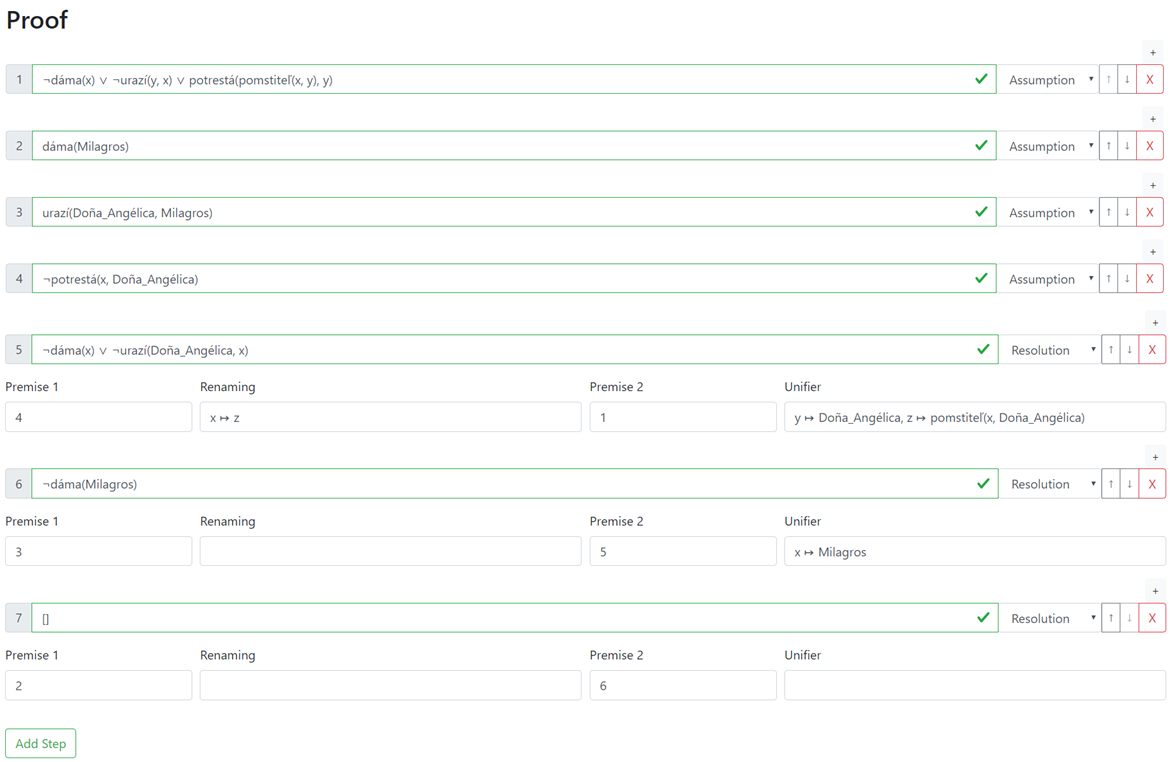
Obrázok 7 Komponent Language

Nasleduje sekcia pre zadávanie jednotlivých krokov dôkazu. Na vykreslenie rozhrania pre zadávanie krokov dôkazu sú určené komponenty `Proof` a `Step`. V `Proof` sa pre každý krok inicializujú požadované vlastnosti a komponent `Step` vykreslí daný krok tak, ako to bolo inicializované v komponente `Proof`. V závislosti od akcie používateľa, konkrétne od

zvoleného typu kroku: predpoklad/`Assumption`, rezolvencia/`Resolution` alebo faktorizácia/`Factoring` je potrebné vykresliť ďalšie komponenty na stránke.

Pre typ kroku rezolvencie sa použije komponent `Resolution` a pre krok faktorizácie komponent `Factoring`, ktoré vykreslia potrebné polia pre zadanie ďalších atribútov kroku. Komponent `Reference` vykreslí pole pre zadanie referencie, komponent `Renaming` zodpovedá prvku premenovania a komponent `Unifier` vykresľuje prvok unifikátor.

Ukážka vizualizácie zadaného dôkazu je na nasledujúcom obrázku (*Obrázok 8*).



Obrázok 8 Komponent Proof

`ErrorMsg` komponent slúži na zobrazenie chybovej hlášky. Len pre upresnenie, hodnoty chybových hlášok sú definované v stave a do stavu ich zapíšu reduktory.

Poslednou sekciou je komponent `Help`, ktorý používateľovi zobrazuje pomôcky a nápovedy pre prácu s editorom. Tento komponent je statický, kedže nie je prepojený so stavom a slúži len na vykreslenie textu.

5.3 Import a export

Definovaný jazyk a kroky dôkazu, či už vyriešený správny dôkaz, alebo rozpracované riešenie, teda ľubovoľný stav aplikácie je možné uložiť do lokálneho priečinka v počítači a opätovne nahrať späť do aplikácie. Stav aplikácie, dátový objekt `state`, vrátane definovaných objektov jazyka `language` a krokov `steps`, je uložený ako reťazec do súboru vo formáte JSON. Tento dátový formát bol zvolený preto, lebo je nezávislý od počítačovej platformy a plne vyhovuje na prenos dátových objektov, v našom prípade stavu, v čitateľnej forme pre používateľa. Pred uložením sa v stave niektoré zložitejšie štruktúry ako mapy, či množiny transformujú na polia a pri vkladaní stavu do aplikácie sa spätne vytvoria potrebné objekty, kvôli správnosti zachovania údajov pri formátovaní na reťazec.

Import a export je taktiež možný pri integrácií v iných aplikáciách. Implementácia sa odvíjala na základe spolupráce s Nikolajom Knihom a jeho bakalárskou prácou Interaktívny pracovný hárok pre výučbu logiky pre informatikov [14], ktorá sa zaoberá zjednotím aplikácií zameraných na výučbu matematickej logiky. Hárok ukladá dáta z aplikácií na GitHub a pri ďalšom otvorení rovnakého súboru sú nástroje s týmito dátami opäť načítané. Preto bolo potrebné klásť dôraz na jednoznačnú reprezentáciu dát a ich správnu validáciu pri opätovnom nahratí do aplikácie.

6 Testovanie

Testovanie je kľúčom k lepšej kvalite aplikácie. Dokáže identifikovať potencionálne chyby na základe ktorých je možné vylepšiť funkčnosť aplikácie, sprehľadniť prácu s editorom alebo zjednodušiť používateľské rozhranie.

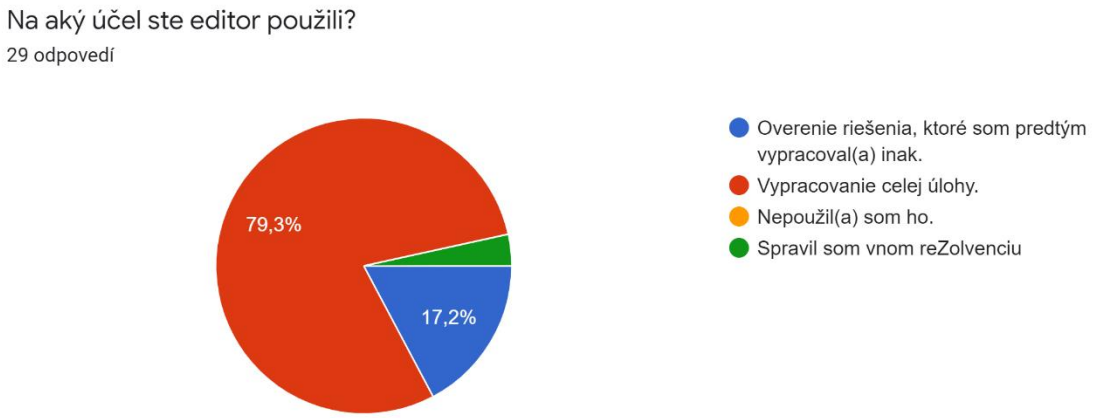
6.1 Spôsob testovania

Najlepšími testermi pre webový editor sú v tomto prípade práve študenti predmetu Matematika (4) - Logika pre informatikov, pre ktorých je stránka určená. Aplikácia bola sprístupnená pre študentov na cvičeniach a tí mali možnosť riešiť teoretické úlohy 12.3 a 12.4 práve prostredníctvom webového editora. Rovnako ho mohli použiť na riešenie domácej úlohy 12.5. Sada úloh sa nachádza v prílohe (Príloha A – Testovacie zadanie). Zároveň boli študenti požiadaní o vyplnenie krátkeho dotazníka (Príloha B – Dotazník) a jeho spätné zaslanie elektronickou formou.

6.2 Výsledky testovania

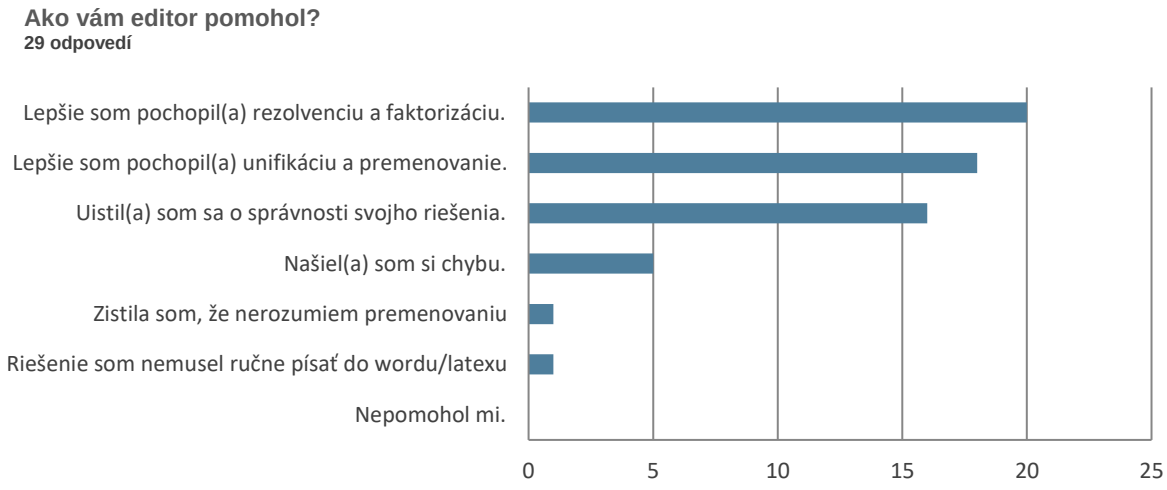
Úlohu odovzdalo 41 študentov, z toho 31 použilo editor a 29 sa k nemu vyjadrilo v dotazníku. Analýzou ich odpovedí je možné prezentovať nasledovné výstupy, kde každá odpoveď je doplnená o grafické vyhodnotenie.

Otázka – Na aký účel ste editor použili? Až 79,3% študentov, ktorí odovzdali dotazník ho použili na vypracovanie celej úlohy. Na overenie riešenia ho použilo 5 študentov a 1 študent ho využil iba na čiastkové kroky dôkazu. Týmto sa potvrdil predpoklad, že tvorba výučbových stránok má zmysel hlavne pri zložitejších typoch úloh, kde je potrebné priebežné overenie medzivýsledkov. Predpokladá sa, že študenti ktorí nepoužili editor, zvládli problematiku samostatne.



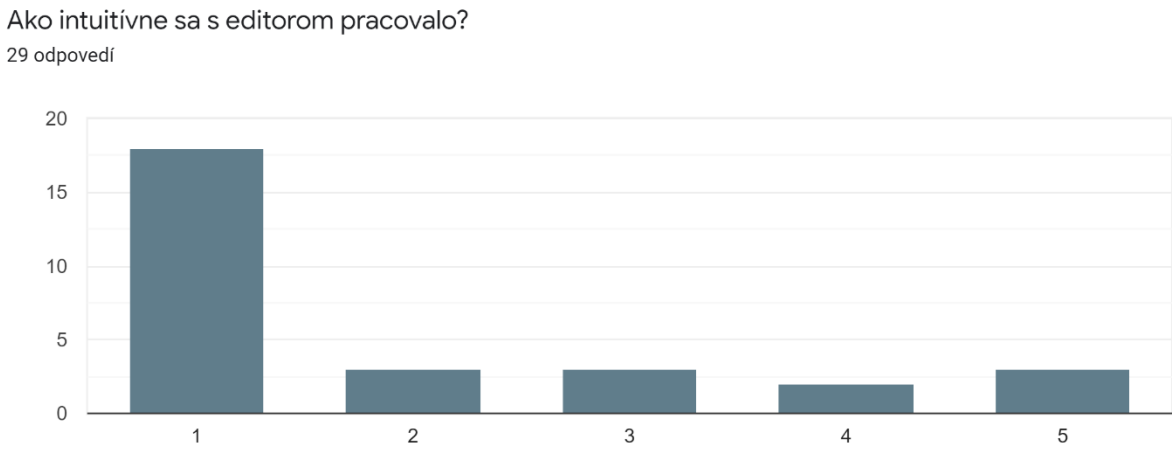
Obrázok 9 Na aký účel ste editor použili? [%]

Otázka – Ako vám editor pomohol? Editor študenti nielen použili, ale aj ocenili. Najčastejšou odpoveďou boli prínos pre pochopenie princípov rezolvencie a faktorizácie, čo uviedlo 20 študentov a unifikácie a premenovania, zastúpené 18 študentskými odpoveďami. Chybu vo vlastnom riešení príkladu si našlo 5 študentov a 1 študent priznal, že na základe editora zistil, že vlastne nerozumel správne premenovaniu. Je možné predpokladať, že editor splnil aj edukačnú funkciu.



Obrázok 10 Ako vám editor pomohol? [počet odpovedí]

Otázka – Ako intuitívne sa s editorom pracovalo? Priemer známok intuitívnosti bol zhruba 1,93, čo znamená že nástroj bol navrhnutý zrozumiteľne. 62 % študentov oznámkovalo prácu s editorom najvyššou známkou. Vo zvyšných 38% je možné nájsť priestor na zlepšenie.



Obrázok 11 Ako intuitívne sa s editorom pracovalo? [počet odpovedí]

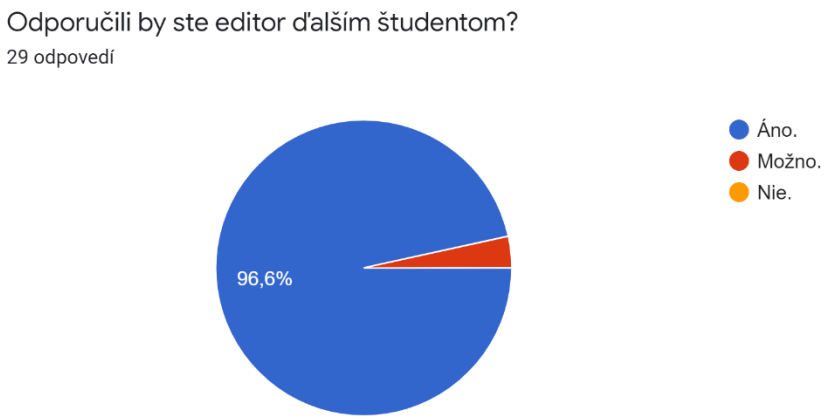
Otázka - Čo sa vám na editore páčilo? Študenti na editore ocenili najmä nasledovné aspekty:

- jednoduché a intuitívne ovládanie,
- samotnú kontrolu syntaktických a logických chýb,
- presúvanie a pridávanie krokov, ktoré neznehodnotia už vytvorený dôkaz,
- pomoc pri pochopení problematiky,
- undo/redo operácie.

Otázka - Čo sa vám na editore nepáčilo, chýbalo, čo by ste zmenili? Ako návrhy na vylepšenie a prípadne pripomienky k chybám uviedli respondenti prevažne tieto odvetvia:

- skompaktniť zobrazovanie dôkazu,
- možnosť exportovať do verzie pdf pre tlač,
- nefunkčnosť v prehliadači Microsoft Edge,
- zabezpečenie zachovania stavu pri obnovení stránky v prehladači,
- spísať detailnejšie nápovedy a chybové hlásenia.

Otázka – Odporučili by ste editor ďalším študentom? 28 študentov by odporučilo aplikáciu ďalším študentom. 1 študent váhal a nikto sa nenašiel, kto by editor neodporučil ďalej.



Obrázok 12 Odporučili by ste editor ďalším študentom? [%]

Na základe odpovedí, prevažne pozitívnych, možno usudzovať, že editor rezolvenčných dôkazov je pre študentov užitočný. Umožnil im lepšie pochopiť preberanú tému, bol dobre použiteľný a študenti hlavne ocenili poskytované možnosti úpravy štruktúry dôkazu. Na základe pripomienok študentov bolo zoptimalizované rozloženie komponentov editora tak, aby bol dôkaz kompaktnejší a boli dopracované detailnejšie nápovedy.

Záver

Výsledkom bakalárskej práce je funkčný webový editor, ktorý ponúka študentom nástroj na precvičovanie pravidiel rezolvencie a faktorizácie v logike prvého rádu. Nástroj má reálne využitie a to pri riešení domácich úloh študentmi alebo na cvičeniach pri objasňovaní preberanej látky. Aplikácia celkovo dosiahla zväčša pozitívne hodnotenie, čo naznačujú nie len odpovede z testovania, ale aj percentuálny pomer odovzdaných úloh v elektronickej podobe riešených v editore, voči iným formám. Na základe spolupráce bola aplikácia taktiež integrovaná do združujúceho hárku, čo zjednoduší prístup k editoru.

Pri spracovaní témy bakalárskej práce, implementácie webového editora, boli aplikované všetky fázy životného cyklu informačného systému. Na základe logickej nadväznosti vývojových fáz, cez analýzu problematiky logiky prvého rádu a dostupných webových technológií, zber požiadaviek, návrh systému a používateľského rozhrania, implementáciu aplikačnej logiky, testovania a nasadenia editora do prevádzky, bol dosiahnutý cieľ práce. Webový editor bol implementovaný ako JavaScript aplikácia s využitím knižníc React, Redux, Bootstrap a externého parsera pre logiku prvého rádu. Používateľské rozhranie je riešené cez jednu samostatnú webovú stránku, výhodou čoho je jednoduchá orientácia a prehľadnosť riešenia. Výsledkom je funkčná, stabilná a praktická aplikácia.

Aplikáciu je možné ďalej rozvíjať vo viacerých oblastiach. Jednou z nich je napríklad dopracovanie tvorby klauzálnej formy, ktorú si aktuálne musí používateľ zrealizovať sám mimo editora. Ďalej by mohla sekcia s nápovedami obsahovať detailnejší popis ako funguje rezolvencia a tiež chybové hlášky by mohli presnejšie zobrazovať prečo nie je možné konkrétny chybný krok rezolvencie vykonať. Ďalší priestor na zlepšenie je možný vo vylepšení používateľského rozhrania. Dĺžka webovej stránky pri zložitých zadaniach narastá a je vhodné sa zamyslieť nad jej rozčlenením do menších celkov. Rozhranie je dostupné iba v anglickom jazyku, čo tiež nemusí vyhovovať všetkým študentom.

Literatúra

[1] Ján Kľuka and Jozef Šiška. Prednášky z matematiky (4) - logiky pre informatikov., 2019. [Online; 13.05.2019]

[2] Michael Genesereth and Eric Kao. Introduction to Logic. Morgan & Claypool, 2013. ISBN 9781627052

[3] Vladimír Kvasnička. Úvod do logiky pre informatikov, 2012. ISBN 978-1108401296.

[4] Nyitraiová, A.: Educational tools for first order logic. Bakalárska práca – Univerzita Komenského, Bratislava, 2018. https://fmfi-uk-1-ain-412.github.io/tableauEditor/

[5] Onódy, Z.: A proof assistant for first-order logic. Bakalárska práca – Univerzita Komenského, Bratislava, 2018. https://fmfi-uk-1-ain-412.github.io/proof-assistant/

[6] Facebook Inc, React: A JavaScript library for building user interfaces, https://reactjs.org/ [Online; 6.2.2020]

[7] Bootstrap team, Bootstrap: Framework for building responsive websites, https://getbootstrap.com/ [Online; 6.2.2020]

[8] Dan Abramov and the Redux documentation authors, Redux: A Predictable State Container for JS Apps, https://redux.js.org/ [Online; 6.2.2020]

[9] Ján Kľuka, Parsers for first-order languages, https://github.com/FMFI-UK-1-AIN-412/js-fol-parser [Online; 6.2.2020]

[10] Visual Studio: Best-in-class tools for any developer, https://visualstudio.microsoft.com/cs/ [Online; 6.2.2020]

[11] Stanford University: Robinson, http://intrologic.stanford.edu/logica/homepage/robinson.php [Online; 6.2.2020]

[12] Cifra, M.: Prieskumník sémantiky logiky prvého rádu. Bakalárska práca – Univerzita Komenského, Bratislava, 2018. https://fmfi-uk-1-ain-412.github.io/structure-explorer/

[13] J.A. Robinson (Jan 1965). "A Machine-Oriented Logic Based on the Resolution Principle". Journal of the ACM. 12 (1): 23–41. doi:10.1145/321250.321253

[14] Kniha, N.: Interaktívny pracovný hárok pre výučbu logiky pre informatikov. Bakalárska práca – Univerzita Komenského, Bratislava, 2020.

[15] Preeti Jaiswal, React Redux: A Complete Guide to Beginners, https://www.cronj.com/blog/react-redux-a-complete-guide-to-beginners/ [Online; 31.5.2020]

Príloha A – Testovacie zadanie

Matematika 4 — Logika pre informatikov
Teoretická úloha 12

Riešenie hodnotenej a prémiovej časti tejto úlohy **odovzdajte** najneskôr v pondelok **25. mája 2020 o 12:20** cez odovzdávací formulár pre tu12[1].

Odovzdávajte URL odkaz na

- **jeden PDF dokument s právom na komentovanie** nahratý na Google Drive; dokument musí obsahovať celé riešenie vrátane rezolvenčného dôkazu;
- **export z editora rezolvenčných dôkazov**[4], ak ho použijete pri riešení; čitateľný dôkaz sa musí nachádzať aj v PDF, aby sme ho mohli komentovať.

Neodovzdávajte: priečinky; dokumenty s riešeniami viacerých úloh.

Odovzdané riešenia musia byť **čitateľné** a mať primerane **malý** rozsah. Na riešenia všetkých úloh sa vzťahujú všeobecné **pravidlá**[2].

Čísla úloh v zátvorkách odkazujú do zbierky[3], kde nájdete riešené príklady a ďalšie úlohy na precvičovanie.

Riešenia niektorých úloh môžete skontrolovať pomocou editora rezolvenčných dôkazov[4].

Ak nie je uvedené inak, v každom použitom jazyku $\mathcal{L}$ logiky prvého rádu predpokladáme množinu indivíduových premenných $\mathcal{V}_{\mathcal{L}} = \{k, l, m, \ldots, x, y, z, k_1, l_1\ m_1, \ldots, x_1, y_1, z_1, k_2, l_1\ m_2, \ldots\}$.

[1] https://forms.gle/PUqqH4CFnD6Z3JnG8
[2] https://dai.fmph.uniba.sk/w/Course:Mathematics_4/sk#pravidla-uloh
[3] https://github.com/FMFI-UK-1-AIN-412/lpi/blob/master/teoreticke/zbierka.pdf
[4] https://norbertju.github.io/ResolutionEditor/

Cvičenie 12.1. (7.7.3) Zistite, či sú nasledujúce dvojice postupností symbolov unifikovateľné, a nájdite ich najvšeobecnejší unifikátor.

a) Arabela — prvý_majiteľ(x)

b) kupujúci(Kolobežka6259, y) — kupujúci(t, prvý_majiteľ(t))

c) predaj(x, prvý_majiteľ(t), t, p) — predaj(x, y, Kolobežka6259, 35eur)

d) predaj(u, u, w, r) — predaj(kupujúci(y, t), y, t, p)

e) predaj(x, Ingrid, t, cena(t)) — predaj(kupujúci(y, t), y, t, p)

1

Cvičenie 12.2. (7.7.4) Sfaktorizujte klauzuly:

a) $\neg\text{dáma}(x) \vee \text{urazil}(y, x) \vee \neg\text{dáma}(\text{Milagros})$

b) $\neg\text{chráni}(\text{osobný\_strážca}(x), x) \vee \neg\text{chráni}(x, y)$

Cvičenie 12.3. (7.7.5) V rezolvenčnom kalkule dokážte nesplniteľnosť množín klauzúl:

a) $T = \{(\text{šteká}(x) \vee \neg\text{pes}(x)),\ (\neg\text{pes}(x) \vee \text{hryzie}(x)),$
$(\neg\text{pes}(x) \vee \neg\text{šteká}(x) \vee \neg\text{hryzie}(x)),\ \text{pes}(\text{Dunčo})\}$

b) $T = \{(\text{dom}(x) \vee \text{strom}(y) \vee \text{pri}(x, y)),$
$(\text{strom}(y) \vee \neg\text{pri}(x, y)),\ (\neg\text{dom}(x) \vee \neg\text{strom}(y))\}$

c) $T = \{(\text{c}(x, y) \vee \text{b}(x)),\ (\neg\text{c}(x, \text{L}) \vee \text{a}(\text{L})),\ (\text{c}(\text{P}, y) \vee \neg\text{b}(\text{P})),\ (\neg\text{a}(y) \vee \neg\text{c}(x, y))\}$

Cvičenie 12.4. (7.7.9) Uvažujme nasledovné tvrdenia a ich formalizáciu v jazyku logiky prvého rádu *bez rovnosti* $\mathcal{L}$, kde $\mathcal{C}_{\mathcal{L}} = \{\text{Hanka}\}$, $\mathcal{F}_{\mathcal{L}} = \emptyset$ a $\mathcal{P}_{\mathcal{L}} = \{\text{autíčko}^1, \text{bábika}^1, \text{červené}^1, \text{dievčenské}^1, \text{hračka}^1, \text{hračkárstvo}^1, \text{chlapčenské}^1, \text{matfyzáčka}^1, \text{P}^1, \text{šaty}^1, \text{mama}^2, \text{má}^2, \text{zakúpené\_v}^2, \text{kúpi}^3\}$:

(A_1) Autíčka sú chlapčenské hračky a bábiky sú dievčenské hračky.

$$\forall x(\text{autíčko}(x) \to \text{chlapčenské}(x) \wedge \text{hračka}(x)) \wedge$$
$$\forall x(\text{bábika}(x) \to \text{dievčenské}(x) \wedge \text{hračka}(x))$$

(A_2) Hanka má dve autíčka.

$$\exists x\, \exists y(\text{P}(x) \wedge \neg\text{P}(y) \wedge \text{má}(\text{Hanka}, x) \wedge \text{autíčko}(x) \wedge \text{má}(\text{Hanka}, y) \wedge \text{autíčko}(y))$$

(A_3) Každá hračka bola zakúpená v hračkárstve.

$$\forall x(\text{hračka}(x) \to \exists y\,(\text{zakúpené\_v}(x, y) \wedge \text{hračkárstvo}(y)))$$

(A_4) Každé dievča má aspoň jednu dievčenskú hračku.

$$\forall x(\text{dievča}(x) \to \exists y(\text{má}(x, y) \wedge \text{dievčenské}(y) \wedge \text{hračka}(y)))$$

(A_5) Hanka je dievča, ktoré má bábiku, ktorá má červené šaty.

$$(\text{dievča}(\text{Hanka}) \wedge$$
$$\exists x(\text{má}(\text{Hanka}, x) \wedge \text{bábika}(x) \wedge \exists y(\text{má}(x, y) \wedge \text{červené}(y) \wedge \text{šaty}(y))))$$

(A_6) Každá mama kúpi svojmu dieťaťu nejakú hračku.

$$\forall x\, \forall y(\mathsf{mama}(x, y) \to \exists z(\mathsf{hračka}(z) \wedge \mathsf{kúpi}(x, y, z)))$$

(A_7) Dievčatá, ktoré majú nejakú chlapčenskú hračku, sa stanú matfyzáčkami.

$$\forall x(\mathsf{dievča}(x) \to (\exists y(\mathsf{hračka}(y) \wedge \mathsf{chlapčenské}(y)) \to \mathsf{matfyzáčka}(x)))$$

Zistite pomocou rezolvencie, či sa Hanka stane matfyzáčkou, teda, či z teórie $T = \{A_1, \dots, A_7\}$ vyplýva formula:

$$\mathsf{matfyzáčka}(\mathsf{Hanka})$$

Hodnotená časť

Úloha 12.5. (7.7.11) Uvažujme nasledujúce tvrdenia:

(V_1) Každý vták spí na nejakom strome.

(V_2) Potápky sú vtáky a sú tiež vodnými živočíchmi.

(V_3) Strom, na ktorom spí nejaký vodný vták, sa nachádza blízko jazera.

(V_4) Všetko, čo spí na niečom, čo sa nachádza blízko nejakého jazera, sa živí rybami.

Vyriešte nasledujúce úlohy:

a) Sformalizujte tvrdenia ako teóriu $T = \{V_1, \dots, V_4\}$ vo vhodnom jazyku logiky prvého rádu.

   Zvoľte predikátové a funkčné symboly podľa potreby tak, aby formalizácia dávala zmysel, teda aby sformalizované pojmy neboli izolované a formalizácia bola splniteľná.

b) Upravte teóriu T na ekvisplniteľnú klauzálnu teóriu T'.

c) Pre nasledujúcu otázku sformulujte príslušný logický problém a zodpovedzte problém aj otázku pomocou rezolvencie pre logiku prvého rádu:

   Je na základe tvrdení V_1–V_4 pravda, že každá potápka sa živí rybami?

Prémiová časť

Prémiová úloha 12.6. (1 bod, 4.3.1) Dokážte alebo vyvráťte:

a) Nech $\mathcal{L}$ je jazyk logiky prvého rádu bez funkčných symbolov ($\mathcal{F}_{\mathcal{L}} = \emptyset$). Jazyk $\mathcal{L}_1$ vytvoríme z $\mathcal{L}$ pridaním novej indivíduovej konštanty c a zmenou všetkých predikátových symbolov na funkčné, teda: $\mathcal{C}_{\mathcal{L}_1} = \mathcal{C}_{\mathcal{L}} \cup \{c\}$, $\mathcal{P}_{\mathcal{L}_1} = \emptyset$, $\mathcal{F}_{\mathcal{L}_1} = \mathcal{P}_{\mathcal{L}}$, $\mathcal{V}_{\mathcal{L}_1} = \mathcal{V}_{\mathcal{L}}$.

   Nech A je formula v jazyku $\mathcal{L}$. Formulu B v jazyku $\mathcal{L}_1$ vytvoríme tak, že každý predikátový atóm $P(a_1, a_2, \dots, a_n)$ v A nahradíme rovnostným atómom $P(a_1, a_2, \dots, a_n) \doteq c$. Potom platí:

   A je pravdivá v nejakej štruktúre pre $\mathcal{L}$ s aspoň dvojprvkovou doménou, vtt B je pravdivá v nejakej štruktúre pre $\mathcal{L}_1$ s aspoň dvojprvkovou doménou.

b) Ak vo výrokovologickej tautológii nahradíme všetky atómy prvorádovými formulami (tak, že za ten istý atóm vždy dosadíme tú istú formulu), dostaneme platnú prvorádovú formulu.

Prémiová úloha 12.7. (0,5 bodu, 4.3.4) Zadefinujte vzťah *z teórie T vyplýva formula X* ($T \models_{\mathrm{p}} X$) a pojem *nesplniteľná formula* vo výrokovologickej časti logiky prvého rádu.

Dokážte alebo vyvráťte: Nech S je množina výrokovologických formúl a nech X je výrokovologická formula. Ak X je nesplniteľná a $S \models_{\mathrm{p}} X$, tak S je nesplniteľná.

Príloha B – Dotazník

Rezolvenčný editor

Volám sa Norbert Jurík a som študentom FMFI. Implementácia rezolvenčného editora je predmetom mojej bakalárskej práce. Chcel by som vás poprosiť o vyplnenie krátkeho dotazníka. Vaša spätná väzba mi pomôže pri posudzovaní užitočnosti, hľadaní potencionálnych chýb alebo vylepšení samotného nástroja.
https://norbertju.github.io/ResolutionEditor/
Vopred ďakujem za odpovede :)

* Povinné

Na aký účel ste editor použili? *

- ◯ Overenie riešenia, ktoré som predtým vypracoval(a) inak.
- ◯ Vypracovanie celej úlohy.
- ◯ Nepoužil(a) som ho.
- ◯ Iné: _____

Ako vám editor pomohol? *

- ☐ Našiel(a) som si chybu.
- ☐ Uistil(a) som sa o správnosti svojho riešenia.
- ☐ Lepšie som pochopil(a) rezolvenciu a faktorizáciu.
- ☐ Lepšie som pochopil(a) unifikáciu a premenovanie.
- ☐ Nepomohol mi.
- ☐ Iné: _____

Ako intuitívne sa s editorom pracovalo? *

	1	2	3	4	5	
Dobre	○	○	○	○	○	Zle

Čo sa vám na editore páčilo?

Vaša odpoveď

Čo sa vám na editore nepáčilo, chýbalo, čo by ste zmenili?

Vaša odpoveď

Odporučili by ste editor ďalším študentom? *

- ○ Áno.
- ○ Možno.
- ○ Nie.

Príloha C – Elektronická príloha

Zdrojový kód rezolvenčného editora tvorí elektronickú prílohu tejto práce. Taktiež je dostupný online na adrese https://github.com/NorbertJu/ResolutionEditor. Skompilovaná funkčná verzia aplikácie sa nachádza na adrese https://norbertju.github.io/ResolutionEditor/.

Výstupné dáta z dotazníka sú uložené v súbore `dotaznik_vystup.xlsx` ako súčasť elektronickej prílohy.