Automatski dokazivači teorema

Živković, Alen

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Alen Živković

AUTOMATSKI DOKAZIVAČI TEOREMA

Diplomski rad

Voditelj rada:
prof. dr. sc. Mladen Vuković

Zagreb, rujan, 2021.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

U današnjem svijetu digitalne revolucije računala ulaze u sve faze svakodnevnog života pa nije neobično pitati se kako ona mogu doprinijeti na području matematike. Iako se u očima javnosti računarstvo i matematika često svrstavaju u isti koš, sami matematičari nisu tako otvoreni prema prodoru računala u njihovo područje. Glavni je argument da računala nikad neće moći imati ljudsku domišljatost u najvažnijem dijelu matematičarevog posla - dokazivanju tvrdnji. Taj je argument danas teško opovrgnuti, no razvoj automatskih dokazivača teorema pokazuje da za računalom potpomognutu matematiku još ima nade.

Automatski dokazivači teorema su programi čije ime savršeno objašnjava njihovu svrhu, a to je dokazati teorem koji im je dan kao ulaz. Dosad su ovakvi programi bili limitirani, no neki od njih, poput programa *Agda* ili *Coq*, pokazali su se vrlo efikasnim u nekim situacijama. Napomenimo da literatura na engleskom jeziku često koristi naziv "proof assistant" pa umjesto o automatskim dokazivačima možemo govoriti i o dokazima potpomognutim računalom.

Ovaj se diplomski rad ne bavi općenito automatskim dokazivačima teorema, već je posvećen jednom primjeru programa koji se naziva HERBY. U ovom ćemo radu prvo proći kroz teorijske osnove ovog programa, proučiti način na koji se program HERBY koristi dobivenim rezultatima i na kraju dokazati dva teorema koristeći ga.

U prvom poglavlju uvodimo osnovne pojmove kao što su Herbrandova baza i Herbrandov univerzum te navodimo Herbrandov teorem koji je ključ funkcioniranja programa HERBY. Vidjet ćemo da program HERBY zapravo dokazuje da skup svih pretpostavki zajedno s negiranim zaključkom nije ispunjiv. Možemo reći da program HERBY imitira jedan dobro poznati "ljudski" način dokazivanja: pretpostavi suprotno pa pokušava doći do kontradikcije.

U drugom poglavlju promatramo reprezentaciju logike prvog reda unutar programa HERBY. Zatim, bavimo se transformacijom formula logike prvog reda u skup klauzula što je traženi oblik za ulaz programa. Na kraju ovog poglavlja razmatramo nekoliko heuristika koje se koriste kako bi se što efikasnije došlo do željenog dokaza.

U trećem poglavlju pokrećemo program HERBY i dokazujemo dva teorema. Prvi primjer koristi Peanove aksiome kako bi dokazao jednostavnu tvrdnju o nejednakosti prirodnih brojeva. U drugom primjeru dokazujemo teorem iz teorije grupa. Proći ćemo kroz ispis

programa HERBY te pratiti postupak dokazivanja.

Poglavlje 1

Teorijska osnova

U ovom ćemo poglavlju uvesti osnovne teorijske pojmove koji su nam potrebni kako bi razumjeli funkcioniranje programa HERBY. Uvodimo pojmove Herbrandov univerzum, Herbrandova baza i semantička stabla. Nakon otoga navodimo tri verzije Herbrandovog teorema. To su rezultati koji će nam dati jasan uvid u način na koji program HERBY dokazuje teoreme. Za naše izlaganje najviše ćemo koristiti knjige [1] i [4].

1.1 Herbrandov univerzum i Herbrandova baza

Koristit ćemo alfabet logike prvog reda koji se sastoji od prebrojivo mnogo konstantskih simbola, prebrojivo mnogo funkcijskih simbola i prebrojivo mnogo relacijskih simbola. U nastavku ćemo relacijske simbole označavati velikim slovima $P, Q, R, \ldots$. Funkcijske simbole označavat ćemo s $f, g, h, \ldots$, a individualne varijable s $x, y, z, \ldots$. Konstantski simboli biti će označeni malim slovima $a, b, c, \ldots$

Kako bismo definirali Herbrandov univerzum prvo nam je potreban pojam klauzule. *Literal* je atomarna formula ili negacija atomarne formule. *Klauzula* je skup literala koji označava njihovu disjunkciju. Za sve varijable koje se pojavljuju u klauzuli pretpostavljamo da su implicitno univerzalno kvantificirane. Dakle, $\{P(x), \neg Q(x)\}$ je klauzula koja označava formulu $(\forall x)(P(x) \vee \neg Q(x))$.

Definicija 1.1.1. *Neka je S skup klauzula, $\mathcal{A}$ skup svih konstantskih simbola koji se pojavljuju u S te $\mathcal{F}$ skup svih funkcijskih simbola koji se pojavljuju u S.* Herbrandov univerzum od S*, u oznaci H_S, se definira rekurzivno ovako:*

- *za svaki $a_i \in \mathcal{A}$ imamo $a_i \in H_S$*
- *za sve $n \in \mathbb{N}$, $f_i^n \in \mathcal{F}$, $t_1, \ldots t_n \in H_S$ vrijedi $f_i^n(t_1, \ldots, t_n) \in H_S$.*

Ako skup klauzula S ne sadrži konstantski simbol tada po dogovoru dodajemo u Herbrandov univerzum H_S konstantski simbol kojeg označavamo sa a.

Napomena 1.1.2. *Ako skup klauzula S ne sadrži ni funkcijske ni konstantske simbole, tada će pripadni Herbrandov univerzum biti jednočlan skup: $H_S = \{a\}$.*

Skupovi klauzula koje će program HERBY primati kao ulaz bit će svi konačni. Iz tog razloga pretpostavimo da je S neki konačan skup klauzula. Ako S ne sadrži niti jedan funkcijski simbol tada je Herbrandov univerzum H_S konačan skup, a ako S sadrži funkcijski simbol tada je H_S prebrojivo beskonačan skup.

Primjer 1.1.3. *Navodimo nekoliko primjera Herbrandovog univerzuma za zadane skupove klauzula:*

$$\begin{aligned} S_1 &= \{\{P(a), P(b), \neg Q(a)\}, \{\neg P(x), Q(b)\}\} \\ H_{S_1} &= \{a, b\} \end{aligned} \tag{1.1}$$

$$\begin{aligned} S_2 &= \{\{P(f(x), x), Q(y)\}, \{\neg P(f(y), g(y))\}\} \\ H_{S_2} &= \{a, f(a), g(a), f(f(a)), g(f(a)), f(g(a)), g(g(a)), \dots\} \end{aligned} \tag{1.2}$$

$$\begin{aligned} S_3 &= \{\{P(x, f(x, y)), \neg P(a, g(x, y))\}, \{\neg Q(g(x, y), b)\}\} \\ H_{S_3} &= \{a, b, f(a, a), g(a, a), f(a, b), g(a, b), f(b, a), g(b, a), \dots\} \end{aligned} \tag{1.3}$$

U (1.1) imamo primjer konačnog skupa klauzula S_1 bez funckijskih simbola. Vidimo da je onda i H_{S_1} konačan. U primjeru (1.2) vidimo konstrukciju Herbrandovog univerzuma kada pripadni skup klauzula S_2 ne sadrži konstantski simbol. Po dogovoru smo u H_{S_2} dodali novi konstantski simbol a. Primjer (1.3) je standardan primjer Herbrandovoog univerzuma od S_3 koji sadrži i konstantske i funkcijske simbole.

Definirajmo sada pojam *zatvorenih terma* kako bi dobili još jednu ekvivalentnu definiciju Herbrandovog univerzuma.

Definicija 1.1.4.

- Zatvoreni term *je term koji ne sadrži niti jednu varijablu.*
- Zatvorena atomarna formula *je atomarna formula čiji su svi termi zatvoreni.*
- Zatvoreni literal *je ili zatvorena atomarna formula ili negacija zatvorene atomarne formule.*

Za neki skup klauzula S će po definiciji H_S biti skup svih pripadnih zatvorenih terma. Dakle, Herbrandov univerzum od S je skup svih zatvorenih terma koje je moguće dobiti pomoću funkcijskih i konstantskih simbola koji se pojavljuju u S.

Definicija 1.1.5. *Neka je S neki skup klauzula i H_S Herbrandov univerzum od S.* Herbrandova baza od S*, u oznaci B_S, je skup zatvorenih atomarnih formula koje se mogu dobiti koristeći relacijske simbole koji se pojavljuju u S i terme iz H_S.*

Primjer 1.1.6. *Promotrimo Herbrandove baze za skupove klauzula iz primjera 1.1.3.*

$$B_{S_1} = \{P(a), Q(a), P(b), Q(b)\}$$

$$\begin{aligned} B_{S_2} =\{&P(a), Q(a), P(f(a)), Q(f(a)), P(g(a)), Q(g(a)), P(f(f(a))), Q(f(f(a))),\\ &P(g(f(a))), Q(g(f(a))), P(f(g(a))), Q(f(g(a))), P(g(g(a))), Q(g(g(a))), \dots\} \end{aligned}$$

$$\begin{aligned} B_{S_3} =\{&P(a), Q(a), P(b), Q(b), P(f(a,a)), Q(f(a,a)), P(g(a,a)), Q(g(a,a)),\\ &P(f(a,b)), Q(f(a,b)), P(g(a,b)), Q(g(a,b)), P(f(b,a)), Q(f(b,a)),\\ &P(g(b,a)), Q(g(b,a)), \dots\} \end{aligned}$$

Kako za rad programa HERBY promatramo konačne skupove klauzula S, Herbrandova baza B_S će opet biti ili konačan skup ili prebrojiv beskonačan skup. Kad su zatvorene atomarne formule poredane u odnosu na točno definirani uređaj kažemo da su u kanonskom uređaju. Kažemo da je Herbrandova baza zapisana u kanonskom uređaju ako su zatvorene atomarne formule u kanonskom uređaju.

Primjer 1.1.7. *Pogledajmo primjere kanonskih Herbrandovih baza za dane skupove klauzula:*

$$\begin{aligned} S_1 &= \{\{P(x)\}, \{\neg P(a), Q(x)\}, \{\neg Q(f(x))\}\}\\ B_{S_1} &= \{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), \dots\} \end{aligned}$$

$$\begin{aligned} S_2 &= \{\{P(h(x,y), x), Q(x,y)\}, \{\neg P(x,y), Q(y,x)\}, \{\neg Q(x,y)\}\}\\ B_{S_2} &= \{P(a,a), Q(a,a), P(a,h(a,a)), Q(a,h(a,a)), P(h(a,a),a), Q(h(a,a),a), \dots\} \end{aligned}$$

Želimo li dobiti ekvivalentnu definiciju Herbrandove baze moramo definirati pojam zatvorene instance, a samim time i pojam supstitucije.

Definicija 1.1.8. Supstitucija *varijabli termima je skup $\{x_1 \longleftarrow t_1, \dots, x_n \longleftarrow t_n\}$, gdje je svaki x_i varijabla, a svaki t_i term koji je različit od x_i. Prazna substitucija je prazni skup.*

Neka je E formula i neka je $\theta = \{x_1 \longleftarrow t_1, \dots, x_n \longleftarrow t_n\}$ supstitucija. Instanca $E\theta$ od E je dobivena simultanom supstitucijom svakog pojavljivanja od x_i u E s t_i.

Definicija 1.1.9. *Neka je S skup klauzula, H_S njegov Herbrandov univerzum i $C \in S$.* Zatvorena instanca *klauzule C je svaka klauzula dobivena supstitucijom svake slobodne varijable iz C elementima Herbrandovog univerzuma H_S.*

Primjer 1.1.10. *Pogledajmo sve zatvorene instance klauzule* $\{\neg P(x), Q(b)\} \in S_1$ *iz 1.1. Budući da je Herbrandov univerzum* $H_{S_1} = \{a, b\}$ *imamo dvije zatvorene instance:* $\{\neg P(a), Q(b)\}$ *i* $\{\neg P(b), Q(b)\}$ *gdje su pripadne supstitucije dane sa* $\theta_1 = \{x \longleftarrow a\}$ *te* $\theta_2 = \{x \longleftarrow b\}$*. Ako početnu klauzulu označimo s* C *i promatramo kao elementarnu disjunkciju, zatvorene se instance mogu označiti redom s* $C\theta_1$ *i* $C\theta_2$*.*

Sada možemo reći da je Herbrandova baza nekog skupa klauzula S skup svih zatvorenih atomarnih formula koje se pojavljuju (negirane ili nenegirane) kao literali u svim zatvorenim instancama svih klauzula iz S.

Interpretacija Herbrandove baze B_S nekog skupa klauzula S je proizvoljna σ–struktura $\mathfrak{M}$, pri čemu σ sadrži sve nelogičke simbole koji se pojavljuju u formulama iz B_S. Primijetimo da svaka klauzula $C = \{F_1, ..., F_n\}$ predstavlja zapravo univerzalno zatvorenu formulu $\forall x_1 ... \forall x_k (F_1 \vee \cdots \vee F_n)$, gdje skup $\{x_1, ..., x_k\}$ sadrži sve varijable koje imaju slobodni nastup u formuli $F_1 \vee \cdots \vee F_n$. Dakle ne moramo promatrati interpretaciju kao uređeni par $(\mathfrak{M}, v)$ gdje je v valuacija.

Interpretacije Herbrandove baze možemo promatrati tako da svakoj zatvorenoj atomarnoj formuli iz Herbrandove baze (u nastavku ćemo ih zvati atomima) pridružimo istinitosnu vrijednost. Dakle, elemente Herbrandove baze možemo promatrati kao propozicionalne varijable.

1.2 Rezolucija

U sljedećem poglavlju bavit ćemo se ispunjivošću skupa klauzula. Želimo povezati interpretaciju Herbrandove baze s ispunjivošću skupa klauzula. Za ovo će nam trebati pojam rezolvente.

Definicija 1.2.1. *Neka su* C_1 *i* C_2 *klauzule takve da postoji formula* L *takva da vrijedi* $L \in C_1$ *i* $\neg L \in C_2$*. Tada kažemo da su* C_1 *i* C_2 *kompatibilne klauzule te da su kompatibilne na komplementarnim literalima* L *i* $\neg L$*.* Rezolventa *klauzula* C_1 *i* C_2 *je klauzula*

$$Res(C_1, C_2) = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\})$$

Kažemo da su klauzule C_1 *i* C_2 *roditelji klauzule* $Res(C_1, C_2)$*.*

Primjer 1.2.2. *Neka su dane klauzule* $C_1 = \{P(a), Q(f(b)), \neg R(x)\}$ *i* $C_2 = \{\neg Q(f(b)), R(f(a))\}$*. Ovo su kompatibilne klauzule na komplementarnim literalima* $Q(f(b))$ *i* $\neg Q(f(b))$*. Njihova je rezolventa* $Res(C_1, C_2) = \{P(a), \neg R(x), R(f(a))\}$*.*

Mi ćemo željeti pronalaziti rezolvente klauzule iz skupa klauzula i atoma iz pripadajuće Herbrandove baze. Tada ova definicija rezolvente nije dovoljno snažna. Želimo definirati

rezolventu tako da je moguće dobiti istu iz dvije klauzule koje ne sadrže nužno komplementarne literale, ali se možda dva literala mogu *svesti* na komplementarne supstitucijom. U tu svrhu uvodimo pojam unifikacije.

Primjer 1.2.3. *Promotrimo opet klauzule iz primjera 1.2.2. Vidimo da je* $\neg R(x) \in C_1$ *i* $R(f(a)) \in C_2$*. Uz trenutnu definiciju rezolvente preko ovih literala ne možemo dobiti rezolventu jer nisu komplementarni. Primijetimo da supstitucijom* $\theta = \{x \longleftarrow f(a)\}$ *dobivamo komplementarne literale* $\neg R(f(a))$ *i* $R(f(a))$*, a samim time i kompatibilne klauzule* $C_1\theta$ *i* $C_2\theta$ *na parovima literala* $Q(f(b))$ *i* $\neg Q(f(b))$ *te* $\neg R(f(a))$ *i* $R(f(a))$*.*

Definicija 1.2.4. *Neka je* $U = \{A_1, \ldots, A_n\}$ *neki skup atomarnih formula.* Unifikator θ *je supstitucija takva da* $A_1\theta = \cdots = A_n\theta$*.* Najopćenitiji unifikator *od U je unifikator* μ *takav da se svaki unifikator* θ *od U može izraziti kao* $\theta = \mu\lambda$ *za neku supstituciju* λ*.*

Nemaju svake dvije atomarne formule unifikator. Očito je nemoguće pronaći unifikator za atomarne formule čiji su relacijski simboli različiti poput $P(x)$ i $Q(x)$, kao ni one čiji su vanjski funkcijski simboli različiti poput $P(f(x))$ i $P(g(y))$. Ni atomarne formule oblika $P(f(x))$ i $P(x)$ nemaju unifikator. Budući da je x u prvoj atomarnoj formuli dio većeg terma $f(x)$, niti jedna supstitucija ne može biti njihov unifikator. Dok god ova tri uvjeta nisu zadovoljena, dvije će atomarne formule imati unifikator.

Primjer 1.2.5. *Promotrimo atomarne formule* $A_1 = P(f(f(a)), g(y))$ *i* $A_2 = P(f(x), g(g(z)))$*. Dva unifikatora za formule* A_1 *i* A_2 *su, primjerice:*

$$\theta_1 = \{x \longleftarrow f(a), y \longleftarrow g(a), z \longleftarrow a\}$$

$$\theta_2 = \{x \longleftarrow f(a), y \longleftarrow g(g(a)), z \longleftarrow g(a)\}$$

jer vrijedi

$$A_1\theta_1 = P(f(f(a)), g(g(a))) = A_2\theta_1$$

$$A_1\theta_2 = P(f(f(a)), g(g(g(a)))) = A_2\theta_2.$$

Tvrdimo da je μ *dan sa*

$$\mu = \{x \longleftarrow f(a), y \longleftarrow g(z)\}.$$

najopćenitiji unifikator. Da bi supstitucija θ *bila unifikator za formule* A_1 *i* A_2 *očito mora sadržavati* $x \longleftarrow f(a)$*, a ako sadrži* $z \longleftarrow t$ *gdje je t neki term, onda mora sadržavati i* $y \longleftarrow g(t)$*. Tada je* $\theta = \mu\lambda$ *gdje je* $\lambda = \{z \longleftarrow t\}$*. U slučaju da* θ *ne sadrži* $z \longleftarrow t$*, onda mora sadržavati* $y \longleftarrow g(z)$ *pa je u tom slučaju* $\theta = \mu$*. Dakle,* μ *stvarno jest najopćenitiji unifikator za* A_1 *i* A_2*. Sada, uz supstitucije* $\lambda_1 = \{z \longleftarrow a\}$ *i* $\lambda_2 = \{z \longleftarrow g(a)\}$*, vrijedi* $\theta_1 = \mu\lambda_1$ *i* $\theta_2 = \mu\lambda_2$*. Na primjer,*

$$A_1\mu\lambda_1 = (P(f(f(a)), g(g(z))))\lambda_1 = P(f(f(a)), g(g(a))) = A_1\theta_1$$

$$A_1\mu\lambda_2 = (P(f(f(a)), g(g(z))))\lambda_2 = P(f(f(a)), g(g(g(a)))) = A_1\theta_2.$$

Neka je C klauzula s literalima $L_1, L_2, \dots L_n$ te θ neka supstitucija. Tada je $C\theta$ klauzula za koju vrijedi $L_i\theta \in C\theta$ za svaki $i \in \{1, \dots, n\}$. Sada možemo definirati rezolventu dvije klauzule koje ne sadrže komplementarne literale.

Definicija 1.2.6. *Neka su C_1 i C_2 dvije klauzule takve da postoje $L_1 \in C_1$ i $L_2 \in C_2$ takvi da za $\{L_1, \neg L_2\}$ postoji najopćenitiji unifikator μ. Tada je rezolventa klauzula C_1 i C_2 definirana sa*

$$Res(C_1, C_2) = (C_1\mu \setminus \{L_1\mu\}) \cup (C_2\mu \setminus \{L_2\mu\}).$$

Primjer 1.2.7. *Neka su $C_1 = \{P(a), Q(f(x)), \neg R(x)\}$ i $C_2 = \{\neg Q(f(b)), R(f(a))\}$. Promotrimo skup atomarnih formula $A = \{Q(f(x)), Q(f(b))\}$. Vrijedi $Q(f(b)) = \neg\neg Q(f(b))$ što je negacija literala $\neg Q(f(b)) \in C_2$. Lako se vidi da je najopćenitiji unifikator za skup A upravo $\mu = \{x \longleftarrow b\}$. Tada je $C_1\mu = \{P(a), Q(f(b)), \neg R(b)\}$ i $C_2\mu = \{\neg Q(f(b)), R(f(a))\}$ pa je rezolventa $Res(C_1, C_2) = \{P(a), \neg R(b), R(f(a))\}$.*

Primjer 1.2.8. *Neka su $C_1 = \{P(x)\}$ i $C_2 = \{\neg P(f(a))\}$. Lako se vidi da je najopćenitiji unifikator atomarnih formula $P(x)$ i $P(f(a))$ upravo $\mu = \{x \longleftarrow f(a)\}$. Tada je $Res(C_1, C_2) = \emptyset$, odnosno prazna klauzula.*

U gornjem primjeru dobili smo da je rezolventa dvije klauzule prazna klauzula $\emptyset$. Prazna klauzula nije ispunjiva jer ne može biti istinita niti za jednu interpretaciju. U sljedećem poglavlju bit će nam važne upravo takve klauzule, čija je rezolventa prazna klauzula. Razlog tomu je teorem iskazan nakon sljedeće pomoćne tvrdnje.

Lema 1.2.9. *Neka je C neka klauzula, θ neka supstitucija, te $\mathfrak{M}$ neka struktura. Ako vrijedi $\mathfrak{M} \models C$ tada imamo i $\mathfrak{M} \models C\theta$.*

Dokaz. Primijetimo da $C = \{F_1, \dots, F_n\}$ predstavlja zapravo univerzalno zatvorenu formulu $\forall x_1 \dots \forall x_k(F_1 \vee \dots \vee F_n)$. gdje skup $\{x_1, \dots, x_k\}$ sadrži sve varijable koje imaju slobodni nastup u formuli $F_1 \vee \dots \vee F_n$. Treba dokazati da $\mathfrak{M} \models \forall x_1 \dots \forall x_k(F_1 \vee \dots \vee F_n)$ povlači $\mathfrak{M} \models \forall x_1 \dots \forall x_k(F_1\theta \vee \dots \vee F_n\theta)$. Dokazat ćemo ekvivalentnu tvrdnju: ako za svaku valuaciju v vrijedi $\mathfrak{M} \models_v F_1 \vee \dots \vee F_n$, tada za svaku valuaciju u vrijedi $\mathfrak{M} \models_u F_1\theta \vee \dots \vee F_n\theta$.

Neka za svaku valuaciju v vrijedi $\mathfrak{M} \models_v F_1 \vee \dots \vee F_n$ te neka je u proizvoljna valuacija. Definiramo valuaciju w s pravilom pridruživanja $w(x) := u(x\theta)$. Budući da za svaku valuaciju v vrijedi $\mathfrak{M} \models_v F_1 \vee \dots \vee F_n$, onda posebno vrijedi i $\mathfrak{M} \models_w F_1 \vee \dots \vee F_n$. Dakle, postoji $i \in \{1, \dots, n\}$ tako da $\mathfrak{M} \models_w F_i$. Formule F_i može biti ili atomarna formula ili negacija atomarne formule. Pretpostavimo da je atomarna formula, to jest vrijedi $F_i = R(s_1, \dots, s_p)$. Tada po definiciji vrijedi $(w(s_1), \dots, w(s_p)) \in R^{\mathfrak{M}}$.

Dokazat ćemo indukcijom po duljini terma da za svaki term t vrijedi $w(t) = u(t\theta)$. Ako je term duljine 1, onda je ili varijabla ili konstanta. Ako je varijabla, tvrdnja vrijedi po definiciji, a ako je $t = a$, gdje je a neka konstanta, onda:

$$w(a) = a^{\mathfrak{M}} = u(a) = u(a\theta).$$

Pretpostavimo da vrijedi $w(t) = u(t\theta)$ za sve terme duljine manje od $m \in \mathbb{N}$. Neka je t term duljine m. Tada je $t = f(t_1, \ldots, t_r)$ za neki funkcijski simbol f i terme $t_1, \ldots, t_r$ koji su svi duljine manje od m. Imamo:

$$\begin{aligned} w(t) &= w(f(t_1, \ldots, t_r)) = f^{\mathfrak{M}}(w(t_1), \ldots, w(t_r)) = f^{\mathfrak{M}}(u(t_1\theta), \ldots, u(t_r\theta)) \\ &= u(f(t_1\theta, \ldots, t_r\theta)) = u(f(t_1, \ldots, t_r)\theta) = u(t\theta), \end{aligned}$$

gdje u trećoj jednakosti koristimo pretpostavku indukcije. Sada iz $(w(s_1), \ldots, w(s_p)) \in R^{\mathfrak{M}}$ slijedi $(u(s_1\theta), \ldots, u(s_p\theta)) \in R^{\mathfrak{M}}$. Dakle, $\mathfrak{M} \models_u F_i\theta$ pa onda i $\mathfrak{M} \models_u F_1\theta \vee \ldots \vee F_n\theta$. Slično se može dokazati i slučaj kad je F_i negacija atomarne formule. □

Teorem 1.2.10. *Ako je C rezolventa nekih klauzula C_1 i C_2, tada C logički slijedi iz $\{C_1, C_2\}$.*

Dokaz. Kako je C rezolventa klauzula C_1 i C_2, onda postoje $L_1 \in C_1$ i $L_2 \in C_2$ takvi da za $\{L_1, \neg L_2\}$ postoji najopćenitiji unifikator μ te $C = (C_1\mu \setminus \{L_1\mu\}) \cup (C_2\mu \setminus \{L_2\mu\})$. Neka je $\mathfrak{M}$ struktura tako da vrijedi $\mathfrak{M} \models C_1$ i $\mathfrak{M} \models C_2$. Iz leme 1.2.9 znamo da također vrijedi $\mathfrak{M} \models C_1\mu$ i $\mathfrak{M} \models C_2\mu$. Također, ne može istovremeno vrijediti $\mathfrak{M} \models L_1\mu$ i $\mathfrak{M} \models L_2\mu$ jer je $L_1\mu = \neg L_2\mu$. Bez smanjenja općenitosti pretpostavimo da $\mathfrak{M} \not\models L_1\mu$. Tada $C_1\mu \setminus \{L_1\mu\}$ ne može biti prazna klauzula, to jest mora sadržavati barem jedan literal L takav da $\mathfrak{M} \models L$ jer vrijedi $\mathfrak{M} \models C_1\mu$. Dakle, vrijedi $\mathfrak{M} \models C_1\mu \setminus \{L_1\mu\}$ pa onda i za $C = (C_1\mu \setminus \{L_1\mu\}) \cup (C_2\mu \setminus \{L_2\mu\})$ vrijedi $\mathfrak{M} \models C$. □

Sada vidimo što možemo zaključiti kada dobijemo da je rezolventa dvije klauzule prazna klauzula. Budući da prazna klauzula nije ispunjiva, onda nije ispunjiv ni skup koji se sastoji od početnih dviju klauzula. Treba primijetiti da se ovo može primijeniti samo na skup koji se sastoji od dvije jednočlane klauzule što je veliko ograničenje. Za veći skup složenijih klauzula trebat će nam općenitiji rezultat.

1.3 Herbrandov teorem

Pitanje kada neki skup klauzula nije ispunjiv bit će od velike važnosti u kasnijim poglavljima kada ćemo proučavati rad programa HERBY. U suštini, program HERBY će dokazivati teoreme tako što će negirati zaključak pa će do dokaza doći uspije li dokazati da takav skup klauzula nije ispunjiv. No, za to nam treba neka efektivna metoda. Pomoći će nam Herbrandov teorem koji u svom općenitom obliku (iz [2] gdje je i dokaz) glasi:

Teorem 1.3.1 (Herbrandov teorem)**.** *Skup klauzula S nije ispunjiv ako i samo ako postoji konačan skup zatvorenih instanci klauzula iz S koji nije ispunjiv.*

Želimo doći do verzije teorema koju možemo koristiti u praksi. Za to ćemo koristiti interpretaciju Herbrandove baze skupa klauzula.

Definicija 1.3.2. *Neka je S skup klauzula, B_S pripadna Herbrandova baza, te $\mathfrak{M}$ neka interpretacija Herbrandove baze B_S. Za svaku atomarnu formulu $A \in B_S$ uvodimo oznaku $A^{\mathfrak{M}}$ ovako:*

$$A^{\mathfrak{M}} = \begin{cases} A, & \text{ako } \mathfrak{M} \models A; \\ \neg A, & \text{ako } \mathfrak{M} \not\models A \end{cases}$$

U nastavku ćemo govoriti da interpretacija $\mathfrak{M}$ poništava *klauzulu $C \in S$ ako postoji konačan $B_C \subseteq B_S$ tako da za svaki literal $L \in C$ postoji atomarna formula $A \in B_C$ tako da $Res(L, A^{\mathfrak{M}}) = \emptyset$.*

Kažemo da interpretacija poništava skup klauzula ako poništava barem jednu klauzulu iz tog skupa.

Primjer 1.3.3. *Neka je $S = \{\{\neg P(a)\}, \{P(x), \neg Q(a)\}\}$ klauzula. Promotrimo interpretaciju $\mathfrak{M}$ koju čini struktura $\mathfrak{M} = (\mathbb{N}, \varphi)$ gdje je $P^{\mathfrak{M}}$ jednomjesna relacija na $\mathbb{N}$ definirana sa: $n \in P^{\mathfrak{M}}$ ako i samo ako je n paran, $Q^{\mathfrak{M}}$ jednomjesna relacija definirana je sa: $n \in Q^{\mathfrak{M}}$ ako i samo ako je n neparan i gdje je $a^{\mathfrak{M}} = 1$.*

Herbrandova baza skupa S je $B_S = \{P(a), Q(a)\}$. Očito vrijedi $\mathfrak{M} \not\models P(a)$ i $\mathfrak{M} \models Q(a)$. Iz ovoga slijedi da $P(a)^{\mathfrak{M}} = \neg P(a)$ i $Q(a)^{\mathfrak{M}} = Q(a)$. Označimo $C_1 := \{\neg P(a)\}$ i $C_2 := \{P(x), \neg Q(a)\}$. Neka je $B_{C_2} = B_S$. Lako se vidi da je $Res(P(x), P(a)^{\mathfrak{M}}) = \emptyset$ i $Res(\neg Q(a), Q(a)^{\mathfrak{M}}) = \emptyset$, to jest za svaki $L \in C_2$ postoji $A \in B_{C_2}$ tako da $Res(L, A^{\mathfrak{M}}) = \emptyset$. Dakle, interpretacija $\mathfrak{M}$ poništava klauzulu C_2.

Lako se vidi da $\mathfrak{M}$ ne poništava C_1 jer nije moguće naći konačan $B_{C_1} \subseteq B_S$ sa željenim svojstvom. Ako definiramo interpretaciju $\mathfrak{M}' = (\mathbb{N}, \varphi')$ tako da $P^{\mathfrak{M}'}$ označava neparnost, a $Q^{\mathfrak{M}'}$ parnost, onda možemo vidjeti da takva interpretacija poništava C_1. Neka je $B_{C_1} = \{P(a)\}$. Kako je $\mathfrak{M}' \models P(a)$, onda $P(a)^{\mathfrak{M}'} = P(a)$, a iz ovoga se lako vidi $Res(\neg P(a), P(a)^{\mathfrak{M}'}) = \emptyset$. Odnosno, $\mathfrak{M}'$ poništava C_1.

Za ovu verziju Herbrandovog teorema, treba nam pojam parcijalne interpretacije Herbrandove baze. Ako je S neki skup klauzula, onda je *parcijalna interpretacija* od B_S proizvoljna σ–struktura $\mathfrak{M}$, pri čemu σ sadrži sve nelogičke simbole koji se pojavljuju u formulama iz B gdje je $B \subseteq B_S$ konačan. Može se promatrati kao i dodijeljivanje istinitosnih vrijednosti nekom određenom broju atomarnih formula iz B_S.

Teorem 1.3.4. *Neka je S skup klauzula. Ako postoji neki $k \in \mathbb{N}$ takav da svaka parcijalna interpretacija već na prvih k atoma kanonske Herbrandove baze B_S poništava skup S, tada S nije ispunjiv.*

Dokaz. Neka je $k \in \mathbb{N}$ takav da svaka parcijalna interpretacija na prvih k atoma kanonske Herbrandove baze B_S poništava skup S. Označimo sa B skup prvih k atomarnih formula iz B_S. Promotrimo svaku atomarnu formulu iz B kao propozicionalnu varijablu. Tada će istinitosna tablica imati 2^k redaka. Svaki redak tablice možemo promatrati kao skup svih interpretacija $\mathfrak{M}$ za koje za svaku atomarnu formulu $A \in B$ vrijedi $\mathfrak{M} \models A$ ako je atomu A

pridružena istinitosna vrijednost 1 te $\mathfrak{M} \not\models A$ ako je A pridružena istinitosna vrijednost 0. Za svaki redak tablice fiksirajmo jednu interpretaciju tako da dobijemo skup $\{\mathfrak{M}_1, \ldots, \mathfrak{M}_{2^k}\}$.

Neka je $\mathfrak{M}_i$, gdje je $i \in \{1, \ldots, 2^k\}$, proizvoljna interpretacija iz tog skupa. Tada $\mathfrak{M}_i$ poništava skup S pa postoji klauzula $C_i = \{L_1, L_2, \ldots, L_n\}$ iz S takva da je $\mathfrak{M}_i$ poništava. Iz definicije tada slijedi da postoji konačan B_{C_i} podskup od B_S tako da za svaki literal $L_j \in C_i$ postoji atomarna formula $A_j \in B_{C_i}$ tako da vrijedi $Res(L_j, A_j^{\mathfrak{M}_i}) = \emptyset$. Po definiciji rezolvente znamo da postoji najopćenitiji unifikator za sve L_j i $\neg A_j^{\mathfrak{M}_i}$ te ga označimo s μ_j. Dakle, imamo

$$L_j\mu_j = \neg A_j^{\mathfrak{M}_i}\mu_j.$$

Budući da je A_j element Herbrandove baze, A_j je zatvorena atomarna formula ili negacija zatvorene atomarne formule, a po definiciji je

$$A_j^{\mathfrak{M}_i} = \begin{cases} A_j & \text{ako } \mathfrak{M}_\text{i} \models A_j \\ \neg A_j & \text{ako } \mathfrak{M}_i \not\models A_j \end{cases}$$

pa je i $A_j^{\mathfrak{M}_i}$ zatvorena atomarna formula ili negacija iste. Zato vrijedi $L_j\mu_j = \neg A_j^{\mathfrak{M}_i}\mu_j = \neg A_j^{\mathfrak{M}_i}$ za svaki $j \in \{1, \ldots, n\}$.

Za svaki $j \in \{1, \ldots, n\}$ po definiciji vrijedi $\mathfrak{M}_\text{i} \models A_j^{\mathfrak{M}_i}$ pa onda i $\mathfrak{M}_\text{i} \not\models L_j\mu_j$ jer je $L_j\mu_j = \neg A_j^{\mathfrak{M}_i}$. Definirajmo klauzulu $C_i' := \{L_1\mu_1, L_2\mu_2, \ldots, L_n\mu_n\}$. Vrijedi $\mathfrak{M}_i \not\models C_i'$ jer je C_i' disjunkcija literala $L_1\mu_1, \ldots, L_n\mu_n$. Primijetimo još da je C_i' zatvorena instanca klauzule C_i.

Budući da je i bio proizvoljan, za svaku takvu parcijalnu interpretaciju $\mathfrak{M}_i$, gdje je $i \in \{1, 2, \ldots, 2^k\}$ možemo na prethodan način konstruirati klauzulu C_i' za koju vrijedi $\mathfrak{M}_i \not\models C_i'$. Označimo sa S' skup svih tako dobivenih klauzula. Vrijedi $card(S') \leq 2^k$ jer za neke $i \neq j$ može vrijediti $C_i' = C_j'$. Iz načina na koji smo odabrali interpretacije $\{\mathfrak{M}_1, \ldots, \mathfrak{M}_n\}$ lako se vidi da skup S' nije ispunjiv. Štoviše, to je konačan skup zatvorenih instanci od S koji nije ispunjiv. Po teoremu 1.3.1 skup S nije ispunjiv. □

Sada pomoću teorema 1.3.4 ispunjivost možemo odrediti istinitosnom tablicom na podskupu Herbrandove baze.

Primjer 1.3.5. *Promotrimo skup klauzula*

$$S = \{\{\neg P(a), Q(x)\}, \{P(f(x)), Q(a)\}, \{\neg P(f(x)\}, \{\neg Q(x)\}\}.$$

Herbrandov univerzum ovog skupa je $H_S = \{a, f(a), f(f(a)), f(f(f(a))), \ldots\}$ *pa je onda kanonska Herbrandova baza* $B_S = \{P(a), Q(a), P(f(a)), Q(f(a)), \ldots\}$. *Zapišimo skup klauzula* S *u preglednijem obliku:*

$$\neg P(a) \vee Q(x) \quad (1)$$

$$P(f(x)) \vee Q(a) \quad (2)$$

$$\neg P(f(x)) \quad (3)$$

$$\neg Q(x) \quad (4)$$

Pogledajmo kako izgleda istinitosna tablica za prva tri atoma kanonske Herbrandove baze B_S.

P(a)	*Q(a)*	*P(f(a))*	*poništava*
1	*1*	*1*	*(3), (4)*
1	*1*	*0*	*(4)*
1	*0*	*1*	*(1), (3)*
1	*0*	*0*	*(1), (2)*
0	*1*	*1*	*(3), (4)*
0	*1*	*0*	*(4)*
0	*0*	*1*	*(3)*
0	*0*	*0*	*(2)*

U posljednjem stupcu gornje tablice imamo oznake klauzula koje određena parcijalna interpretacija poništava. Vidimo da niti jedan redak tog stupca nije prazan, dakle svaka parcijalna interpretacija na prva tri atoma kanonske Herbrandove baze od S poništava barem jednu klauzulu iz S. To točno znači da svaka takva interpretacija poništava S. Odnosno, po teoremu 1.3.4, skup klauzula S nije ispunjiv.

1.4 Semantička stabla

Kao što smo mogli vidjeti u primjeru 1.3.5 za neke jednostavne skupove klauzula dovoljno je koristiti istinosnu tablicu ili čak istinosnu tablicu za prvih k atoma kanonske Herbrandove baze da bi se dokazalo da skup klauzula nije ispunjiv. Za kompliciranije primjere, kakvima se uglavnom bavi program HERBY, potrebno je uvesti još i pojam semantičkog stabla. Uz semantička stabla i još jednu verziju Herbrandovog teorema moći ćemo napisati algoritam kojim program HERBY dokazuje teoreme.

Definicija 1.4.1. *Neka je S skup klauzula.* Semantičko stablo *dubine D skupa klauzula S je binarno stablo T s korijenom R, takvo da vrijedi:*

- *Neka je* $B_S = \{hb_1, hb_2, \ldots, hb_d, \ldots\}$ *Herbrandova baza od S. Tada je svaki lijevi brid na dubini d označen s* hb_d*, dok je svaki desni brid označen s* $\neg hb_d$.

- *Svakom čvoru N pridružen je skup klauzula $K(N)$ takav da:*
  - *Ako je $N = R$ korijen, tada je $K(N) = S$.*
  - *Neka je N čvor na dubini d, neka je $R, N_1, \ldots, N_{d-1}, N$ put od korijena R do čvora N te je brid koji vodi iz N_{d-1} u N označen nekim literalom L. Tada je $K(N)$ skup svih rezolventi literala L sa svim klauzulama u skupu $K(R) \cup K(N_1) \cup \cdots \cup K(N_{d-1})$ te sa svim klauzulama tako dobivenima. Skup $K(N)$ ne sadrži one rezolvente koje su već sadržane u $K(R) \cup K(N_1) \cup \cdots \cup K(N_{d-1})$.*
- *Čvor N je list ako $\emptyset \in K(N)$.*
- *Svi čvorovi na dubini D su listovi.*

Ako je $\emptyset \in K(N)$ za neki čvor N, gdje je $\emptyset$ prazna klauzula, onda parcijalna interpretacija dobivena skupom atoma na putu od R do N poništava barem jednu klauzulu unutar S. Dakle, ta parcijalna interpretacija poništava skup S.

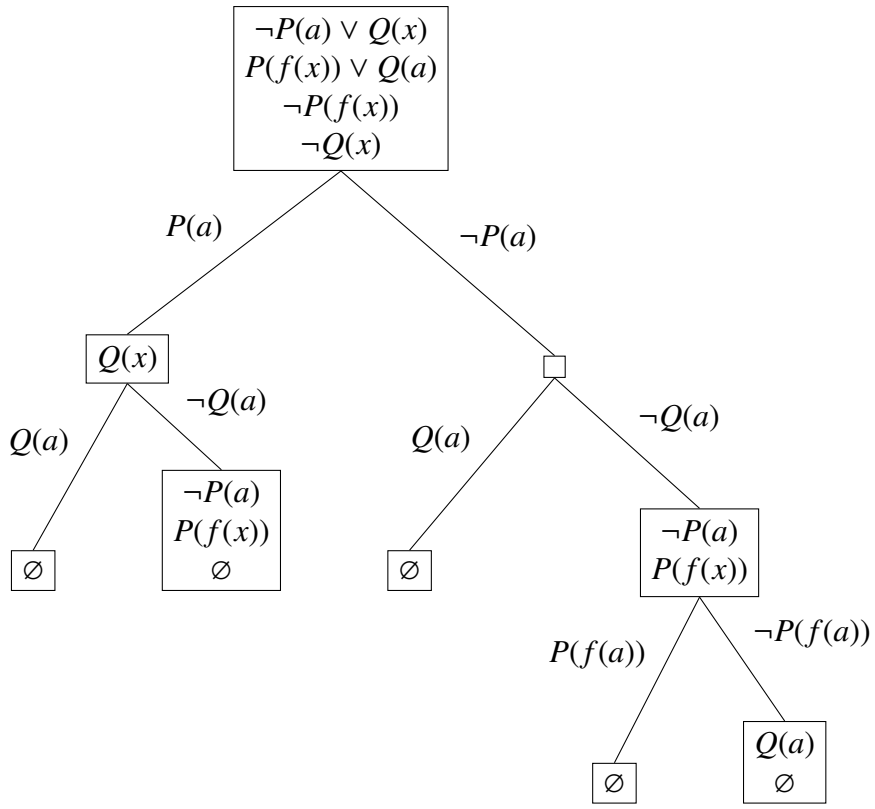
Kanonsko semantičko stablo dubine D je semantičko stablo dubine D u kojem je svaki lijevi brid na dubini d označen d-tim atomom kanonske Herbrandove baze, a svaki desni brid označen komplementom tog atoma. Nekanonsko semantičko stablo ima bridove označene atomima neke parcijalne interpretacije.

Primjer 1.4.2. *Neka je skup klauzula $S = \{\{\neg P(a), Q(x)\}, \{P(f(x)), Q(a)\}, \{\neg P(f(x)\}, \{\neg Q(x)\}\}$ iz primjera 1.3.5. Na slici 1.1 vidimo kanonsko semantičko stablo dubine 3. Kanonska Herbrandova baza skupa S je $B_S = \{P(a), Q(a), P(f(a)), Q(f(a)), \ldots\}$ pa tim redoslijedom uzimamo atome i označavamo bridove stabla. Bridovi na dubini 1 su označeni atomom $P(a)$ i njegovom negacijom, na dubini 2 atomom $Q(a)$ i njegovom negacijom, a na dubini 3 atomom $P(f(a))$ i njegovom negacijom. Svakom je čvoru onda pridružen skup rezolvenata kako je opisano u definiciji.*

Uz uveden pojam semantičkih stabala te teorem 1.3.4 možemo dobiti još jednu verziju Herbrandovog teorema.

Teorem 1.4.3. *Ako S nije ispunjiv skup klauzula, tada postoji neki $k \in \mathbb{N}$ takav da svaki put u kanonskom semantičkom stablu od S, koji počinje u korijenu R i duljine je najviše k, završava čvorom N takvim da $\emptyset \in K(N)$. Kažemo da je semantičko stablo tada zatvoreno.*

Sada imamo sve što nam je potrebno kako bi program HERBY uspješno dokazao teorem. Samo moramo pokazati da zatvoreno semantičko stablo postoji za skup klauzula koji se sastoji od aksioma i negiranog zaključka. To detaljno navodimo u sljedećoj napomeni.



Slika 1.1: Semantičko stablo dubine 3 za skup klauzula iz primjera 1.3.5

Napomena 1.4.4 (Dokazivanje teorema konstrukcijom zatvorenog semantičkog stabla)**.** *Neka je S skup klauzula. Konstruiramo semantičko stablo skupa S na sljedeći način.*

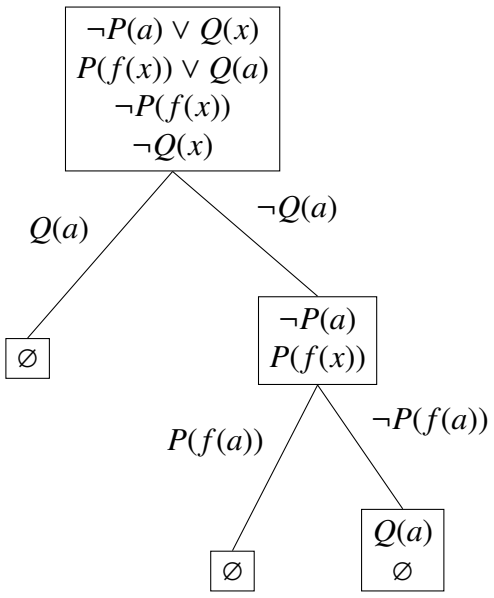
1. *Postavimo dubinu D semantičkog stabla na* 0*.*
2. *Provodimo* depth-first *konstrukciju semantičkog stabla dubine D.*
   a) *Ako za sve čvorove dubine manje ili jednake D vrijedi $\varnothing \in K(N)$ konstrukcija je gotova i algoritam staje.*
   b) *Kada za neki čvor M na dubini D ne vrijedi $\varnothing \in K(N)$, povećamo D za jedan, izaberemo neki atom iz Herbrandove baze te označimo sve bridove na dubini D ovim atomom ili njegovom negacijom. Ne smijemo izabrati atom koji je već ranije izabran. Nastavljamo s konstrukcijom M.*

Primjer 1.4.5. *Konstruirat ćemo zatvoreno semantičko stablo za skup klauzula iz primjera 1.3.5. Znamo da taj skup klauzula nije ispunjiv te da će algoritam stati.*

Zatvoreno semantičko stablo nalazi se na slici 1.2.

Objasnimo postupak. U prvom koraku označili smo bridove atomima $Q(a)$ i $\neg Q(a)$. Za atom $Q(a)$ jedina kompatibilna klauzula iz početnog skupa klauzula je četvrta $\neg Q(x)$ gdje supstitucijom x s a dobivamo praznu klauzulu čime smo dobili i list. Klauzula $\neg Q(a)$ kompatibilna je s prvom i drugom klauzulom početnog skupa iz čega dobivamo skup rezolvenata u pripadajućem čvoru. Sljedeći odabrani atom je $P(f(a))$ koji lijevo odmah daje praznu klauzulu kao rezolventu u paru s trećom klauzulom početnog skupa, a desno s drugom klauzulom prethodnog čvora. Time smo dobili zatvoreno semantičko stablo i pokazali da početni skup klauzula nije ispunjiv.

Da smo se u prethodnom primjeru odlučili na kanonski redoslijed odabira atoma, po stablu sa slike 1.1 vidimo da bi nam trebala još jedna razina dubine kako bi došli do zatvorenog stabla. Prirodno se postavlja pitanje na koji se način odabire atom iz Herbrandove baze za konstrukciju stabla. Uzimati atome kanonskim redom je jedan od načina, ali nije vrlo efektivan. U drugom ćemo poglavlju predstaviti heuristike koje program HERBY koristi za odabir atoma.



Slika 1.2: Zatvoreno semantičko stablo za skup klauzula iz primjera 1.3.5.

Poglavlje 2

Opis rada programa HERBY

U ovom poglavlju dajemo opis programa HERBY. Vidjet ćemo kako se unutar programa HERBY realizira logika prvog reda. Zatim, razmatrat ćemo kako se koristi teorija uvedena u prethodnom poglavlju te na koji način korisnik treba zadati teoreme. Objasnit ćemo heuristike za izbor atoma prilikom konstrukcije zatvorenog semantičkog stabla.

2.1 Reprezentacija formula logike prvog reda unutar programa HERBY

U ovoj točki ćemo navesti koji se znakovi koriste za pisanje formula logike prvog reda u programu HERBY. Sada navodimo simbole za logičke veznike i kvantifikatore te pomoćne znakove koji se koriste u programu HERBY.

Za logičke veznike koriste se sljedeći simboli: & (konjunkcija), | (disjunkcija), ~ (negacija), => (implikacija) i <=> (ekvivalencija).

Za kvantifikatore koriste se sljedeći simboli: @ (univerzalni kvantifikator) i ! (egzistencijalni kvantifikator).

Važno je istaknuti da se uobičajni simboli, kao što su primjerice $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ i $\exists$ ne koriste jer nisu dostupni na standardnim tipkovnicama računala.

Oznake pomoćnih simbola su standardne pa ih navodimo bez istaknutih naziva: **(**, **)**, **{**, **}**, **:**, **.** i **,** (zarez).

Primjer 2.1.1. *Promotrimo kako jedna formula logike prvog reda izgleda zapisana korištenjem simbola programa HERBY. Izabrat ćemo formulu u kojoj se pojavljuju oba kvantifikatora i svi logički veznici:*

$$\forall x(\neg P(x) \rightarrow Q(f(y))) \wedge ((P(a) \leftrightarrow \forall x Q(x)) \vee \exists y P(y))$$

Zapisano simbolima programa HERBY prethodna formula postaje:

$$@x\{\sim P(x) => Q(f(y))\}\&\{\{P(a) <=> @xQ(x)\}|!yP(y)\}$$

2.2 Zadavanje teorema u ispravnom obliku

Teorem koji želimo dokazati mora biti zadan kao skup klauzula, a osim pretpostavke i zaključka samog teorema mora se sastojati i od aksioma koji su relevantni za područje o kojem se radi. Program HERBY implementira algoritam iz napomene 1.4.4 pomoću kojeg dokazujemo da skup klauzula nije ispunjiv. Dakle, da bi mogli dokazati teorem pomoću programa HERBY prvo moramo negirati zaključak te zatim pretvoriti sve formule (aksiome, pretpostavke i negirani zaključak) u skup klauzula. Za ovo se koristi pomoćni program COMPILE, ali može se učiniti i ručno. Postupak navodimo ovdje.

Napomena 2.2.1 (Postupak pretvorbe formule logike prvog reda u skup klauzula)**.**

1. *Eliminiramo* $\rightarrow$ *i* $\leftrightarrow$*:*
   - *Zamijenimo* $A \leftrightarrow B$ *s* $(A \rightarrow B) \wedge (B \rightarrow A)$.
   - *Zamijenimo* $A \rightarrow B$ *s* $\neg A \vee B$.
2. *Izvršavamo sljedeće pretvorbe s negacijom dok svaka negacija nije oblika* $\neg A$ *gdje je A atomarna formula:*
   - *zamijenimo* $\neg\neg A$ *s A,*
   - *zamijenimo* $\neg(A \vee B)$ *s* $\neg A \wedge \neg B$*,*
   - *zamijenimo* $\neg(A \wedge B)$ *s* $\neg A \vee \neg B$*,*
   - *zamijenimo*$\neg(\forall xA)$ *s* $(\exists x \neg A)$*,*
   - *zamijenimo* $\neg(\exists xA)$ *s* $(\forall x \neg A)$.
3. *Preimenujemo varijable tako da niti jedna dva kvantifikatora ne kvantificiraju istu varijablu.*
4. *Svaku egzistencijalno kvantificiranu varijablu zamijenimo* Skolemovom konstantom *ili* Skolemovim funkcijskim simbolom *na sljedeći način:*
   - *Ako* $\exists x$ *nije u dosegu nekog univerzalnog kvantifikatora, zamijenimo svako pojavljivanje od x novom konstantom - Skolemovom konstantom - koja se ne pojavljuje nigdje drugdje u formuli.*

- *Ako je $\exists x$ unutar dosega jednog ili više univerzalnih kvantifikatora $\forall u, \forall v, \forall w, \ldots$, zamijenimo svako pojavljivanje od x Skolemovom funkcijom koja se ne pojavljuje nigdje drugdje u formuli i čiji argumenti su univerzalno kvantificirane varijable $u, v, w, \ldots$.*

5. *Sada su sve varijable univerzalno kvantificirane. Izbacimo sve univerzalne kvantifikatore iz formule.*

6. *Pretvaramo formulu u konjuktivnu normalnu formu ponavljanjem sljedećih pravila:*

   - *zamijenimo $A \vee (B \wedge C)$ s $(A \vee B) \wedge (A \vee C)$ i*
   - *zamijenimo $(A \wedge B) \vee C$ s $(A \vee C) \wedge (B \vee C)$.*

7. *Zapišemo dobivenu KNF kao skup klauzula gdje svaki $\wedge$ dijeli dvije klauzule.*

Primjer 2.2.2. *Pokazat ćemo svaki korak pretvorbe za formulu*

$$\forall x(\neg P(x) \leftrightarrow \forall y Q(y)) \vee \exists x \neg P(x).$$

Prvo se moramo riješiti bikondicionala u formuli $\neg P(x) \leftrightarrow \forall y Q(y)$ pa dobivamo:

$$\neg P(x) \rightarrow \forall y Q(y) \wedge \forall y Q(y) \rightarrow \neg P(x).$$

Sada eliminacijom kondicionala dobivamo:

$$(\neg\neg P(x) \vee \forall y Q(y)) \wedge (\neg(\forall y Q(y)) \vee \neg P(x)).$$

Ekvialentna forma početne formule nakon prvog koraka izgleda ovako:

$$\forall x((\neg\neg P(x) \vee \forall y Q(y)) \wedge (\neg(\forall y Q(y)) \vee \neg P(x))) \vee \exists x \neg P(x)$$

U drugom se koraku bavimo negacijama. Imamo jednu duplu negaciju, tj. podformulu $\neg\neg P(x)$, koju mijenjamo u $P(x)$. Podformula $\neg(\forall y Q(y))$ nije u željenom obliku $\neg A$ gdje je A atomarna formula. To po navedenim pravilima mijenjamo u $\exists y \neg Q(y)$. Dakle, nakon drugog koraka imamo:

$$\forall x((P(x) \vee \forall y Q(y)) \wedge (\exists y \neg Q(y) \vee \neg P(x))) \vee \exists x \neg P(x)$$

U sljedećem koraku samo mijenjamo imena varijabli tako da svi kvantifikatori kvantificiraju različitu varijablu. Egzistencijalno kvantificiranu varijablu y mijenjamo u varijablu z dok egzistencijalno kvantificiranu varijablu x mijenjamo u varijablu u. Nakon trećeg koraka dobivamo sljedeću formulu:

$$\forall x((P(x) \vee \forall y Q(y)) \wedge (\exists z \neg Q(z) \vee \neg P(x))) \vee \exists u \neg P(u)$$

Sada mijenjamo egzistencijalno kvantificirane varijable Skolemovim funkcijama ili konstantama, dakle upravo one varijable z i u uvedene u prošlom koraku. Varijablu z mijenjamo Skolemovom funkcijom $f(x)$ jer je $\exists z$ unutar dosega kvantifikatora $\forall x$. Varijablu u mijenjamo samo Skolemovom konstantnom a jer $\exists u$ nije u dosegu niti jednog univerzalnog kvantifikatora. Na taj način poslije četvrtog koraka dobivamo sljedeću formulu:

$$\forall x((P(x) \vee \forall y Q(y)) \wedge (\neg Q(f(x)) \vee \neg P(x))) \vee \neg P(a)$$

U petom koraku jednostavno izbacujemo sve univerzalne kvantifikatore. Dakle:

$$((P(x) \vee Q(y)) \wedge (\neg Q(f(x)) \vee \neg P(x))) \vee \neg P(a)$$

Kako bi pretvorili formulu u konjunktivnu normalnu formu koristimo zamjenu $(A \wedge B) \vee C$ u $(A \vee C) \wedge (B \vee C)$ gdje je nama $C = \neg P(a)$. Nakon šestog koraka imamo:

$$(P(x) \vee Q(y) \vee \neg P(a)) \wedge (\neg Q(f(x)) \vee \neg P(x) \vee \neg P(a))$$

Ostaje samo pretvoriti formulu u skup klauzula. Nakon sedmog (i posljednjeg) koraka konačno dobivamo traženi skup klauzula:

$$\{\{P(x), Q(y), \neg P(a)\}, \{\neg Q(f(x)), \neg P(x), \neg P(a)\}\}$$

2.3 Heuristike za odabir atoma

U prethodnom smo poglavlju pokazali da možemo konstruirati zatvoreno kanonsko semantičko stablo za skup klauzula koji nije ispunjiv. U praksi konstruiranje zatvorenih kanonskih semantičkih stabala nije jako efikasno jer često zahtijeva odabir previše atoma prije nego dobijemo zatvoreno stablo. No, u prethodnom smo poglavlju također primijetili da semantička stabla ne moraju biti kanonska, a kada je to slučaj, može se dobiti puno jači dokazivač teorema. Program HERBY koristi nekoliko heuristika kako bi poboljšao odabir atoma u svakom čvoru semantičkog stabla.

Radi se o pet heuristika za izbor atoma te jednoj heuristici za izbor konstanti. Šesta heuristika za odabir atoma daje samom korisniku opciju izbora atoma. Nećemo detaljno opisati sve heuristike, ali ćemo dati glavnu ideju kojom se koriste za efikasan izbor atoma. Zato je potrebno uvesti pojam dedukcije ili rezolucije.

Definicija 2.3.1. *Neka je S skup klauzula.* Dedukcija *(rezolucija) klauzule C iz S je konačan niz klauzula $C_1, C_2, \ldots, C_k$ takav da je svaki C_i ili klauzula iz S ili rezolventa neke dvije klauzule C_j i C_l tako da $j, l < i$ te je $C_k = C$.*

Primjer 2.3.2. *Neka je $S = \{\{P(f(a)), \neg Q(x)\}, \{Q(a)\}, \{\neg P(x)\}\}$. Dedukcija prazne klauzule iz S je sljedeća:*

1. $C_1 = P(f(a)) \vee \neg Q(x)$
2. $C_2 = Q(a)$
3. $C_3 = \neg P(x)$
4. $C_4 = Res(C_1, C_2) = P(f(a))$
5. $C_5 = Res(C_3, C_4) = \varnothing$

Upravo skupovi klauzula kao iz prethodnog primjera, oni iz kojih se dedukcijom može dobiti prazna klauzula, su temelj heuristika za odabir atoma programa HERBY. Ako je N čvor za koji je potrebno izabrati atom, program HERBY čuva klauzule pridružene čvorovima na putu od korijena do (i uključujući) čvora N u listi koja se zove *clist* koja može čuvati i do 10 000 klauzula. Polazne klauzule nalaze se na početku ove liste.

Prilikom ispisa rezultata program HERBY će ispisati sve atome korištene za konstrukciju zatvorenog semantičkog stabla kao i heuristike korištene za njihovo pronalaženje. Te će heuristike biti označene skraćenicama. Ovdje proučavamo četiri takve heuristike označene s ASH1, ASH2, ASH3 i ASH4 (*atom selection heuristic*). Ove heuristike za odabir atoma pretražuju *clist* pokušavajući pronaći skupove klauzula iz kojih je dedukcijom moguće dobiti praznu klauzulu.

U slučaju da je takav skup pronađen, promatramo dedukciju $C_1, \ldots, C_{k-1}, \varnothing$. Točnije, promatraju se sve klauzule C_i koje su nastale kao rezolventa neke dvije klauzule C_j i C_l takve da $j, l < i$. Iz $Res(C_j, C_l) = C_i$ znamo da postoje literali $L_1 \in C_j$ i $L_2 \in C_l$ takvi da postoji najopćeniti unifikator μ za $\{L_1, \neg L_2\}$. Kao jedan od odabranih atoma uzimamo $L_1\mu$, odnosno njegovu negaciju. Atomi se biraju "unatrag", to jest najprije će se promatrati ona rezolventa koja je posljednja nastupila u dedukciji pa one ranije. Naravno da ovako odabrane atomarne formule mogu sadržavati varijable, ali program HERBY ima heuristiku za odabir konstanti upravo za takav slučaj kojom supstituira varijable odabranih atomarnih formula nekom konstantom.

Heuristika ASH1 traži par klauzula od kojih se obje sastoje od samo jednog literala takve da je njihova rezolventa prazna klauzula. Klauzule koje se sastoje od samo jednog literala još se zovu i *unarne klauzule* te ćemo ih nadalje tako zvati. Ako odaberemo atom kako je prethodno opisano, sigurno će oba djeteta nastala odabirom tog atoma u svom skupu dobivenih klauzula sadržavati praznu klauzulu.

Primjer 2.3.3. *Pretpostavimo da se u* clist *nalaze unarne klauzule* $C_1 = \{P(x, f(a))\}$ *i* $C_2 = \{\neg P(f(y), y)\}$. *Lako je vidjeti da vrijedi* $Res(C_1, C_2) = \varnothing$ *pa će heuristika ASH1 kao sljedeći odabrani atom odabrati* $(P(x, f(a))\mu$ *gdje je* μ *najopćenitiji unifikator za* $\{P(x, f(a)), P(f(y), y)\}$. *Dakle,* $\mu = \{x \longleftarrow f(f(a)), y \longleftarrow f(a)\}$, *a odabrani atom je* $P(f(f(a)), f(a))$.

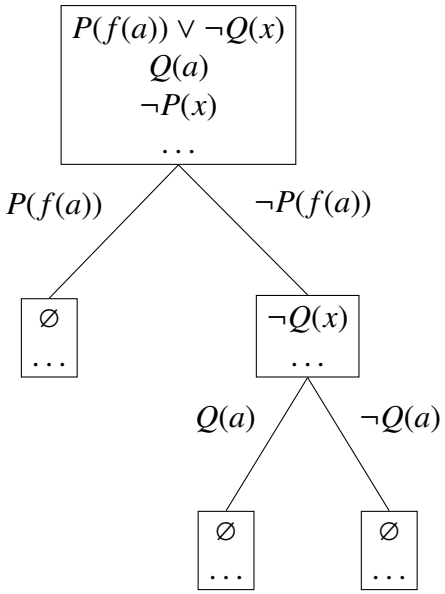
Heuristika ASH2 traži tri klauzule od kojih su dvije unarne, a jedna se sastoji od dva literala. Iz ovakvog se skupa dedukcijom može dobiti prazna klauzula ako prvo dobijemo rezolventu jedne unarne klauzule i jedine s dva literala, a zatim rezolventu dobivene rezolvente i preostale unarne klauzule. Jedno dijete će odmah dati praznu klauzulu dok će djeca drugog djeteta u svojim skupovima klauzula sadržavati praznu klauzulu.

Primjer 2.3.4. *Recimo da se nalazimo u čvoru N te da je program HERBY u* clist *uspio pronaći tročlani skup klauzula* $S = \{\{P(f(a)), \neg Q(x)\}, \{Q(a)\}, \{\neg P(x)\}\}$ *isti kao iz primjera 2.3.2. Promotrimo dedukciju* $C_1, \ldots, C_5$ *raspisanu u istom primjeru. Ovdje će program HERBY koristiti heuristiku ASH2.*

Lako je vidjeti da će iz $C_5 = \emptyset = Res(\{\neg P(x)\}, \{P(f(a))\})$ *opisanim postupkom program HERBY kao prvi odabrani atom uzeti* $P(f(a))$*. Isto tako je lako vidjeti iz* $C_4 = P(f(a)) = Res(\{P(f(a)), \neg Q(x)\}, \{Q(a)\})$ *da će sljedeći odabrani atom biti* $Q(a)$*.*

Na slici 2.1 vidimo kako uz odabrane atome izgleda podstablo s korijenom u N. Možemo vidjeti da čak i ne znajući koje se ostale klauzule nalaze u clist *ovakvim odabirom atoma uspješno zatvaramo podstablo s korijenom u N.*

Heuristika ASH3 traži četveročlani skup klauzula iz kojeg je dedukcijom moguće dobiti praznu klauzulu. Traženi se skup mora sastojati od tri unarne klauzule i jedne klauzule koja



Slika 2.1: Podstablo s korijenom u N iz primjera 2.3.4.

sadrži tri literala. Ako označimo klauzule sa C_1, C_2, C_3, C_4 gdje je C_4 tročlana klauzula, dedukcija prazne klauzule iz ovog skupa može izgledati ovako:

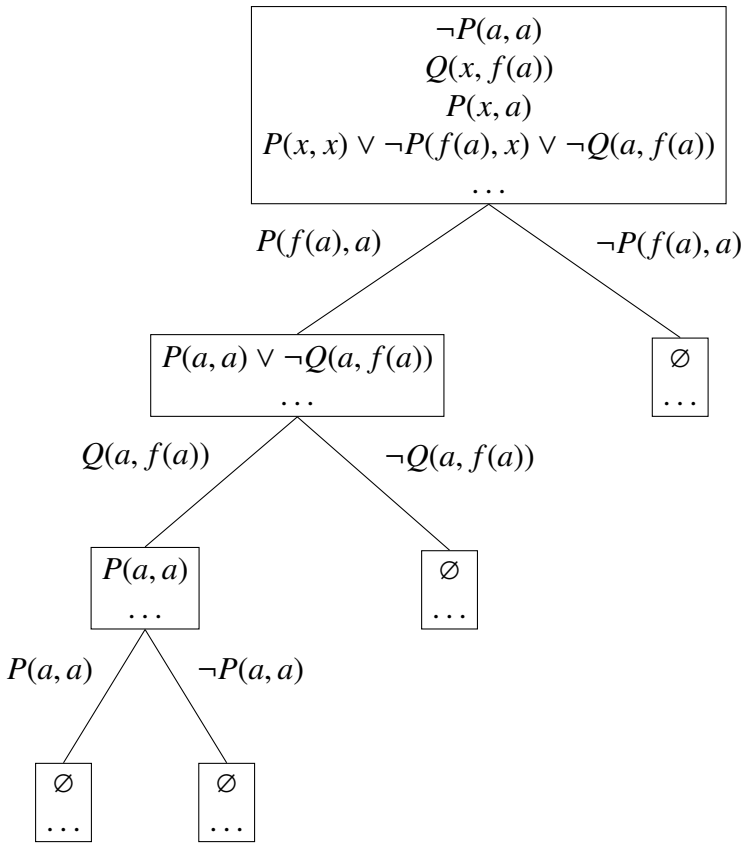
1. C_1
2. C_2
3. C_3
4. C_4
5. $C_5 = Res(C_1, C_4)$
6. $C_6 = Res(C_2, C_5)$
7. $C_7 = Res(C_3, C_6) = \varnothing$

Ako odaberemo atome ovom heuristikom jedno će dijete sigurno sadržavati praznu klauzulu. Drugo će dijete imati dvoje djece od kojih će jedno sadržavati praznu klauzulu, a drugo imati dvoje djece koja će oba sadržavati praznu klauzulu.

Primjer 2.3.5. *Pretpostavimo da se nalazimo u čvoru N i da se u* clist *nalaze klauzule* $C_1 = \{\neg P(a,a)\}, C_2 = \{Q(x, f(a))\}, C_3 = \{P(x,a)\}, C_4 = \{P(x,x), \neg P(f(a),x), \neg Q(a, f(a))\}$ *te ih je heuristika ASH3 pronašla. Dedukcija prazne klauzule iz ovog skupa je sljedeća:*

1. $C_1 = \{\neg P(a,a)\}$
2. $C_2 = \{Q(x, f(a))\}$
3. $C_3 = \{P(x,a)\}$
4. $C_4 = \{P(x,x), \neg P(f(a),x), \neg Q(a, f(a))\}$
5. $C_5 = Res(C_1, C_4) = \{\neg P(f(a),a), \neg Q(a, f(a))\}$
6. $C_6 = Res(C_2, C_5) = \{\neg P(f(a),a)\}$
7. $C_7 = Res(C_3, C_6) = \varnothing$

Odabrani atomi bit će redom: $P(f(a),a)$*,* $Q(a, f(a))$ *i* $P(a,a)$*. Sada možemo skicirati podstablo s korijenom u čvoru N i vidjeti da će ovakvim izborom atoma stvarno biti zatvoreno. Dobiveno stablo je na slici 2.2.*



Slika 2.2: Podstablo s korijenom u N iz primjera 2.3.5.

Heuristika ASH4 generira sve unarne klauzule koje je moguće dobiti kao rezolvente iz dvočlanog ili tročlanog skupa klauzula iz *clist*. Ako je pronađeni skup klauzula dvočlan, tada jedna njegova klauzula mora biti unarna, a druga imati dva literala. Ako je skup tročlan, tada dvije klauzule moraju biti unarne, a jedna imati tri literala. Za ovu se heuristiku koristi posebna lista *unit_list* u kojoj se čuvaju sve dosad generirane unarne klauzule. Kada je generirana nova unarna klauzula U, tražimo rezolvente te novodobivene klauzule i svih unarnih klauzula već u *unit_list*. Ako $U = \{L_1\}$ u paru s drugom unarnom klauzulom $V = \{L_2\}$ daje praznu klauzulu kao rezovlentu, onda znamo da postoji najopćenitiji unifikator μ od L_1 i $\neg L_2$. Tada za daljnu konstrukciju stabla odabiremo $L_1\mu$ kao i sve literale korištene u dedukciji od U.

Ova heuristika ne garantira odmah zatvaranje podstabla jer se unarne klauzule U i V mogu nalaziti na različitim putevima. Međutim, ako su sve klauzule korištene u dedukciji U i V unutar *clist*, onda će se podstablo sigurno zatvoriti što se i događa u većini slučajeva.

Primjer 2.3.6. *Neka se u* clist *nalaze klauzule* $\{\neg Q(a)\}$, $\{Q(x), \neg P(f(y))\}$, $\{Q(f(a))\}$, $\{\neg R(a)\}$ *i* $\{P(x), \neg Q(x), R(a)\}$. *Neka je heuristika ASH4 već ranije u* unit_list *stavila unarnu klauzulu* $V = \{P(f(a))\}$ *dobivenu sljedećom dedukcijom.*

1. $C_1 = \{Q(f(a))\}$
2. $C_2 = \{\neg R(a)\}$
3. $C_3 = \{P(x), \neg Q(x), R(a)\}$
4. $C_4 = Res(C_1, C_3) = \{P(f(a)), R(a)\}$
5. $C_5 = V = Res(C_2, C_4) = \{P(f(a))\}$

Sada klauzulu U dobivamo kao $Res(\{\neg Q(a)\}, \{Q(x), \neg P(f(y))\}) = \{\neg P(f(y))\}$. *Lako je vidjeti da vrijedi* $Res(U, V) = \varnothing$. *Dakle, bit će odabran atom* $P(f(a))$ *te* $Q(a)$.

Postoje i druge heuristike za izbor atoma, ali se ne koriste tako često pa ih nećemo posebno obrađivati. Primjerice, heuristika ASH5 je "očajnički pokušaj" pronalaženja atoma ako su prve četiri heuristike podbacile. Ova heuristika generira atome pomoću klauzula iz *clist* i *unit_list*. Ima tri podheuristike koje nasumično alternira. Dvije podheuristike traže atom na *clist*, a jedna na *unit_list*. Posljednja heuristika za izbor atoma je ASH0 koja se koristi samo onda kada korisnik uključi tu opciju. U ovoj heuristici zapravo sam korisnik bira sljedeći atom.

Osim heuristika za izbor atoma program HERBY koristi i neke dodatne heuristike. Već smo spomenuli heuristiku za izbor konstanti, a tu su još i heuristike za uređivanje listi i druge. Nama će najvažnija od ovih pomoćnih heuristika biti BCR (*base clause resolution*) heuristika ili Faza 0. Ova heuristika se pokreće odmah na početku dokazivanja teorema, prije konstrukcije semantičkog stabla, odakle i naziv faza 0. Program HERBY koristi BCR heuristiku kako bi proširio skup baznih klauzula. Heuristika prvo traži rezolvente posljednje bazne klauzule u *clist* i svih ostalih baznih klauzula te dodaje dobivene rezolvente u prošireni skup baznih klauzula. Zatim traži rezolvente posljednje i pretposljednje bazne klauzule sa svim klauzuluma u proširenom skupu pa i njih dodaje. Postupak se nastavlja dok se ne prijeđe granica od 25 klauzula. Ova heuristika čini heuristike za izbor atoma učinkovitijima, a time i povećava efikasnost konstrukcije zatvorenog semantičkog stabla.

Poglavlje 3

Primjeri korištenja

U ovom ćemo poglavlju pokreniti program HERBY. Prvo ćemo proći kroz sve opcije dostupne korisniku, a zatim ćemo dokazati dva teorema kao primjere. Uključit ćemo opciju detaljnog ispisa procesa dokazivanja te pratiti sve odabrane atome i korištene heuristike. Na kraju ćemo konstruirati zatvoreno semantičko stablo pomoću kojeg je program HERBY dokazao teorem.

3.1 Opcije dostupne korisniku

Odmah nakon pokretanja programa HERBY, korisnik ima dvije opcije: ili upisati ime teorema ili *?* za pomoć. Upisivanjem upitnika te prihvaćanjem dobiva se sljedeći ispis.

```
-----------------------------------------------------
| Enter name of the theorem and then options as below:
|
| photon:ths:stark103 t200 d2
|
| where:
|    t200 denotes a maximum of 200 seconds
|    d2   denotes a debug level of 2
|    r1   find a res-ref proof also
|    f1   complete iteration before beginning next
|    k0   skip initial simplification
|    xX   limit characters in resolvent to X
|    lX   limit literals in resolvent to X
|    i1   select atoms interactively
|    tptp theorem is in tptp format
```

```
|    batch prove a batch of theorems
-------------------------------------------------
```

Ako korisnik samo upiše ime teorema bez dodatnih opcija, bit će postavljena uobičajena ograničenja od 3600 sekundi i dubine stabla od 200 razina. Ako program HERBY uspije dokazati teorem, ispisat će se prigodna poruka, svi atomi potrebni za konstrukciju zatvorenog semantičkog stabla, vrijeme potrebno za dokazivanje i broj čvorova u stablu.

Sve opcije opisane su u prethodnom ispisu, a za nas će najbitnija biti druga opcija *d2* pomoću koje ćemo dobiti detaljno raspisan proces konstrukcije semantičkog stabla. Ispisivat će nam se kad je točno koji atom odabran, prema kojem čvoru vodi te kako se u određenom čvoru dobije prazna klauzula.

3.2 Prvi primjer

Kao prvi primjer uzet ćemo teorem spremljen u datoteku *STARK075.THM*. Iskaz teorema slijedi.

Teorem 3.2.1. *Za niti jedan $n \in \mathbb{N}$ ne vrijedi $n < n$.*

Budući da sam teorem nema pretpostavki, već samo zaključak, programu HERBY morat ćemo dati samo relevantne Peanove aksiome pretvorene u potrebni kauzalni oblik. Zaključak ćemo negirati te također pretvoriti u klauzulu. Dobiva se sljedeći skup klauzula, odnosno sadržaj datoteke *STARK075.THM*. Iza svakog dijela navodimo aksiome ekvivalentne ispisanim klauzulama u standardnom obliku.

```
1  ~Equal(S(x),S(y)) | Equal(x,y)
2  ~Equal(S(x),0())
3  ~Equal(x,y) | ~Equal(x,z) | Equal(y,z)
4  ~Equal(x,y) | Equal(S(x),S(y))
5  Equal(P(x,0()),x)
6  Equal(P(x,S(y)),S(P(x,y)))
7  Equal(M(x,0()),0())
8  Equal(M(x,S(y)),P(M(x,y),x))
```

Upravo navedeni skup klauzula predstavlja sljedeće Peanove aksiome (u zagradama pišemo retke na koje se određeni aksiom odnosi u prethodnom ispisu):

- $\forall x, y(x = y \iff S(x) = S(y))$ (1, 4)
- $\forall x(S(x) \neq 0)$ (2)

- $\forall x, y, z(x = y \wedge x = z \implies y = z)$ (3)
- $\forall x(x + 0 = x \wedge x \cdot 0 = 0)$ (5,7)
- $\forall x, y(x + S(y) = S(x + y))$ (6)
- $\forall x, y(x \cdot S(y) = x \cdot y + x)$ (8)

```
9  Equal(x,x)
10 ~Equal(x,y) | Equal(y,x)
11 ~Equal(x,y) | ~Equal(y,z) | Equal(x,z)
```

Ovim skupom klauzula osiguravamo da je relacija jednakosti jedna relacija ekvivalencije. Točnije, ovi aksiomi redom iskazuju svojstva refleksivnosti, simetričnosti i tranzitivnosti relacije jednakosti (u zagradama opet navodimo na koje prethodno navedene retke se pojedini aksiom odnosi):

- $\forall x(x = x)$ (9)
- $\forall x, y(x = y \implies y = x)$ (10)
- $\forall x, y, z(x = y \wedge y = z \implies x = z)$ (11)

```
12 Equal(P(S(F(x,y)),x),y) | ~L(x,y)
13 ~Equal(P(S(x),y),z) | L(y,z)
14 Equal(P(x,y),P(y,x))
15 ~Equal(P(x,y),P(z,y)) | Equal(x,z)
```

Kao i obično, navodimo standardni zapis prethodno navedenog skupa klauzula:

- $\forall x, y(x < y \implies \exists z(S(z) + x = y))$ (12)
- $\forall x, y, z(S(x) + y = z \implies y < z)$ (13)
- $\forall x, y(x + y = y + x)$ (14)
- $\forall x, y, z(x + y = z + y \implies x = z)$ (15)

```
16 negated_conclusion
17 L(A(),A())
```

Ovo posljednje je negirani traženi zaključak. Budući da tražena tvrdnja glasi $\forall x \neg(x < x)$ tada je njezina negacija sljedećeg oblika: $\exists x(x < x)$. Ako to prevedmo u klauzulu u jeziku progrma HERBY dobivamo: `L(A(),A())` (zbog upotrebe Skolemove konstante pri uklanjanju egzistencijalnog kvantifikatora).

Kao što vidimo, prije negiranog zaključka potrebno je upisati oznaku *negated_conclusion*. Objasnimo klauzule i značenje svih simbola. U ovom skupu klauzula pojavljuju se relacijski simboli `Equal` i `L`. `Equal` očito predstavlja relaciju jednakosti, dok `L` predstavlja relaciju uređaja. Dakle, formulu `L(x, y)` interpretiramo sa $x < y$. Konstantski simboli su `A` i `0`, a funkcijski su `P`, `M`, `F` i `S`. Funkcijski simbol `P` predstavlja zbrajanje, `M` množenje, a `S` funkciju sljedbenika. Simbol `F` je Skolemova funkcija uvedena pri pretvaranju aksioma $\forall x, y(x < y \implies \exists z(x + S(z) = y))$ u ekvivalentnu klauzulu `Equal(P(S(F(x,y)),x),y) | ~L(x,y)` (primijetimo da je Skolemova funkcija `F` argument funkcijskog simbola `S` kako bi bili sigurni da je element koji dodajemo `x` veći od `0`).

Ako sad pokrenemo program HERBY i na ulaz mu damo ime *STARK075.THM* (zajedno s putanjom do datoteke ako je potrebno) bez dodatnih opcija, početak ispisa izgledat će ovako:

```
Predicates: Equal L
Functions:  A . 0 : P M F S
EQ: 9.10.3
.11
ESAF: 4
ESAP:

NEXTC=43 TIME=3600 XARS=35
  1: ~EqualAP0A   H4a
  2: EqualPSFAAAA   -H4
  3: ~EqualPSFAAAP0A   -H4  T0  N1  C0  U78
  4: ~EqualPPA00PA0   H1  T0  N11  C43  U78

TIME:0 (NODES: 13, ATOMS: 4)
** Proof Found! **
```

U prvom reda iza oznake `Predicates:` ispisani su svi relacijski simboli. U drugom redu nakon `Functions:`, a prije točke ispisane su svi konstantni simboli koji se pojavljuju u negiranom zaključku (u ovom slučaju samo `A`), a poslije točke i prije dvotočke svi oni koji se pojavljuju u aksiomima (ovdje `0`). Nakon dvotočke ispisuju se svi funkcijski simboli koji se pojavljuju u čitavom skupu klauzula danom kao ulaz.

`NEXTC=43` označava da smo koristili prošireni skup baznih klauzula koji se sastoji od 43 klauzule. Zatim, `TIME=3600` označava da je vremensko ograničenje 3600 sekundi,

a `XARS=35` označava da je maksimalan broj terma u literalu ograničen na 35. Nakon toga ispisana su četiri odabrana atoma korištena za konstrukciju zatvorenog semantičkog stabla.

Pogledajmo sad popis odabranih atoma. Treba naglasiti da program HERBY ispisuje klauzule bez zagrada ili znaka |. Prva tri atoma odabrana su zajedno koristeći četvrtu heuristiku što vidimo iz oznaka `H4a` i `-H4`. Dakle, program HERBY je pronašao tri klauzule iz kojih se dedukcijom može dobiti prazna klauzula. U trećem retku oznaka `T0` označava da mu je za to trebalo nula sekundi, `N1` da je zasad konstruiran samo jedan čvor (korijen), `C0` da se u listi *clist* ne nalazi niti jedna nebazna klauzula (klauzula koja se ne nalazi u početnom skupu klauzula), a `U78` da se u listi unarnih klauzula *unit_list* nalazi 78 klauzula. Četvrti je odabrani atom odabran koristeći prvu heuristiku nakon što je već konstruirano 11 čvorova te je *clist* sadržavao 43 nebazne klauzule.

Pokrenimo sada dokaz istog teorema s uključenom opcijom `d2` te pratimo konstrukciju zatvorenog semantičkog stabla. Nakon istog početka, program HERBY ispisuje sve klauzule dodane u *clist* pomoću BCR heuristike.

```
27 clauses are added by BCR Heuristic:

 17: (16a,12b) EqualPSFAAAA
 18: (15b,13a) ~EqualPPSxyzPuz Lyu
 19: (15a,12a) EqualSFxPyxy ~LxPyx
 ...
 41: (14a,3b)  ~EqualPxyz EqualzPyx
 42: (15a,41b) Equalxy ~EqualPzyPxz
 43: (15b,41a) ~EqualPPxyzPuz EqualuPyx
```

U ovom smo slučaju u početni skup klauzula dodali novih 27 klauzula. Na ovaj način proširujemo skup klauzula s kojim radimo i činimo konstrukciju semantičkog stabla jednostavnijim. Nakon ovoga ispisuje se generiranje već spomenuta tri atoma pomoću četvrte heuristike.

```
GENERATE ATOM USING ASH4b:
C1,C2 resolve to C4, which resolves with C3

C1:  17: (16a,12b) EqualPSFAAAA
C2:  11: ~Equalxy ~Equalyz Equalxz
C3:  23: (15b,2a)  ~EqualPSxyP0y
C4:  44: (17a,11a) ~EqualAx EqualPSFAAAx
The atom generated is: 45: (23a,11b) ~EqualAP0A

  1: ~EqualAP0A   H4a
```

```
  2: EqualPSFAAAA   -H4
  3: ~EqualPSFAAAP0A   -H4  T1  N1  C0  U78
```

Ovdje su ispisane bazne klauzule C1, C2 i C3 s time da je C2 iz originalnog skupa klauzula dok su C3 i C4 iz proširenog skupa baznih klauzula. Klauzula C4 dobivena je rezolucijom klauzula C2 i C1. Pomoću zapisa u zagradama, na primjer kod klauzule C1, možemo zaključiti da je ista dobivena kao rezolventa klauzula označenih s brojevima 16 i 12. Slova a i b označavaju poziciju komplementarnih literala unutar klauzula. Dobivamo $Res(\{L(A,A)\},\{Equal(P(S(F(x,y)),x),y),\neg L(x,y)\}) = \{Equal(P(S(F(A,A)),A),A)\}$.

Sukladno heuristici 4, dobili smo na kraju unarnu klauzulu EqualAP0A koja u paru s nekom klauzulom iz *unit_list* daje praznu klauzulu kao rezolventu. Zatim su popisana sva tri odabrana atoma proizašla iz ove heuristike. Sad napokon kreće konstrukcija stabla. Na lijevom bridu program HERBY negira odabrani atom, a na desnom bira nenegirani.

```
Branch on atom:   1: EqualAP0A to node 1

GENERATE CLAUSES AT NODE 1
 44: (1a,3a)   ~EqualAP0A#1  ~EqualAx EqualP0Ax
 45: (1a,3b)   ~EqualAx ~EqualAP0A#1  EqualxP0A
 46: (1a,4a)   ~EqualAP0A#1  EqualSASP0A
 47: (1a,10a)  ~EqualAP0A#1  EqualP0AA
 48: (1a,11a)  ~EqualAP0A#1  ~EqualP0Ax EqualAx
 49: (1a,11b)  ~EqualxA ~EqualAP0A#1  EqualxP0A
 50: (1a,39a)  ~EqualAP0A#1  EqualAPA0
 51: (1a,3b)   ~EqualAP0A#1  ~EqualAP0A#1  EqualP0AP0A
GENERATED 8 CLAUSES

PATH: 1
```

Program HERBY kreće s grananjem prema čvoru s oznakom 1 te ispisuje sve generirane klauzule na tom čvoru. Naravno da sve dobivene klauzule moraju biti rezolvente klauzule EqualAP0A te neke druge klauzule iz baznog skupa zato je prvi element u zagradi uvijek 1a. Nakon toga ispisuje se put u stablu, to jest PATH: 1. Ovdje 1 označava grananje na lijevo, a 2 na desno. Budući da se u čvoru 1 ne pojavljuje prazna klauzula, nastavljamo s konstrukcijom stabla grananjem na istom čvoru.

```
Branch on atom:   2: ~EqualPSFAAAA to node 2

GENERATE CLAUSES AT NODE 2

PATH: 11 FAILS:  58: (2a,17a)  EqualPSFAAAA#2
```

Kada se dobije prazna klauzula, nije potrebno generirati ostale klauzule. Ovdje se na putu 11 dobila prazna klauzula kao rezolventa odabranog atoma ~EqualPSFAAAA i klauzule EqualPSFAAAA. Taj je put u stablu zatvoren te se vraćamo na višu razinu i nastavljamo s grananjem na put 12.

```
Branch on atom:   2: EqualPSFAAAA to node 3

GENERATE CLAUSES AT NODE 3
 52: (2a,3a)   ~EqualPSFAAAA#2  ~EqualPSFAAAx EqualAx
 53: (2a,3b)   ~EqualPSFAAAx ~EqualPSFAAAA#2  EqualxA
 54: (2a,4a)   ~EqualPSFAAAA#2  EqualSPSFAAASA
 55: (2a,10a)  ~EqualPSFAAAA#2  EqualAPSFAAA
 56: (2a,11a)  ~EqualPSFAAAA#2  ~EqualAx EqualPSFAAAx
 57: (2a,11b)  ~EqualxPSFAAA ~EqualPSFAAAA#2  EqualxA
 58: (2a,38a)  ~EqualPSFAAAA#2  EqualPASFAAA
 59: (2a,41a)  ~EqualPSFAAAA#2  EqualAPASFAA
 60: (2a,11a)  ~EqualPSFAAAA#2  ~EqualAP0A#1  EqualPSFAAAP0A
 61: (2a,3b)   ~EqualPSFAAAA#2  ~EqualPSFAAAA#2  EqualAA
GENERATED 10 CLAUSES

PATH: 12
```

Čvor 3 na putu 12 generirao je 10 klauzula, ali ne i praznu pa ćemo morati iskoristiti i posljednji odabrani atom kako bi pokušali zatvoriti lijevo podstablo.

```
Branch on atom:   3: EqualPSFAAAP0A to node 4

GENERATE CLAUSES AT NODE 4

PATH: 121 FAILS:  72: (3a,23a)  ~EqualPSFAAAP0A#3

Branch on atom:   3: ~EqualPSFAAAP0A to node 5

GENERATE CLAUSES AT NODE 5

PATH: 122 FAILS:  82: (3a,11c)  ~EqualPSFAAAA#2  ~EqualAP0A#1
        EqualPSFAAAP0A#3

PATH: 12 FAILS:
```

```
PATH: 1 FAILS:
```

Sada je prazna klauzula dobivena na oba čvora 4 i 5 te se program HERBY vraća unatrag i zaključuje da je lijevo podstablo (u odnosu na korijen) zatvoreno. Konstrukcija desnog podstabla odvija se na sličan način pa preskačemo čitavi ispis. Ključna je razlika u tome što dolaskom do čvora 10 na dubini 3 ne dobivamo praznu klauzulu, a sva su tri odabrana atoma potrošena. To znači da program HERBY mora posegnuti za još jednom heuristikom kako bi dobio novi odabrani atom i zatvorio stablo.

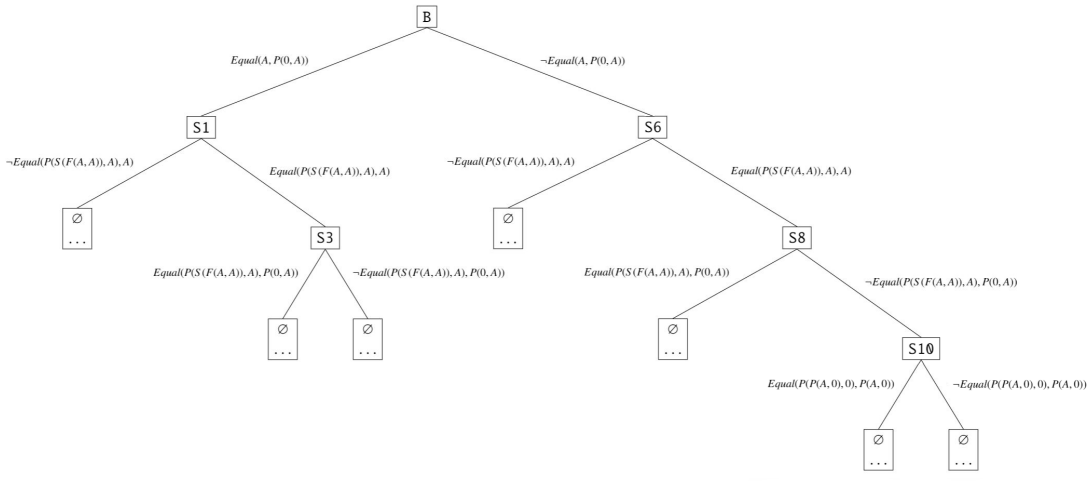
```
PATH: 222

GENERATE ATOM USING ASH1:
C1 and C2 resolve to the NULL clause

C1:   59: (1a,43b)  ~EqualPPA0xPAx EqualAP0A#1
C2:    5: EqualPx0x
The atom generated is: 87: ~EqualPPA00PA0

  4: ~EqualPPA00PA0   H1  T8  N11  C43  U78
```

Na putu 222 pomoću prve heuristike odabiremo atom `EqualPPA00PA0` te nastavljamo s konstrukcijom.

```
Branch on atom:   4: EqualPPA00PA0 to node 11

GENERATE CLAUSES AT NODE 11

PATH: 2221 FAILS: 100: (4a,43a)  ~EqualPPA00PA0#4  EqualAP0A#1

Branch on atom:   4: ~EqualPPA00PA0 to node 12

GENERATE CLAUSES AT NODE 12

PATH: 2222 FAILS:  89: (4a,5a)   EqualPPA00PA0#4

PATH: 222 FAILS:

PATH: 22 FAILS:

PATH: 2 FAILS:
```

```
PATH:  FAILS:

TIME:187 (NODES: 13, ATOMS: 4)
** Proof Found! **
```

Jednom kad smo zaključili da je prazni put zatvoren, zapravo smo zatvorili korijen, a time i cijelo stablo te dokazali teorem. Nakon toga se ispisuje čitavi *unit_list* te *clist* te je proces gotov. Sada možemo skicirati zatvoreno semantičko stablo dobiveno ovom konstrukcijom. Neka ovdje `B` označava skup baznih klauzula (unesene klauzule te proširenje početnom heuristikom). Oznake `S1`, `S3`, `S6`, `S8` i `S10` označavaju skupove klauzula dobivene na čvorovima označenim tim brojevima. Na slici 3.1 je prikazano stablo.



Slika 3.1: Konstruirano zatvoreno semantičko stablo

3.3 Drugi primjer

Za drugi primjer uzet ćemo teorem iz teorije grupa. Za početak nam je potrebna definicija grupe, odnosno tri aksioma koja ćemo morati pretvoriti u klauzule i dati programu HERBY.

Definicija 3.3.1. Grupa *je uređeni par* $(S, \cdot)$ *gdje je* S *skup,* $a \cdot : S \times S \to S$ *binarna operacija takva da vrijedi:*

- *(Neutralni element) Postoji* $e \in S$ *takav da za svaki* $f \in S$ *vrijedi* $e \cdot f = f \cdot e = f$.
- *(Inverz) Za svaki* $f \in S$ *postoji* $g \in S$ *takav da* $f \cdot g = g \cdot f = e$.
- *(Asocijativnost) Za sve* $f, g, h \in S$ *vrijedi* $(f \cdot g) \cdot h = f \cdot (g \cdot h)$.

Inverz elementa grupe x ćemo zbog lakšeg razumijevanja označavati s $I(x)$ jer je na taj način označen u ulaznoj datoteci. Slijedi iskaz teorema kojeg ćemo dokazivati korištenjem programa HERBY,

Teorem 3.3.2. *Ako je* S *neprazan podskup od* G*, gdje je* $(G, \cdot)$ *grupa, takav da za svaki* $x, y \in S$ *vrijedi* $(x \cdot I(y)) \in S$*, onda vrijedi i* $e \in S$.

Teorem pretvoren u potrebni oblik nalazi se u datoteci CHANG5.THM. Navodimo sadržaj te datoteke s time da nakon svakog ispisanog dijela objašnjavamo navedene klauzule i pišemo ih u obliku formule logike prvog reda prije pretvorbe u klauzulu. Također ćemo objašnjavati sve simbole koji se pojavljuju.

```
P(e(),x,x)
P(x,e(),x)
P(x,I(x),e())
P(I(x),x,e())
S(a())
```

Očito konstantni simbol `e()` označava neutralni element. Funkcijski simbol `I(x)` označava inverz od x. Relacijski simbol `P` predstavlja binarnu operaciju $\cdot$, tj. $P(x, y, z)$ vrijedi ako i samo ako $x \cdot y = z$. Prve dvije klauzule iskazuju svojstva neutralnog elementa `e()`. Točnije, klauzula `P(e(),x,x)` je zapravo drugi zapis formule $e \cdot x = x$, a `P(x,e(),x)` predstavlja formulu $x \cdot e = x$.

Sljedeće dvije klauzule iskazuju svojstvo inverza. Dobivene su iz sljedeće formule:

$$\forall x \exists g (P(x, g, e) \wedge P(g, x, e)).$$

Prethodna formula zapravo tvrdi da za svaki element x postoji neki element g tako da vrijedi $f \cdot g = g \cdot f = e$.

Posljednja klauzula u ovom dijelu uvodi novi relacijski simbol S, pri čemu imamo da vrijedi $S(x)$ ako i samo ako $x \in S$. Klauzula `S(a())` dobivena je uvođenjem Skolemove konstante pri pretvaranju formule $\exists x S(x)$ u klauzulu. Ovo je pretpostavka o nepraznosti skupa S iz iskaza teorema.

```
~S(x) | ~S(y) | ~P(x,I(y),z) | S(z)
```

Ova je klauzula još jedna pretpostavka teorema. Napisana kao formula logike prvog reda prije pretvorbe u klauzulu glasi:

$$\forall x, y, z((S(x) \wedge S(y) \wedge P(x, I(y), z)) \rightarrow S(z)).$$

Prethodna formula očito tvrdi da za sve $x, y \in S$ vrijedi $(x \cdot I(y)) \in S$.

```
~P(x,y,u) | ~P(y,z,v) | ~P(x,v,w) | P(u,z,w)
~P(x,y,u) | ~P(y,z,v) | ~P(u,z,w) | P(x,v,w)
```

Ove dvije klauzule iskazuju svojstvo asocijativnosti grupovne operacije. Radi ilustracije promotrimo detaljnije samo prvu klauzulu. Formula logike prvog reda prije pretvorbe u klauzulu izgleda ovako:

$$\forall x, y, u, z, v, w((P(x, y, u) \wedge P(y, z, v) \wedge P(x, v, w)) \rightarrow P(u, z, w)).$$

Dakle, ako vrijedi $x \cdot y = u$, $y \cdot z = v$ i $x \cdot v = w$, onda vrijedi i $u \cdot z = w$. Koristeći zagrade ovo posljednje možemo zapisati i ovako: $x \cdot (y \cdot z) = (x \cdot y) \cdot z$, čime je iskazano svojstvo asocijativnosti grupovne operacije. Sasvim na isti način mogli bi analizirati i drugu klauzulu.

```
negated_conclusion
~S(e())
```

Ostao je još negirani zaključak koji tvrdi da neutralni element e nije element skupa S što je upravo ova posljednje klauzula. Sada smo spremni dokazati teorem pomoću programa HERBY. Odmah ćemo uključiti opciju `d2` i pratiti konstrukciju stabla.

```
Predicates: P S
Functions:  e . a : I
ESAF:
ESAP:

27 clauses are added by BCR Heuristic:
```

```
 10: (9a,6d)   ~Sx ~Sy ~PxIye
 11: (8a,4a)   ~Pxyz ~Peyu PIxzu
 ...
 35: (6a,5a)   ~Sx ~PaIxy Sy
 36: (6b,5a)   ~Sx ~PxIay Sy

 NEXTC=36 TIME=3600 XARS=20
```

Na početku opet dobivamo popis relacijskih, konstantskih i funkcijskih simbola koji smo već analizirali. Vidimo da i ovaj put BCR heuristika dodaje novih 27 klauzula u skup baznih klauzula. Vremenska ograničenja i ograničenja na broj terma u literalu ostaju ista kao i za prvi primjer.

```
GENERATE ATOM USING ASH3:
C1,C2 resolve to C5, which resolves with C3 to C6
C4 and C6 resolve

C1:   5: Sa
C2:  36: (6b,5a)   ~Sx ~PxIay Sy
C3:   3: PxIxe
C4:   9: ~Se
C5:  37: (5a,36a)  ~PaIax Sx
C6:  38: (37a,3a)  Se
The atom generated is: 39: Se

  1: Se   H3
  2: Sa   -H3
  3: PaIae   -H3  T0  N1  C0  U0
```

Program HERBY ovaj put koristi heuristiku ASH3 za izbor atoma. To jest, pronašli smo tri klauzule iz kojih se dedukcijom može dobiti prazna klauzula. Dedukcija je napisana, a nakon nje i odabrani atomi. Primijetimo da će ovaj put lista *unit_list* ostati prazna jer nismo koristili heuristiku ASH4 što je i vidljivo u zadnjem redu gdje je `U0`.

```
Branch on atom:   1: ~Se to node 1

GENERATE CLAUSES AT NODE 1
 37: (1a,35c)  ~Sx ~PaIxe Se#1
 38: (1a,36c)  ~Sx ~PxIae Se#1
GENERATED 2 CLAUSES
```

```
PATH: 1
```

Sada počinjemo konstrukciju stabla. Na čvoru 1 koji je lijevo dijete korijena dobivamo dvije nove klauzule, ali ne i praznu klauzulu što znači da se konstrukcija nastavlja. Vidimo da smo nove klauzule dobili kao rezolvente odabranog atoma i posljednjih dvaju klauzula u proširenom skupu baznih klauzula. Klauzulu označenu brojem 37 dobili smo kao $Res(\{\neg S(e)\}, \{\neg S(x), \neg P(a, I(x), y), S(y)\}) = \{\neg S(x), \neg P(a, I(x), e)\}$, a klauzulu označenu brojem 38 kao $Res(\{\neg S(e)\}, \{\neg S(x), \neg P(x, I(a), y), S(y)\}) = \{\neg S(x), \neg P(x, I(a), e)\}$. Nastavljamo dalje s čvorem 2 koji je lijevo dijete od čvora 1.

```
Branch on atom:   2: ~Sa to node 2

GENERATE CLAUSES AT NODE 2

PATH: 11 FAILS:  39: (2a,5a)   Sa#2
```

Ovdje je izabrani atom $\neg S(a)$ što odmah daje praznu klauzulu kao rezolventu u paru s petom klauzulom iz ulazne datoteke $S(a)$. Dakle zatvorili smo put 11 te nastavljamo konstrukciju na putu 12.

```
Branch on atom:   2: Sa to node 3

GENERATE CLAUSES AT NODE 3
 39: (2a,35a)  ~Sa#2  ~PaIax Sx
 40: (2a,35a)  ~Sa#2  ~PaIae Se#1
GENERATED 2 CLAUSES

PATH: 12
```

Ovdje smo opet generirali nove dvije klauzule. Jedna je dobivena u rezolventi opet s predzadnjom klauzulom baznog skupa, a druga s klauzulom iz skupa dobivenih klauzula na čvoru 1 označenom brojem 38. Konstrukcija se nastavlja.

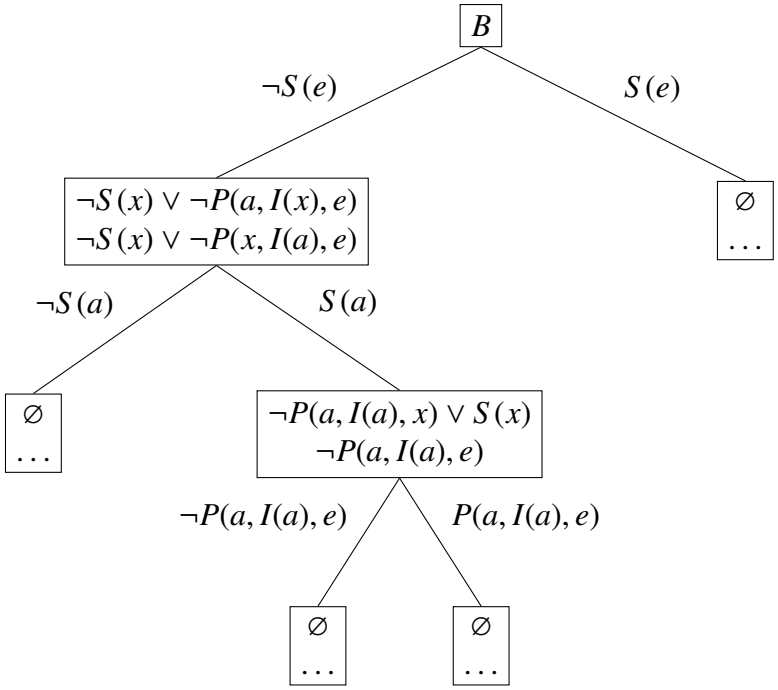
```
Branch on atom:   3: ~PaIae to node 4

GENERATE CLAUSES AT NODE 4

PATH: 121 FAILS:  41: (3a,3a)   PaIae#3
```

```
Branch on atom:   3: PaIae to node 5

GENERATE CLAUSES AT NODE 5

PATH: 122 FAILS:  77: (3a,35b)  ~Sa#2  ~PaIae#3  Se#1

PATH: 12 FAILS:

PATH: 1 FAILS:
```

Ovdje zatvaramo podstablo s korijenom u čvoru 1. U lijevom djetetu odabrani atom je $\neg P(a, I(a), e)$ što s trećim atomom ulazne datoteke $P(x, I(x), e)$ daje praznu klauzulu kao rezolventu. Na desnom djetetu odabrani atom je $P(a, I(a), e)$ koji daje praznu klauzulu kao rezolventu u paru s klauzulom označenom brojem 40, $\neg P(a, I(a), e)$, dobivenom na čvoru 3. Dakle, vraćamo se u korijen i nastavljamo s konstrukcijom na desnom djetetu.

```
Branch on atom:   1: Se to node 6

GENERATE CLAUSES AT NODE 6

PATH: 2 FAILS:  39: (1a,9a)   ~Se#1

PATH:  FAILS:

TIME:121 (NODES: 7, ATOMS: 3)
** Proof Found! **
```

Kao što vidimo, ovdje odmah dobivamo praznu klauzulu u paru s klauzulom iz proširenog skupa baznih klauzula što pokazuje koliko je korisna heuristika BCR. Dokaz je pronađen i možemo prikazati konstruirano zatvoreno semantičko stablo. Ako sa B označimo prošireni skup baznih klauzula, semantičko je stablo prikazano na slici 3.2.



Slika 3.2: Podstablo s korijenom u N iz primjera 2.3.5.

Poglavlje 4

Zaključak

Kao što smo mogli vidjeti, program HERBY uspio je dokazati ova dva jednostavna teorema. Program HERBY daleko je od dokazivača teorema kakav bi jednog dana mogao biti koristan u daljnem razvoju matematike, ali služi kao dobar primjer prepreka s kakvima se danas suočavaju čak i najnapredniji dokazivači. Najveći problem nama kao korisnicima programa HERBY zapravo je odrediti koji će aksiomi biti relevantni u konstrukciji zatvorenog semantičkog stabla, a koji su suvišni. Ako kao ulaz uz pretpostavku teorema i negirani zaključak dajemo premalo aksioma, vjerojatno je da dokaz neće biti pronađen, a ako damo previše, mogao bi biti prekompleksan i premašiti vremensko ograničenje. Treba uzeti u obzir da se svi odabrani aksiomi moraju prvo formalizirati kao formule logike prvog reda, a zatim navedenim postupkom pretvoriti u klauzule.

Cilj matematičara koji se bave dokazivačima teorema danas je pokazati da isti mogu biti korisni na "prvoj crti" matematike. Ako ne u istom obujmu kao matematičari, to jest tako da sami predlažu nove tvrdnje te ih dokazuju, onda barem kao nepogrešivi lektori koji će sa stopostotnom sigurnošću moći potvrditi korektnost nekog dokaza. Tu se susreću s istim preprekama na kakve smo i mi naišli u radu s programom HERBY.

Pogledajmo kao primjer jednog od najpopularnijih modernih dokazivača, Lean. Program Lean se bazira na teoriji tipova i pokazao se uspješnim pri dokazivanju kompliciranih teorema. Naravno, kao i u slučaju programa HERBY, programu Lean ne može se dati samo teorem. On mora na raspolaganju imati sve što mu je potrebno kako bi tvrdnja bila dokazana. Mi smo dokazivali jednostavne tvrdnje za koje je bilo potrebno nekoliko aksioma kao pozadina, ali za dosege na kakve ciljaju korisnici programa Lean potrebna je velika matematička biblioteka iz koje program Lean može crpiti materijale za izgradnju dokaza. Danas se na stotine matematičara bavi upravo izgradnjom te biblioteke, odnosno formalizacijom matematike na način koji je razumljiv programu Lean.

Napredak je vidljiv, no spor. Ipak, uz dosadašnje uspjehe i popuštanje skepticizma oko dokazivača teorema, nije nerazumno pretpostaviti da će nekad u budućnosti isti postati

važan alat svakog matematičara.

Bibliografija

[1] M. Ben-Ari, *Mathematical Logic for Computer Science*, Springer, 2012.

[2] C. L. Chang i R. C. T. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[3] M. Huth i M. Ryan, *Logic in Computer Science: Modelling and Reasoing about Systems*, Cambridge University Press, 2004.

[4] M. Newborn, *Automated Theorem Proving: Theory and Practice*, Springer, 2001.

[5] D. A. Plaisted i Y. Zhu, *The Efficiency of Theorem Proving Strategies*, Springer, 1999.

Sažetak

U ovom radu proučavamo automatski dokazivač teorema HERBY. Najprije se bavimo teorijskom pozadinom njegovog rada i uvodimo pojmove Herbrandove baze i Herbarandovog univerzuma da bi došli do ključnog rezultata, Herbrandovog teorema. Iz jedne verzije Herbrandovog teorema dobivamo način na koji program HERBY dokazuje teoreme, tj. konstrukcijom zatvorenog semantičkog stabla. Zatim proučavamo reprezentaciju logike prvog reda unutar programa HERBY te pretvaranje formula logike prvog reda u klauzule kako bismo mogli programu HERBY teorem zadati u željenom obliku. Opisujemo nekoliko heuristika koje čine konstrukciju zatvorenog semantičkog stabla efikasnijom. Napokon pokrećemo program HERBY te pomoću njega dokazujemo dva teorema. Za svaki teorem skiciramo zatvoreno semantičko stablo pomoću kojeg smo ga dokazali.

Summary

In this paper we present the proof assistant HERBY. Firstly, we take a look into the theoretical basis of its work. We introduce the Herbrand base and the Herbrand universe to arrive at the key conjecture, Herbrand theorem. One version of the theorem serves as an insight into the method HERBY uses to prove theorems, which is by constructing closed semantic trees. We then present a representation of first order logic formulas within HERBY and we study the transformation of first order logic formulas into clauses as they are the form theorems need to take to be used as input. We observe some of the heuristics that are used for efficiency reasons in constructing a closed semantic tree. Finally, we run and test HERBY by proving two theorems. For each of the theorems we offer a sketch of the closed semantic tree used to prove it.

Životopis

Rođen sam 22. listopada 1996. godine u Zagrebu, ali cijeli život živim u Šibeniku gdje sam pohađao osnovnu školu Jurja Šižgorića i osnovnu školu Jurja Dalmatinca. Upisao sam gimnaziju Antuna Vrančića 2011. godine te sam položio maturu 2015. godine nakon čega sam započeo preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu. Godine 2019. završio sam preddiplomski studij te upisao diplomski studij Računarstvo i matematika koji trenutno pohađam.