Univerzitet u Beogradu
Matematički fakultet

Predrag Janičić

Ugradnja procedura odlučivanja u sisteme za automatsko rezonovanje

Building-in decision procedures into systems for automated reasoning

doktorska disertacija/doctoral dissertation

Mentori/Advisors:

prof. Žarko Mijajlović
Matematički fakultet, Univerzitet u Beogradu

prof. Alan Bundy
Division of Informatics, University of Edinburgh

Članovi komisije/Committee members:

prof. Gordana Pavlović–Lažetić
Matematički fakultet, Univerzitet u Beogradu

prof. Duško Vitas
Matematički fakultet, Univerzitet u Beogradu

Beograd, 2000

Posvećeno uspomeni moga oca,
Đorda Janičića (1932–1997)

Dedicated to the memory of my father,
Đorđe Janičić (1932–1997)

Predgovor

Istraživanja u oblasti korišćenja procedura odlučivanja u dokazivačima teorema započeo sam za vreme svog boravka u Edinburgu, u Grupi za matematičko rezonovanje (The Mathematical Reasoning Group, Division of Informatics, University of Edinburgh) tokom akademske 1996/97 godine, kada me je sa tom problematikom upoznao prof. Alan Bundy.

Disertacija sadrži: pregled i analizu najznačnijih shema za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema; opis jedne nove sheme za integrisanje procedura odlučivanja; opis jednog novog, opšteg okvira za kombinovanje i integrisanje procedura odlučivanja; opis implementacija predložene sheme i predloženog okvira i pregled eksperimentalnih rezultata; preglede oblasti odlučivosti, sistema za prezapisivanje i jednakosnih sistema (potrebne za razumevanje ostatka materijala); analizu Prezburgerove aritmetike i nekoliko procedura odlučivanja za tu teoriju. Pod sintagmom "ugradnja procedura odlučivanja" (koja se pojavljuje u naslovu) podrazumevamo i strategije za kombinovanje i strategije za integrisanje procedura odlučivanja u dokazivače teorema; pod sintagmom "sistemi za automatsko rezonovanje" podrazumevamo dokazivače teorema i srodne sisteme.

Deo rezultata iz ove disertacije sadržan je u radovima [60, 59, 58] (koji su objavljeni ili prijavljeni za objavljivanje), a deo u internim izveštajima Grupe za matematičko rezonovanje (u okviru tzv. Blue Book Notes serije).

U radu na doktorskoj disertaciji izuzetnu pomoć pružili su mi mentori prof. Žarko Mijajlović i prof. Alan Bundy i ja im se najtoplije zahvaljujem toj pomoći, na usmeravanju i na dragocenim savetima i sugestijama. Prof. Bundy upoznao me je sa problemom na koji se odnosi ova disertacija i tokom prethodnih godina vodio kroz svet ideja i metodologije automatskog rezonovanja; ja mu se zahvaljujem na velikoj pomoći i svemu što sam od njega naučio. Zahvalnost dugujem i dr Ianu Greenu, mom drugom supervizoru iz Edinburga; njegova pomoć bila je izuzetna a saradnja sa njim je uvek predstavljala zadovoljstvo. Članovi komisije prof. Gordana Pavlović–Lažetić i prof. Duško Vitas svojim dragocenim savetima značajno su mi pomogli u koncipiranju disertacije. Tokom rada na disertaciji, dragocenu pomoć, sugestije i podršku pružile su mi i mnoge kolege, a posebno dr Mateja Jamnik, dr Renato Busatto, Silvio Ranise, Vlado Kešelj, Jan Wielemaker i dr Alessandro Armando. Zahvalan sam i članovima Grupe za matematičko rezonovanje Univerziteta u Edinburgu na gostoprimstvu

i izuzetnoj saradnji; svim svojim kolegama sa Katedre za računarstvo i informatiku Matematičkog fakulteta u Beogradu na stalnim podsticajima i podršci; fondaciji The British Scholarship Trust na finansijskim sredstvima za moj boravak u Edinburgu i gospođi Celiai Hawksworth iz fondacije na pomoći i velikoj ljubaznosti. Posebno se zahvaljujem svojoj majci Stani, kao i svojoj sestri Snežani i njenoj porodici na beskrajnom razumevanju i podršci bez kojih ne bih uspeo u pisanju ove disertacije.

Beograd, septembar 2000. Predrag Janičić

Preface

I started my research in the field of use of decision procedures in theorem proving while I was working as a visiting member of The Mathematical Reasoning Group (Division of Informatics, University of Edinburgh) during academic year 1996/97, when prof. Alan Bundy introduced me to that problem.

The dissertation consists of: the overview and analysis of most influential methods for combining and integrating decision procedures into theorem provers; the description of one new scheme for incorporating decision procedures; the description of a new general setting for combining and incorporating decision procedures; the description of the implementation of the proposed scheme and general setting and the overview of experimental results; the overview of theory of decidability, rewriting systems, equality systems (which makes the dissertation self-contained); the analysis of Presburger arithmetic and several decision procedures for that theory. By "building-in decision procedures" we mean both the strategies for combining decision procedures and the strategies for incorporation of decision procedures into heuristic theorem provers. By "systems for automated reasoning" we mean theorem provers and familiar systems.

Some parts from this dissertation are included in the papers [60, 59, 58] (which are published or submitted for publication) and some parts are included in internal reports of The Mathematical Reasoning Group (in The Blue Book Notes series).

During my work on this dissertation, I was given invaluable help, guidance, advices and suggestions by my advisors — prof. Žarko Mijajlović and prof. Alan Bundy and I am grateful for all that. Prof. Bundy introduced me to the problem of integration of decision procedures into theorem provers and over the past years he has been leading me through the world of ideas and methodology of automated reasoning; I am very grateful for his invaluable help and everything I learnt from him. I wish to thank dr Ian Green, my second supervisor in Edinburgh; he gave me a lot of help and it was always a real pleasure to collaborate with him. The members of the committee prof. Gordana Pavlović–Lažetić and prof. Duško Vitas also gave me great help in preparing the dissertation. Many of my colleagues gave me help, suggestions and support; I am especially grateful to dr Mateja Jamnik, dr Renato Busatto, Vlado Kešelj, Silvio Ranise, Jan Wielemaker and dr Alessandro Armando. I wish to thank the members of the Mathematical Reasoning Group from the University of Edinburgh for

their hospitality and a great collaboration; all my colleagues from the Chair for Computer Science (Faculty of Mathematics, University of Belgrade) for their permanent support; The British Scholarship Trust for the fundings for my stay in Edinburgh and to Mrs Celia Hawksworth from the Trust for her help and her great kindness. Especially, I am grateful to my mother Stana and to my sister Snežana and her family for their unlimited understanding and support without which I couldn't have made this dissertation.

Belgrade, september 2000. Predrag Janičić

Sadržaj

.

Contents

.

Glava 1

Uvod

Savremeni dokazivači teorema i srodni sistemi za automatsko rezonovanje (sistemi za proveravanje modela, simbolički simulatori, verifikatori softvera, statički analizatori programa i sl) nalaze sve brojnije i sve uspešnije primene. Savremeni dokazivači teorema već mogu da dokažu značajne ili do nedavno otvorene matematičke teoreme (ili suštinski pomognu u njihovom dokazivanju). Dokazivači teorema nalaze primene od ekspernih sistema do programa za šah i bridž. U domenu automatske sinteze softvera, dokazivači teorema koriste se u dokazivanju da postoji program koji zadovoljava određenu specifikaciju; ukoliko je taj dokaz konstruktivan, iz njega se može izvesti traženi program. Ovaj pristup daje veoma uspešne rezultate u sintezi nekih klasa specijalizovanih programa. Primena dokazivača rasprostranjena je i uspešna u sintezi i optimizaciji mikroprocesora (domen tih dokazivača je obično iskazna logika).

Jedna od najznačajnijih primena dokazivača teorema je u verifikaciji softvera i hardvera. Već postoje mikroprocesori koji su u potpunosti formalno verifikovani, a neki od primera automatski verifikovanih algoritama su RSA algoritam za kodiranje sa javnim ključem i Boyer/Mooreov algoritam za traženje stringova. Strogi dokaz ispravnosti programa podrazumeva njegov formalni opis (izražen na formalnom jeziku) i par $\langle \phi, \phi' \rangle$ gde ϕ označava svojstva domena (preduslove), a ϕ' označava izlaz programa (postuslove). Da bi se dokazala ispravnost programa P, potrebno je dokazati sledeće: ako objekat x zadovoljava uslove ϕ i ako program P vraća vrednost y za objekat x, onda x i y zadovoljavaju svojstvo ϕ', tj. važi $(\forall x)((\phi[x] \wedge P(x) = y) \rightarrow \phi'[x, y])$. Dodatno, potrebno je dokazati da se za svaki objekat x koji zadovoljava svojstva ϕ, program P zaustavlja: $(\forall x)((\phi[x] \rightarrow (\exists y)(P(x) = y))$. U dokazivanju svojstava navedenog tipa obično je potrebno dokazati da postoje određene invarijante (tj. relacije između promenljivih koje su zadovoljene pre ulaska i nakon svakog izvršenja bloka naredbi unutar neke petlje). Na primer, u Euklidovom algoritmu za određivanje najvećeg zajedničkog delioca dva broja par brojeva (a, b) $(a \geq b > 0)$ čiji se nzd određuje, u svakoj iteraciji se zamenjuje parom $(b, a \ mod \ b)$ sve dok manji od dva broja u paru nije jednak 0; kada je $b = 0$, vrednost a se vraća kao rezultat; da bi bila dokazana korektnost Euklidovog algoritma potrebno je dokazati da važi in-

varijanta $nzd(a, b) = nzd(b, a \ mod \ b)$ (gde je $a \geq b > 0$). U verifikacijskim problemima veoma često figurišu funkcije koje oslikavaju linearne strukture podataka i sâme iteracije petlje, što prirodno iziskuje korišćenje aritmetike ili njenih segmenata u dokazivanju postojanja invarijanti. Razmotrimo primer programa koji realizuje jednostavnu look-up tabelu. Implementacija tabele je niz pozitivne parne dužine D sa ključevima i njihovim pridruženim vrednostima smeštenim u naizmeničnim lokacijama niza. Niz se pretražuje linearno, pomerajući vrednost tekućeg indeksa I od vrednosti 1 sa korakom 2, zaustavljajući se kada je $I > D$. Jedan od verifikacijskih uslova treba da bude sledeći: ako je traženi ključ pronađen i ako se pridruženoj vrednosti pristupa u $I + 1$ lokaciji niza, onda nije prekoračen opseg niza. Dakle, ako je vrednost indeksa I nedaran broj takav da je $I \leq D$, onda je $I + 1 \leq D$. Uzimajući da je $D = 2 \cdot L$ i $I = 2 \cdot K + 1$ gde su L i K proizvoljni pozitivni celi brojevi, navedeno tvrđenje može biti formulisano na sledeći način: $0 < K \ \wedge \ 0 < L \ \wedge \ 2 \cdot K + 1 \leq 2 \cdot L \ \rightarrow \ 2 \cdot K + 2 \leq 2 \cdot L$. Navedeno tvrđenje pripada Prezburgerovoj aritmetici i može biti dokazano nekom procedurom odlučivanja za nju.

U daljem tekstu neće biti reči o konkretnim primenama dokazivača teorema i drugih srodnih sistema za automatsko rezonovanje, već generalno o unapređenju njihovih domena i efikasnosti, pri čemu će motivacija za neka rešenja najčešće da bude upravo u domenu automatske verifikacije programa.

1.1 Uloga procedura odlučivanja u dokazivačima teorema

U mnogim primenama dokazivača teorema i drugih srodnih sistema za automatsko rezonovanje pojedine teorije imaju poseban značaj (npr. aritmetika u verifikaciji programa), pa je, stoga, opšte prihvaćeno uverenje da će široko i praktično upotrebljivi dokazivači teorema morati da sadrže i heurističke komponente i brze, specijalizovane procedure odlučivanja za manje, relevantne teorije. Procedure odlučivanja sa uspehom se koriste u različitim sistemima za automatsko rezonovanje [25, 33, 101, 114, 93, 42].

Iako su procedure odlučivanja najčešće neefikasne (eksponencijalne ili superekseponencijalne složenosti), one mogu da budu efikasnije od drugih dokazivačkih strategija (npr. indukcije). Uloga procedura odlučivanja je često veoma značjna u dokazivanju teorema: one mogu da suze prostor pretrage heurističke komponente dokazivača i time povećaju njegove mogućnosti. Pored koristi od dokazanih tvrđenja, i negativni rezultati vraćeni od strane procedure odlučivanja mogu biti korisni (na primer, u kontroli generalizacije i drugih heuristika koje ne čuvaju zadovoljivost). Osnovni problem u primeni procedura odlučivanja je to što u većini primena samo mali broj tvrđenja pripada domenu jedne procedure odlučivanja. Neka od tih tvrđenja pripadaju uniji više teorija za koje su raspoložive procedure odlučivanja, dok neka ne pripadaju domenu raspoložive procedure odlučivanja zbog neki dodatnih funkcija ili predikata. U tim situacijama potrebno je kombinovati procedure odlučivanja, uključiti leme ili neke

druge pomoćne dokazivačke strategije (kako bi bio proširen domen procedura odlučivanja i kako bi mu zadato tvrđenje pripalo).

1.2 Kombinovanje procedura odlučivanja

U dokazivanju teorema i, posebno, u verifikaciji softvera i hardvera, uloga aritmetike je često suštinska. Aritmetika je neodlučiva, ali ima odlučive fragmente (i odgovarajuće procedure odlučivanja), kao što su Prezburgerova aritmetika (PNA) i strogo multiplicativna aritmetika (SMA) [97, 107, 84]. Međutim, jedna procedura odlučivanja sâma ima veoma ograničenu primenu. Na primer, (implicitno univerzalno kvantifikovana) formula

$$x \leq y \wedge y \leq x + car(cons(0, x)) \wedge p(h(x) - h(y)) \rightarrow p(0)$$

ne pripada Prezburgerovoj aritmetici. Dodatno, formula

$$x \leq y \wedge y \leq x + c \wedge p(h(x) - h(y)) \rightarrow p(0)$$

koja je dobijena apstrahovanjem (ili generalizacijom) $car(cons(0, x))$ promenljivom c pripada Prezburgerovoj aritmetici (proširenoj čistom teorijom jednakosti), ali nije njena teorema. Ipak, ukoliko su raspoložive i procedure odlučivanja za čistu teoriju jednakosti ($\mathcal{E}$) i za teoriju lista sa funkcijama *car*, *cdr* i *cons* ($\mathcal{L}$), moguće je kombinovanom upotrebom procedura odlučivanja za te tri teorije dokazati navedeno tvrđenje. Da bismo dokazali navedeno tvrđenje, dovoljno je dokazati da njegova negacija

$$x \leq y \ \wedge \ y \leq x + car(cons(0, x)) \ \wedge \ p(h(x) + (-1) \cdot h(y)) \ \wedge \ \neg p(0)$$

nije zadovoljiva. Apstrahovanjem stranih termova dobija se formula koju čine samo literali koji pripadaju jednoj od tri teorije koje koristimo:

$$x \leq y \ \wedge \ y \leq x + v_0 \ \wedge \ car(cons(v_1, x)) = v_0 \ \wedge \ 0 = v_1 \ \wedge \ p(v_2) \ \wedge$$

$$v_4 + (-1) \cdot v_3 = v_2 \ \wedge \ h(y) = v_3 \ \wedge \ h(x) = v_4 \ \wedge \ \neg p(v_1)$$

Svi literali mogu biti podeljeni u tri konjunkcije u zavisnosti od toga kojoj teoriji pripadaju: $x \leq y \ \wedge \ y \leq x + v_0 \ \wedge \ 0 = v_1 \ \wedge \ v_4 + (-1) \cdot v_3 = v_2$ (PNA), $car(cons(v_1, x)) = v_0$ ($\mathcal{L}$) i $p(v_2) \ \wedge \ h(y) = v_3 \ \wedge \ h(x) = v_4 \ \wedge \ \neg p(v_1)$ ($\mathcal{E}$). Svaka od tri konjunkcije je pojedinačno zadovoljiva, pa mora da postoji interakcija između odgovarajućih procedura odlučivanja da bi bila dokazana nezadovoljivost. Procedura odlučivanja za $\mathcal{L}$ detektuje jednakost $v_1 = v_5$ (jer je $car(cons(v_1, x)) = v_0 \ \wedge \ v_1 \neq v_5$ nezadovoljivo) i ona se prosleđuje ostalim konjunkcijama tj. teorijama. Procedura za PNA koristi ovu jednakost i, slično, detektuje implicitnu jednakost $x = y$ i prosleđuje je. Nakon toga, procedura odlučivanja za $\mathcal{E}$ povlači $v_3 = v_4$, a procedura za PNA detektuje jednakost $v_2 = v_5$. Konačno, procedura odlučivanja za $\mathcal{E}$ onda detektuje protivrečnost, čime je dokazana polazna formula. Ovaj primer ilustruje situacije u kojima nekoliko procedura odlučivanja treba da *kooperira* i da bude *kombinovano* u jednu proceduru odlučivanja.

1.3 Proširivanje domena procedura odlučivanja i integrisanje procedure odlučivanja u dokazivače teorema

Domen jedne procedure odlučivanja moguće je proširiti dodatnim pravilima prezapisivanja ili dodatnim lemama kako bi bio pokriven deo neke (često neodlučive) nadteorije. Razmotrimo (implicitno univerzalno kvantifikovanu) formulu

$$l \leq minl(\alpha) \ \wedge \ 0 < k \ \ \rightarrow \ \ l < maxl(\alpha) + k$$

Ova formula ne pripada Prezburgerovoj aritmetici i, dodatno, formula

$$l \leq minl \wedge \ 0 < k \ \rightarrow \ l < maxl + k$$

dobijena apstrahovanjem (generalizacijom) $minl(\alpha)$ novom promenljivom $minl$ i $maxl(\alpha)$ novom promenljivom $maxl$ jeste formula Prezburgerove aritmetike, ali nije njena teorema. Međutim, ako je raspoloživa lema

$$\forall \xi \ (minl(\xi) \leq maxl(\xi)) \ ,$$

onda ona, korišćenjem pogodno odabrane instancijacije, vodi do formule:

$$minl(\alpha) \leq maxl(\alpha) \rightarrow (l \leq minl(\alpha) \wedge \ 0 < k \ \rightarrow \ l < maxl(\alpha) + k) \ .$$

Nakon generalizacije, dobija se formula

$$minl \leq maxl \rightarrow (l \leq minl \wedge \ 0 < k \ \rightarrow \ l < maxl + k)$$

koja može biti dokazana procedurom odlučivanja za Prezburgerovu aritmetiku. Ovaj primer ilustruje kako domen procedure odlučivanja za Prezburgerovu aritmetiku može biti proširen raspoloživim lemama. U situacijama kao što je ova, procedura odlučivanja treba da komunicira sa ostalim komponentama dokazivača, kao što su mehanizam za korišćenje lema, mehanizam za prezapisivanje, modul za generalizaciju itd. i onda govorimo o *integrisanju* ili *inkorporiranju* procedure odlučivanja u dokazivač teorema.

1.4 Pristupi kombinovanju i integrisanju procedura odlučivanja

Postoji nekoliko značajnih pristupa problemu efikasnog kombinovanja i integrisanja procedura odlučivanja u dokazivače teorema. Istraživanja koja se bave ovom problematikom su uglavnom išla u dva pravca: jedan koji se odnosi na kombinovanje procedura odlučivanja i jedan koji se odnosi na integrisanje procedura odlučivanja u heurističke dokazivače teorema. Sheme za kombinovanje procedura odlučivanja najčešće se zasnivaju na nekim lokalnim, specifičnim strukturama podataka [87, 104, 32, 9, 11, 100, 10] i usredsređene su na kombinovanje

procedura odlučivanja, ali neke od njih ostavljaju prostor i za korišćenje dodatnih tehnika (kao što je upotreba lema). Integracione sheme koriste različite funkcionalnosti heurističkog dokazivača kao što su tehnike prezapisivanja, mehanizam za leme, različite forme pojednostavljivanja itd. [17, 68, 69, 2, 59, 4]. Sheme za kombinovanje procedura odlučivanja su obično usmerene na primenu u odlučivim kombinacijama više teorija, dok su integracione sheme usmerene na primene u (najčećenje) neodlučivim proširenjima odlučivih teorija. Istraživanja u sferi kombinovanja procedura odlučivanja su bogata i teorijskim i praktičnim rezultatima, dok su istraživanja u sferi integracionih shema još uvek na nešto nižem nivou.

Nelson/Oppenova shema za kombinovanje procedura odlučivanja [86] se koristi u nekoliko dokazivačkih sistema, uključujući Stanford Pascal Verifier [75], ESC [41] i EVES [30]. U ovom pristupu, procedure odlučivanja za disjunktne teorije se kombinuju apstrahovanjem termova koji ne pripadaju pojedinačnim teorijama i prosleđivanjem izvedenih jednakosti iz jedne teorije u drugu. Shostakova shema za kombinovanje procedura odlučivanja [104] koristi se u nekoliko drugih sistema, uključujući EHDM [43], PVS [92], STeP [76, 11] i SVC [9]. U ovom pristupu, funkcije *solve* za specifične teorije (na primer, funkcije *solve* za jednakosnu realnu aritmetiku, teoriju lista itd) su čvrsto kombinovane efikasnim algoritmom za izračunavanje kongruentnog zatvorenja za bazne jednakosti.

Jedan od najznačajnijih metoda u domenu integrisanja procedura odlučivanja u dokazivače teorema je procedura za linearnu aritmetiku integrisana u Boyer-/Mooreov NQTHM [17]. Ovaj sistem uključuje mnoštvo specijalnih struktura podataka i opis procedure je često dat u terminima tih struktura podataka, a ne u terminima njihovog logičkog značenja. Značajne su i sheme za ugradnju procedura odlučivanja za aritmetiku koje se koriste u prezapisivački zasnovanom dokazivaču teorema TECTON [68].

1.5 Fleksibilna ugradnja procedura odlučivanja i kako je dostići

Da bi bila omogućena šira primenljivost neke sheme za kombinovanje ili integrisanje procedura odlučivanja u dokazivače teorema potrebno je:

- da njen opis bude formalan (a ne dât u terminima implementacije); to, s jedne strane, treba da omogući jednostavnu implementaciju sheme u različitim dokazivačima, a, s druge, da omogući formalno rezonovanje o svojstvima sheme;
- da je arhitektura sheme takva da dopušta jednostavnu zamenu teorije (odnosno teorija) i procedura odlučivanja na koje se odnosi.

Krajem devedestih godina značajni napori usmereni su upravo ka fleksibilnoj ugradnji procedura odlučivanja, a za poznate sheme ka formulisanju opisa pogodnih za strogo rezonovanje o njihovim svojstima (kao što su zaustavljanje, saglasnost i potpunost).

Ograničeno kontekstualno prezapisivanje Armanda i Ranisea [2] je zamišljeno (i implementirano) kao shema za fleksibilno integrisanje procedura odlučivanja. Ta shema može biti instancirana specifičnom teorijom i specifičnom procedurom odlučivanja za tu teoriju. Na primer, ograničeno kontekstualno prezapisivanje može biti instancirano tako da je ekvivalentno Boyer/Mooreovoj integracionoj shemi ili tako da je ekvivalentno shemi upotrebljenoj u sistemu TECTON. Zaustavljanje i saglasnost za ograničeno kontekstualno prezapisivanje su dokazani formalno za apstraknu shemu kada su zadovoljeni određeni uslovi koji se odnose na prezapisivački mehanizam i na proceduru odlučivanja [4].

S druge strane, Barrett, Dill i Stump [10] predlažu fleksibilni okvir za kombinovanje procedura odlučivanja. U ovom okviru (izgrađenom nad nekoliko apstraktnih osnovnih metoda) mogu da budu implementirane sheme u stilu Nelson/Oppenove i Shostakove procedure i njihova korektnost (zaustavljanje, saglasnost i kompletnost) može biti formalno dokazana.

U ovoj tezi predlažemo jednu opštu shemu — EPM — za fleksibilno integrisanje procedura odlučivanja[1] [59] i jedan opšti okvir kako za kombinovanje tako i za integrisanje procedura odlučivanja u dokazivače teorema [58]. Shema EPM predstavlja opšti metod koji se instancira konkretnom procedurom odlučivanja A za teoriju $\mathcal{T}$. Ona je zasnovana na idejama iz Boyer/Mooreove integracione sheme, ali predstavlja njenu značajnu formalizaciju i višestruko uopštenje. Ovaj opšti metod može biti upotrebljen za različite teorije i različite procedure odlučivanja. Nova procedura odlučivanja može biti jednostavno uključena u sistem samo ako pruža ograničen skup funkcionalnosti kao što su proveravanje zadovoljivosti i eliminacija kvantifikatora. Opšti okvir za kombinovanje i za integrisanje procedura odlučivanja u dokazivače teorema zasnovan je na makro pravilima izvođenja motivisanim različitim poznatim shemama. Ovaj fleksibilan i modularan okvir omogućava jednostavno implementiranje najznažajnijih shema i za kombinovanje i za integrisanje procedura odlučivanja (pri čemu je najveći deo programskog kôda deljen). Dodatno, ovaj opšti okvir omogućava i jednostavno kombinovanje ideja iz različitih shema (npr. proširivanje Nelson/Oppenove sheme mehanizmom za korišćenje lema), kao i jednostavno poređenje implementiranih shema. Korišćenjem ovog okvira, jedan dokazivač teorema može da ima na raspolaganju veliki broj algoritama za kombinovanje i ugradnju procedura odlučivanja implementiranih i dostupnih na uniforman način. Ovaj pristup opštiji je od pristupa predloženih u radovima [2, 59, 10], jer se on odnosi i na kombinovanje i na integrisanje procedura odlučivanja u dokazivače teorema. Nisu nam poznati drugi pristupi koji se istovremeno odnose na oba ova problema.

1.6 Pregled teze

Drugi deo teze uvodi notaciju, osnovne pojmove i osnovna svojstva koji se koriste u tekstu: pojam višesortnog jezika i teorije prvog reda, pojam odlučivosti teorije,

[1] Radovi [59] i [10] sa analognim imenima (prezentovani na dva uzastopna CADE-a) su paralelna u dva istraživačka pravca — u kombinovanju i integrisanju procedura odlučivanja.

svojstva proširenja teorija, unija teorija i preseka teorija, osnovne pojmove u vezi sa prezapisivanjem termova, jednakosne sisteme i, zbog posebnog značaja u verifikaciji programa, Prezburgerovu aritmetiku.

U trećem delu teze dati su pregled i analiza najznačanijih shema za kombinovanje i integrisanje procedura odlučivanja: Nelson/Oppenove, Shostakove, Boyer/Mooreove (iz sistema NQTHM), Kapur/Nieove (iz sistema TECTON), Armando/Raniseove (ograničeno kontekstualno prezapisivanje) i Barrett/Dill/-Stumpove. Ove sheme služe kao motivacija i osnova za shemu EPM i opšti okvir za korišćenje procedura odlučivanja koji su dati u nastavku teze.

Četvrti deo teze sadrži opis opšte sheme EPM za fleksibino integrisanje procedura odlučivanja u dokazivače teorema, kao i dokaze zaustavljanja i saglasnosti.

U petom delu teze dat je opis opšteg okvira za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema. Nakon opisa makro pravila izvođenja, dat je opis šest shema poznatih iz literature iskazan na jeziku tih pravila. Za neke od tih shema dokazani su je potpunost, zaustavljanje i saglasnost.

Šesti deo teze odnosi se na implementaciju i eksperimentalne rezultate apstraktne sheme EPM i njenih instanci za Prezburgerovu aritmetiku, kao i na implementaciju opšteg okvira za korišćenje procedura odlučivanja. U opštem okviru, implementirane su i četiri sheme poznate iz literature i u ovom delu teze priloženi su dobijeni eksperimentalni rezultati.

Sedmi i osmi deo teze odnose se na pravce daljih istraživanja i na zaključke.

U dodatku teze, dat je detaljniji pregled pojmova koji se odnose na odlučivost (dodatak **A**), prezapisivanje termova, uređenja i zaustavljanje (dodatak **B**), jednakosne teorije (dodatak **C**) i Prezburgerovu aritmetiku (dodatak **D**). Dodaci **A**, **B** i **C** sadrže poznate pojmove i rezultate koji su od neposrednog značaja za glavni materijal prezentovan u tezi; time, u velikoj meri, teza sadrži sve teorijske postavke neophodne za njeno razumevanje. Dodatak **D**, pored poznatih pojmova i rezultata, sadrži i originalne, eksperimentalne rezultate uporedne analize nekoliko algoritama za Prezburgerovu aritmetiku, kao i opis fleksibilne implementacije algoritama na bazi normalizovanja prezapisivanjem.

Osnovni doprinos teze, u sferi kombinovanja i integrisanja procedura odlučivanja, dat je u četvrtom i petom delu (sa praktičnim rezultatima datim u šestom delu), dok dodatak **D** sadrži doprinose koji se odnose na pojedinačne procedure odlučivanja za Prezburgerovu aritmetiku.

1.7 Introduction: Summary

The use of decision procedures is very important in theorem proving. Decision procedures can increase abilities of a prover by reducing the search space of its heuristic components. However, in some applications only a small number of conjectures fall within the scope of a decision procedure. Some of these conjectures could, in an informal sense, fall "just outside" that scope. In these situations some additional knowledge is needed and lemmas have to be invoked, decision procedures have to communicate with each other or with the heuristic component of a theorem prover.

For instance, the (implicitly universally closed) formula

$$x \leq y \wedge y \leq x + car(cons(0, x)) \wedge p(h(x) - h(y)) \rightarrow p(0)$$

is not a Presburger arithmetic formula and cannot be proved by a decision procedure for this theory. However, if there are available decision procedures for the theory of lists with *car*, *cdr* and *cons* and for the pure theory of equality, it is possible to combine them with a decision procedure for Presburger arithmetic and to prove the above conjecture. This example illustrates situations when several decision procedures have to *cooperate* and to be *combined* into one decision procedure. As another example, consider the (implicitly universally closed) formula

$$l \leq minl(\alpha) \ \wedge \ 0 < k \ \rightarrow \ l < maxl(\alpha) + k \ .$$

It is not a Presburger arithmetic formula and can not be proved by a decision procedure for it. However, if the lemma $\forall \xi \ (minl(\xi) \leq maxl(\xi))$ is available, it can be used (with the right instantiation) and the above conjecture can be proved by a decision procedure for Presburger arithmetic (aided by generalisation). This example illustrates how the realm of the decision procedure for Presburger arithmetic can be extended by the use of available lemmas. In situations like this one, a decision procedure has to communicate with other components of a prover, such as a lemma invoking mechanism, a rewriting mechanism, simplification techniques etc. and then we are talking about *integrating* or *incorporating* a decision procedure into a theorem prover.

There are several dominating approaches used in handling the problem of efficient combining and integrating decision procedures into theorem provers. The research concerning these issues have gone mostly along different two lines: one concerning combining decision procedures (e.g. Nelson/Oppen's procedure [86], Shostak's procedure [104]) and one concerning integrating decision procedures into heuristic theorem provers (e.g. Boyer/Moore's procedure [17], Kapur/Nie's procedure [68]). Combination schemes typically rely on some local, specific data structures and are focused on combining decision procedures, but some of them also use additional techniques (such as the use of additional rules and lemmas etc). Integration schemes use different functionalities of heuristic theorem provers such as rewriting techniques, lemma invoking mechanisms, variety of simplifications etc. Combination schemes are typically aimed at decidable combinations of theories, while integration schemes are primarily intended for use in undecidable extensions of some decidable theories. The research on combination of decision procedures is rich in theoretical and practical results, while the research on the integration of decision procedures is at an earlier stage.

In this thesis we propose a flexible combination and integration schemes. Such schemes should be usable with different underlying theories and corresponding decision procedures. In addition, we are interested in making a formal framework capable of giving formal descriptions of different combination/integration schemes (instead of giving their descriptions in an informal way or by the description of an implementation). Such a framework should enable formal reasoning about properties of different combination/integration schemes

(e.g. termination, soundness, completeness). In this thesis we propose one general framework (EPM) for the integrating of a decision procedure into a theorem prover (which is instantiated by a specific underlying theory and specific decision procedure based on quantifier elimination) and one general setting for both combining and integrating decision procedures into theorem provers (which is based on a set of macro inference rules and within which a number of combination/integration schemes can be implemented).

The rest of the thesis is structured as follows: in chapter 2 we give some basic background and notation information, in chapter 3 we give an overview of some of the most important combination/integration schemes from the literature. In chapter 4 we propose the EPM scheme and in chapter 5 we describe a general setting for the combining and integrating decision procedures into theorem provers. In chapter 6 we describe an implementation of the EPM scheme and of the proposed general setting (along with an implementation of four combination and integration schemes) and we give some experimental results. In chapters 7 and 8 we discuss future work and draw some final conclusions. In the appendices, we give a more detailed account of decidable theories (appendix **A**), term rewriting, ordering and termination (appendix **B**), equational theories (appendix **C**) and Presburger arithmetic (appendix **D**). These appendices make the thesis self-contained. In addition, appendix **D** presents some results concerning an experimental comparison between several decision procedures for Presburger arithmetic and concerning flexible implementation of decision procedures for Presburger arithmetic based on specific normalisation rules. The main contribution of the thesis, concerning combining and integrating decision procedures into theorem provers is given in chapters 4 and 5 (with some practical results given in chapter 6), while some contribution concerning specific decision procedures for Presburger arithmetic is given in appendix **D**.

Glava 2

Notacija i osnovni pojmovi

Rezultati izloženi u daljem radu odnose se na teorije prvog reda sa jednakošću. Pogodnosti radi, uglavnom će biti korišćena višesortna logika (iako je sasvim ekvivalentno sorte moguće opisati posebnim unarnim predikatima). U ovom delu dajemo pregled notacije i standardnih pojmova koji se koriste u daljem tekstu [47].

2.1 Višesortni jezik prvog reda

Višesortnim jezikom (rečnikom ili signaturom) prvog reda nazivamo petorku $\mathcal{L} = \langle T, V, \Sigma, \Pi, d\rangle$, gde su T, V, Σ i Π međusobno disjunktni skupovi, a d funkcija koja elementima skupa $\Sigma \cup \Pi$ pridružuje torke skupa T. T je konačan skup sorti (vrsta) $\{\tau_i | i \in I\}$. Ukoliko skup T ima n elemenata, onda za jezik $\mathcal{L}$ kažemo da je *n-sortni*. Za svaku sortu τ_i, neka je V_i prebrojiv skup promenljivih te sorte (x_1^i, x_2^i, x_3^i, ...) i neka je $V_i \cap V_j = \emptyset$ for $i \neq j$. Neka je $V = \cup_{i\in I} V_i$. Σ je konačan skup funkcijskih simbola; funkcijskom simbolu f funkcija d pridružuje torku $\langle \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}, \tau_j\rangle$ $(\tau_{i_1}, \ldots, \tau_{i_k}, \tau_j \in T)$. Vrednost k zovemo *arnošću* funkcijskog simbola. Ako je $k = 0$ za neki funkcijski simbol, tj. ako simbolu f funkcija d pridružuje torku $\langle\tau\rangle$, onda taj funkcijski simbol zovemo *konstantom* sorte τ. Π je konačan skup predikatskih (relacijskih) simbola; predikatskom simbolu p funkcija d pridružuje torku $\langle \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}\rangle$ $(\tau_{i_1}, \ldots, \tau_{i_k} \in T)$. Vrednost k zovemo *arnošću* predikatskog simbola. Vrednost k jednaka je 0 tačno za dva predikatska simbola ($\top$ i $\bot$) i njih zovemo zovemo *logičkim konstantama* ili *istinitosnim konstantama*.

Za dve signature $\mathcal{L}_1 = \langle T_1, V_1, \Sigma_1, \Pi_1, d_1\rangle$, $\mathcal{L}_2 = \langle T_2, V_2, \Sigma_2, \Pi_2, d_2\rangle$, kažemo da su disjunktne ako ne postoji funkcijski simbol f takav da je $f \in \Sigma_1$ i $f \in \Sigma_2$ i ne postoji predikatski simbol p takav da je $p \in \Pi_1$ i $p \in \Pi_2$.

$\mathcal{L}$-term definišemo na sledeći način:

- funkcijski simboli (konstante) i promenljive sorte τ su $\mathcal{L}$-termovi sorte τ;
- ako su $t_1, t_2, \ldots, t_n$ $\mathcal{L}$-termovi sorti $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}$ redom i ako je funkci-

jskom simbolu $f \in \Sigma$ pridružena struktura $\langle \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}, \tau \rangle$, onda je $f(t_1, t_2, \ldots, t_n)$ $\mathcal{L}$-term sorte τ;

- svi termovi mogu biti dobijeni konačnom primenom prethodna dva pravila.

Za term koji je promenljiva x, skup slobodnih promenljivih je $\{x\}$; za term f, $f \in \Sigma$ skup slobodnih promenljivih je prazan; za term $f(t_1, t_2, \ldots, t_n)$ skup slobodnih promenljivih je unija skupova slobodnih promenljvih u termovima $t_1, t_2, \ldots, t_n$. *$\mathcal{L}$-atomička formula* je

- izraz $p(t_1, t_2, \ldots, t_n)$ (gde su $t_1, t_2, \ldots, t_n$ $\mathcal{L}$-termovi sorti $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}$ redom i $p \in \Pi$ je predikatski simbol kojem je pridružena struktura $\langle \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k} \rangle$);
- izraz $t_1 =_\tau t_2$ (gde su t_1 i t_2 $\mathcal{L}$-termovi sorte τ);
- istinitosna konstanta (element skupa $\{\top, \bot\}$).

Za izraz $p(t_1, t_2, \ldots, t_n)$ skup slobodnih promenljivih je unija skupova slobodnih promenljivih u termovima $t_1, t_2, \ldots, t_n$; za $t_1 =_\tau t_2$ skup slobodnih promenljivih je unija skupova slobodnih promenljivih u termovima t_1 i t_2. $\mathcal{L}$-literal je oblika α ili $\neg\alpha$ gde je α $\mathcal{L}$-atomička formula. Ako je F_1 $\mathcal{L}$-atomička formula, onda je ona i *$\mathcal{L}$-formula*. Ako su F_1 i F_2 $\mathcal{L}$-formule i $x \in V$, onda su i $(\forall x : \tau)F_1$, $(\exists x : \tau)F_1$, $\neg F_1$, $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \to F_2)$ $\mathcal{L}$-formule. Skup slobodnih promenljvih u $(\forall x : \tau)F_1$ i $(\exists x : \tau)F_1$ je razlika skupa slobodnih promenljivih u F_1 i skupa $\{x\}$. Skup slobodnih promenljivih u $\neg F_1$ je skup slobodnih promenljivih u F_1. Skup slobodnih promenljivih u $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \to F_2)$ je unija skupova slobodnih promenljivih u F_1 i F_2. Promenljivu x koju sadrži $\mathcal{L}$-formula f i koja nije slobodna zovemo *vezana promenljiva*. $\mathcal{L}$-formulu bez slobodnih promenljivih zovemo *zatvorena $\mathcal{L}$-formula* ili *$\mathcal{L}$-rečenica*. Formula F je *bazna* ako nema promenljivih. Formula F je u *preneks normalnoj formi* ako je oblika $(Q_1 x_1 : \tau_{i_1})(Q_2 x_2 : \tau_{i_2}) \ldots (Q_k x_k : \tau_{i_k})F'$ gde je $Q_i \in \{\forall, \exists\}$, $x_j \in V_{i_j}$ i F' ne sadrži kvantifikatore. Formula F je *univerzalno zatvorena* ako je oblika $(\forall x_1 : \tau_{i_1})(\forall x_2 : \tau_{i_2}) \ldots (\forall x_k : \tau_{i_k})F'$ gde je $x_j \in V_{i_j}$ i F' ne sadrži kvantifikatore. Formula F je *egzistencijalno zatvorena* ako je oblika $(\exists x_1 : \tau_{i_1})(\exists x_2 : \tau_{i_2}) \ldots (\exists x_k : \tau_{i_k})F'$ gde je $x_j \in V_{i_j}$ i F' ne sadrži kvantifikatore. Ako formula F ima slobodne promenljive $x_1, x_2, \ldots, x_k$ koje su redom sorti $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}$, onda formulu $(\forall x_1 : \tau_{i_1})(\forall x_2 : \tau_{i_2}) \ldots (\forall x_k : \tau_{i_k})F$ nazivamo *univerzalnim zatvorenjem* formule F i kraće je označavamo sa $\forall * F$. Ako formula F ima slobodne promenljive $x_1, x_2, \ldots, x_k$ koje su redom sorti $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}$, onda formulu $(\exists x_1 : \tau_{i_1})(\exists x_2 : \tau_{i_2}) \ldots (\exists x_k : \tau_{i_k})F$ nazivamo *egzistencijalnim zatvorenjem* formule F i kraće je označavamo sa $\exists * F$.

U narednim delovima teze, zbog jednostavnosti, najčešće ćemo pisati samo simbol jednakosti ne označavajući sortu na koju se odnosi (osim u slučajevima kada to nije jasno iz konteksta).

Formula koju sadrži formula F ima *polaritet* koji je pozitivan $(+)$ ili negativan $(-)$. Polazna formula (npr. formula koja se dokazuje) ima pozitivan polaritet, što zapisujemo F^+. Komplement polariteta p zapisujemo $\overline{p}$, i definišemo

da važi $\overline{+} = -$, kao i $\overline{-} = +$. Polaritet se definiše rekurzivno nad skupom formula: $(\neg F^{\overline{p}})^p$ (ako je p polaritet formule $\neg F$ onda je $\overline{p}$ polaritet formule F), $((\forall x : \tau)F^p)^p$ (ako je p polaritet formule $(\forall x : \tau)F$, onda je p polaritet i formule F), $((\exists x : \tau)F^p)^p$, (ako je p polaritet formule $(\exists x : \tau)F$, onda je p polaritet i formule F), $(F_1^p \wedge F_2^p)^p$, (ako je p polaritet formule $(F_1 \wedge F_2)$, onda formule F_1 i F_2 imaju polaritet p), $(F_1^p \vee F_2^p)^p$, (ako je p polaritet formule $(F_1 \vee F_2)$, onda formule F_1 i F_2 imaju polaritet p), $(F_1^{\overline{p}} \to F_2^p)^p$ (ako je p polaritet formule $(F_1 \to F_2)$, onda formula F_1 ima polaritet $\overline{p}$, a F_2 ima polaritet p).

2.2 Semantika višesortnog jezika prvog reda

Interpretacija I n-sortnog jezika $\mathcal{L} = \langle T, V, \Sigma, \Pi, d\rangle$ je struktura koju čine:

- *Domeni* $D_1, D_2, \ldots, D_n$ od kojih je svaki neprazan skup;
- Za svaki simbol konstante (tj. za svaki funkcijski simbol f za koji je $d(f) = \langle\tau_i\rangle$), element c_I iz skupa D_i;
- Za svaki funkcijski simbol f za koji je $d(f) = \langle\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}, \tau_j\rangle$ preslikavanje f_I iz $D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k}$ u D_j;
- Za svaki predikatski simbol p za koji je $d(p) = \langle\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}\rangle$ preslikavanje p_I iz $D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k}$ u $\{0, 1\}$.

Valuacija (ili niz dodele) v interpretacije I je niz

$$v_j^l,\ l = 1, 2, \ldots, n;\ j = 1, 2, 3, \ldots\ ,$$

gde vrednost v_j^l pripada skupu D_l. Kažemo da je v_j^l *vrednost* promenljive x_j^l u valuaciji v.

Za interpretaciju I, valuaciju v i term t jezika $\mathcal{L}$ definišemo $V(I, v, t)$ na sledeći način:

- Ako je t simbol konstante, tj. funkcijski simbol f arnosti 0, onda je $V(I, v, t) = c_I$.
- Ako je t promenljiva x_i^l, onda je $V(I, v, t) = v_i^l$.
- Ako je $t = f(x_1, x_2, \ldots, x_m)$ i ako je $V(I, v, x_i) = d_i,\ d_i \in D_{j_i},\ i = 1, 2, \ldots, m$, onda je $V(I, v, t) = f_I(d_1, d_2, \ldots, d_m)$.

Za interpretaciju I, valuaciju v i formulu α jezika $\mathcal{L}$ definišemo $V(I, v, \alpha)$ na sledeći način:

- $V(I, v, \top) = 1$;
- $V(I, v, \bot) = 0$;
- Ako je α oblika $p(x_1, x_2, \ldots, x_m)$ i ako je $V(I, v, x_i) = d_i,\ d_i \in D_{j_i},\ i = 1, 2, \ldots, m$, onda je $V(I, v, \alpha) = p_I(d_1, d_2, \ldots, d_m)$.

- Ako je α oblika $\neg\beta$, onda je

$$V(I,v,\alpha)=\begin{cases}0, & \text{ako je } V(I,v,\beta)=1\\ 1, & \text{inače}\end{cases}$$

- Ako je α oblika $(\beta\to\gamma)$, onda je

$$V(I,v,\alpha)=\begin{cases}0, & \text{ako je } V(I,v,\beta)=1 \text{ i } V(I,v,\gamma)=0\\ 1, & \text{inače}\end{cases}$$

- Ako je α oblika $(\beta\vee\gamma)$, onda je

$$V(I,v,\alpha)=\begin{cases}0, & \text{ako je } V(I,v,\beta)=0 \text{ i } V(I,v,\gamma)=0\\ 1, & \text{inače}\end{cases}$$

- Ako je α oblika $(\beta\wedge\gamma)$, onda je

$$V(I,v,\alpha)=\begin{cases}1, & \text{ako je } V(I,v,\beta)=1 \text{ i } V(I,v,\gamma)=1\\ 0, & \text{inače}\end{cases}$$

- Ako je α oblika $(\beta\leftrightarrow\gamma)$, onda je

$$V(I,v,\alpha)=\begin{cases}1, & \text{ako je } V(I,v,\beta)=V(I,v,\gamma)=1\\ 0, & \text{inače}\end{cases}$$

- Ako je α oblika $(\exists x_i^l:\tau_l)\beta$, onda je $V(I,v,\alpha)=1$ ako postoji valuacija w interpretacije I takva da $v_j^k=w_j^k$ važi za sve vrednosti j i k osim eventualno za $j=i$ i $k=l$ i da je $V(I,w,\alpha)=1$; inače je $V(I,v,\alpha)=0$;

- Ako je α oblika $(\forall x_i^l:\tau_l)\beta$, onda je $V(I,v,\alpha)=0$ ako postoji valuacija w interpretacije I takva da $v_j^k=w_j^k$ važi za sve vrednosti j i k osim eventualno za $j=i$ i $k=l$ i da je $V(I,w,\alpha)=0$; inače je $V(I,v,\alpha)=1$;

Ako je α formula, onda vrednost $V(I,v,\alpha)$ zavisi od vrednosti v_i^l samo ako se promenljiva x_i^l pojavljuje slobodna u α. Dakle, ako je α rečenica, onda vrednost $V(I,v,\alpha)$ ne zavisi od v, pa možemo da je pišemo kraće $V(I,\alpha)$. Za rečenicu α kažemo da je I *zadovoljava*, da je I njen *model* ili da je *tačna u interpretaciji* I ako je $V(I,\alpha)=1$. Za formulu α kažemo da je *zadovoljiva* u I ako je I model rečenice $\exists*\,\alpha$.[1]. Za formulu α kažemo da je I njen model ako je I model rečenice $\forall*\,\alpha$. Za rečenicu α kažemo da je *netačna u interpretaciji* I (ili da je I ne zadovoljava) ako je $V(I,\alpha)=0$. Za formulu α kažemo da je nezadovoljiva u I ako rečenica $\exists*\,\alpha$ nije tačna u I. Ako je $\mathcal{A}$ skup formula jezika $\mathcal{L}$ onda kažemo da je I *model* za $\mathcal{A}$ ako je I model za svako α iz $\mathcal{A}$.

[1]Navedena definicija je dualna standardnoj definiciji valjanje formule, u kojoj se slobodne promenljive smatraju implicitno univerzalno kvantifikovanim.

Neki autori zadovoljivost formule zasnivaju na zadovoljivosti njenog univerzalnog zatvorenja. Pojam zadovoljivosti formule zasnivan na zadovoljivosti egzistencijalnog zatvorenja pogodan je (zbog dualnosti) za sisteme u kojima se dokazi izvode pobijanjem.

Za skup formula $\mathcal{A}$ kažemo da je *konzistentan* (ili neprotivrečan) ako ima bar jedan model, a inače kažemo da je *protivrečan* ili *nekonzistentan*. Za rečenicu α kažemo da je *konzistentna* ili *protivrečna* u zavisnosti od toga da li je skup $\{\alpha\}$ konzistentan ili protivrečan.

Kažemo da je formula α *logička posledica* skupa rečenica $\mathcal{A}$ i pišemo $\mathcal{A} \models \alpha$ ako je svaki model za $\mathcal{A}$ model i za α. U tom slučaju, kažemo i da $\mathcal{A}$ povlači α ili da α sledi iz $\mathcal{A}$. Formule koje pripadaju skupu $\mathcal{A}$ zovemo *premise* ili *hipoteze*, a formulu α zovemo *zaključkom*. Ako je skup $\mathcal{A}$ konačan ($\mathcal{A} = \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$), onda pišemo (izostavljajući zagrade): $\gamma_1, \gamma_2, \ldots, \gamma_m \models \alpha$. Ako je $\mathcal{A}$ prazan skup, tj. ako je formula α tačna u svakoj interpretaciji, onda kažemo da je ona *valjana* i pišemo $\models \alpha$. Sa $\mathcal{A} \not\models \alpha$ označavamo da ne važi $\mathcal{A} \models \alpha$.

Kažemo da je formula α zadovoljiva u $\mathcal{A}$, ako postoji model za $\mathcal{A}$ u kojem je rečenica $\exists * \alpha$ tačna. Važi $\mathcal{A} \models \alpha$ ako i samo ako je rečenica $\forall * \alpha$ tačna u svakom modelu za $\mathcal{A}$, tj. ako i samo ako rečenica $\exists * \neg\alpha$ nije zadovoljiva u $\mathcal{A}$, tj. ako i samo ako formula $\neg\alpha$ nije zadovoljiva u $\mathcal{A}$.

2.3 Izvođenje dokaza u logici prvog reda

Prilikom uvođenja pojma dokaza u logici prvog reda, jednostavnosti radi, ograničićemo se na jezik koji ne sadrži simbole $\vee$, $\wedge$, $\leftrightarrow$ i $\forall$. To nije suštinsko ograničenje, jer ove simbole možemo da smatramo skraćenicama: $(\alpha \vee \beta)$ smatramo kraćim zapisom za $((\neg\alpha) \rightarrow \beta)$, $(\alpha \vee \beta)$ za $(\neg(\alpha \rightarrow \neg\beta))$, $(\alpha \leftrightarrow \beta)$ za $(\neg(\alpha \rightarrow (\neg\beta)))$, $(\forall x)\alpha$ za $\neg((\exists x)(\neg\alpha))$.

Logičke aksiome prvog reda su (gde su α, β i γ formule nekog jezika prvog reda):

1. $\alpha \rightarrow (\beta \rightarrow \alpha)$
2. $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
3. $(\neg\beta \rightarrow \neg\alpha) \rightarrow ((\neg\beta \rightarrow \alpha) \rightarrow \beta)$
4. formule oblika: $\alpha(t) \rightarrow (\exists x)\alpha(x)$, gde je t term.
5. formule oblika: $(\exists x)\alpha \rightarrow \alpha$, gde je α formula u kojoj se ne pojavljuje promenljiva x.

Aksiome jednakosti su:

1. $(\forall s : \tau)(s =_\tau s)$
2. $(\forall s : \tau)(\forall t : \tau)(s =_\tau t \rightarrow t =_\tau s)$
3. $(\forall r : \tau)(\forall s : \tau)(\forall t : \tau)(r =_\tau s \wedge s =_\tau t \rightarrow r =_\tau t)$
4. $(\forall s_1 : \tau_{i_1})(\forall t_1 : \tau_{i_1}) \ldots (\forall s_n : \tau_{i_n})(\forall t_n : \tau_{i_n})\ (s_1 =_{\tau_{i_1}} t_1 \wedge s_2 =_{\tau_{i_2}} t_2 \wedge \ldots \wedge s_n =_{\tau_{i_n}} t_n \rightarrow f(s_1, s_2, \ldots, s_n) =_\tau f(t_1, t_2, \ldots, t_n))$
5. $(\forall s_1 : \tau_{i_1})(\forall t_1 : \tau_{i_1}) \ldots (\forall s_n : \tau_{i_n})(\forall t_n : \tau_{i_n})(s_1 =_{\tau_{i_1}} t_1 \wedge s_2 =_{\tau_{i_2}} t_2 \wedge \ldots \wedge s_n =_{\tau_{i_n}} t_n \rightarrow p(s_1, s_2, \ldots, s_n) \leftrightarrow p(t_1, t_2, \ldots, t_n))$

Pravila izvođenja logike prvog reda su:

1. *Modus ponens*: iz α i $\alpha \to \beta$ sledi β
2. *Egzistencijalna generalizacija*: iz $\beta(x) \to \gamma$ sledi $((\exists x : \tau)\beta(x)) \to \gamma$ (pri čemu promenljiva x nema slobodna pojavljivanja u γ.

Neka je $\mathcal{A}$ skup rečenica jezika prvog reda $\mathcal{L}$. Niz α_1, α_2, ..., α_m formula jezika $\mathcal{L}$ zovemo *dokazom* formule α_m iz $\mathcal{A}$ ako za svako α_i $(1 \leq i \leq m)$ važi:

- $\alpha_i \in \mathcal{A}$
- α_i je logička aksioma prvog reda
- postoje indeksi j i k, manji od i takvi da je α_k oblika $\alpha_j \to \alpha_i$ (tj. formula α_i dobijena je primenom pravila modus ponens iz formula α_j i α_k).
- postoji indeks j manji od i takav da je α_j oblika $\beta(x) \to \gamma$ i α_i je oblika $((\exists x : \tau)\beta(x)) \to \gamma$, pri čemu promenljiva x nema slobodna pojavljivanja u γ (tj. formula α_i dobijena je primenom pravila egzistencijelne generalizacije iz formule α_j).

Ako je $\mathcal{A}$ skup rečenica jezika $\mathcal{L}$, ako je α formula jezika $\mathcal{L}$ i ako postoji dokaz formule α iz $\mathcal{A}$ (koji se sastoji samo od formula jezika $\mathcal{L}$) to zapisujemo

$$\mathcal{A} \vdash_{\mathcal{L}} \alpha$$

i kažemo da je formula α *logički izvodiva* iz skupa rečenica $\mathcal{A}$. Kada se jezik $\mathcal{L}$ podrazumeva iz konteksta pišemo samo $\mathcal{A} \vdash \alpha$. Može se dokazati da ako je $\mathcal{A} \cup \alpha \vdash \beta$, onda je $\mathcal{A} \vdash (\alpha \to \beta)$. Između pojmova logičke posledice i logičke izvodivosti postoji neposredna veza:

Ako je $\mathcal{A}$ skup rečenica jezika $\mathcal{L}$ i α rečenica jezika $\mathcal{L}$ onda važi

$$\mathcal{A} \models \alpha \text{ ako i samo ako } \mathcal{A} \vdash \alpha$$

2.4 Teorija

Teorija može biti zadata *aksiomatski* ili *semantički* (i mi ćemo u daljem tekstu koristiti jedan ili drugi vid u zavisnosti od pogodnosti).

Definicija 2.1 *Neka je H skup rečenica (koje nazivamo* aksiomama*) nad jezikom $\mathcal{L}$ koji uključuje logičke aksiome prvog reda (u njima se, u konstrukciji termova i formula, koriste samo simboli jezika $\mathcal{L}$).* Teorija $Th(H)$ aksiomatizovana *skupom H je skup svih rečenica nad jezikom $\mathcal{L}$ logički izvodivih iz H, tj.*

$$Th(H) = \{\alpha \mid H \vdash \alpha\} .$$

Ako je skup H rekurzivan, za teoriju $Th(H)$ kažemo da je *aksiomatibilna*. Za teoriju $\mathcal{T}$ nad jezikom $\mathcal{L}$ kažemo da je *neprotivrečna* ako ne postoji rečenica α jezika $\mathcal{L}$ takva da je $\alpha \in \mathcal{T}$ i $\neg\alpha \in \mathcal{T}$. Za teoriju $\mathcal{T}$ nad jezikom $\mathcal{L}$ kažemo da je *potpuna* ako za svaku rečenicu α jezika $\mathcal{L}$ važi ili $\alpha \in \mathcal{T}$ ili $\neg\alpha \in \mathcal{T}$. Rečenicu α jezika $\mathcal{L}$ za koju važi $\alpha \in \mathcal{T}$ zovemo teoremom teorije $\mathcal{T}$.

Definicija 2.2 *Neka je* $\mathfrak{A}$ *struktura koja odgovara jeziku* $\mathcal{L}$. Teorija $Th(\mathfrak{A})$ *je skup svih rečenica nad jezikom* $\mathcal{L}$ *koje su tačne u* $\mathfrak{A}$*, tj.*

$$Th(\mathfrak{A}) = \{\alpha \mid \mathfrak{A} \models \alpha\} .$$

Ako je $\mathfrak{K}$ *klasa struktura istog tipa i* $\mathcal{L}$ *jezik koji im odgovara, onda je*

$$Th(\mathfrak{K}) = \bigcap_{\mathfrak{A} \in \mathfrak{K}} Th(\mathfrak{A}) .$$

Struktura $\mathfrak{A}$ je *struktura teorije* $\mathcal{T}$ (ili *model teorije* $\mathcal{T}$), ako im odgovara isti jezik i ako su sve rečenice teorije $\mathcal{T}$ nad tim jezikom tačne u $\mathfrak{A}$. Skup svih modela teorije $\mathcal{T}$ označavamo sa $Mod(\mathcal{T})$. Očigledno, važi $\mathfrak{K} \subseteq Mod(Th(\mathfrak{K}))$. Ako je aksiomatski zadata teorija $\mathcal{T}$ potpuna i ako su sve njene aksiome tačne u nekoj strukturi $\mathfrak{A}$, onda je $\mathcal{T} = Th(\mathfrak{A})$.

Ako je $\mathcal{T}$ teorija nad jezikom $\mathcal{L}$, onda pod $\mathcal{T}$-termom, $\mathcal{T}$-formulom itd. podrazumevamo $\mathcal{L}$-term, $\mathcal{L}$-formulu itd.

Za teoriju kažemo da je bez kvantifikatora ako su sve njene aksiome (i formule) (implicitno) univerzalno zatvorene formule bez kvantifikatora. Da bismo dokazali da je formula F teorema teorije $\mathcal{T}$ bez kvantifikatora, dovoljno je dokazati da je $\neg F$ nezadovoljiva u $\mathcal{T}$ ili dokazati da je $\mathcal{T} \cup \neg F$ neprotivrečno.

2.5 Odlučivost teorije

Ukoliko postoji efektivni algoritam A (postupak, procedura) takav da za svaku rečenicu α daje odgovor *da* ako i samo ako je α teorema teorije $\mathcal{T}$ (i *ne* inače) onda kažemo da je teorija $\mathcal{T}$ odlučiva, a algoritam A zovemo *procedurom odlučivanja* za teoriju $\mathcal{T}$. Ukoliko postoji algoritam A takav da za svaku teoremu α teorije $\mathcal{T}$ daje odgovor *da* (a inače je odgovor *ne* ili nedefinisan) kažemo da je teorija $\mathcal{T}$ poluodlučiva, a algoritam A zovemo procedurom poluodlučivanja.

Postoji nekoliko formalizama kojima se strogo uvodi pojam izračunljivostil; na primer: UR mašine, Turingove mašine, Postove mašine, rekurzivne funkcije, Markovljevi algoritmi. Za navedene formalizme može se dokazati da su ekvivalentni. Churchova teza tvrdi da je klasa intuitivno i neformalno izračunljivih funkcija identična sa tim, strogo zasnovanim klasama izračunljivih funkcija.

Za formalno uveden pojam odlučivosti i pregled svojstava odlučivih teorija videti dodatak A.

2.6 Proširenje teorije i unija teorija

Ako za jezike prvog reda $\mathcal{L}^e = \langle T^e, V^e, \Sigma^e, \Pi^e, d^e \rangle$; i $\mathcal{L} = \langle T, V, \Sigma, \Pi, d \rangle$ važi $T \subseteq T^e$, $\Sigma \subseteq \Sigma^e$, $\Pi \subseteq \Pi^e$ i $V \subseteq V^e$, kao i $d^e(f) = d(f)$ za $f \in \Sigma$, $d^e(p) = d(p)$ za $p \in \Pi$, kažemo da je jezik $\mathcal{L}^e$ proširenje jezika $\mathcal{L}$. Ako je $\mathcal{T}$ teorija nad jezikom $\mathcal{L}$, $\mathcal{T}^e$ teorija nad jezikom $\mathcal{L}^e$ i ako važi $\mathcal{T} \subseteq \mathcal{T}^e$ onda kažemo da je teorija $\mathcal{T}^e$ *proširenje* teorije $\mathcal{T}$ (i da je $\mathcal{T}$ *podteorija* teorije $\mathcal{T}^e$). Za teoriju $\mathcal{T}^e$ kažemo da je

konzervativno proširenje teorije $\mathcal{T}$ ako je presek skupa formula jezika $\mathcal{L}$ i skupa $\mathcal{T}^e$ upravo skup $\mathcal{T}$. Svako definicijsko proširenje teorije je konzervativno. Za teoriju $\mathcal{T}^e$ kažemo da je *konačno proširenje teorije* $\mathcal{T}$ ako postoji konačan skup A teorema teorije $\mathcal{T}^e$ takav da je svaka teorema te teorije izvodiva iz skupa koji čine teoreme teorije $\mathcal{T}$ i rečenice iz skupa A.

Najmanje proširenje $\mathcal{T}$ zajedničko za dve teorije $\mathcal{T}_1$ i $\mathcal{T}_2$ nazivamo *unijom* (ili *kombinacijom*) te dve teorije. Unija $\mathcal{T}$ dve teorije $\mathcal{T}_1$ i $\mathcal{T}_2$ je opisana sledećim uslovima:

- komponente jezika teorije $\mathcal{T}$ su unije odgovarajućih komponenti jezika $\mathcal{T}_1$ i $\mathcal{T}_2$;
- rečenica je valjana u $\mathcal{T}$ ako i samo ako je izvodiva iz skupa rečenica koje su valjane u $\mathcal{T}_1$ ili u $\mathcal{T}_2$ (tj. njen skup rečenica je deduktivno zatvorenje rečenica teorija $\mathcal{T}_1$ i $\mathcal{T}_2$).

Unija više teorija definiše se analogno. Unija teorija je netrivijalna operacija, jer svojstva kao što su neprotivrečnost nisu neposredno prenosiva. Neprotivrečnost teorije dobijene kombinacijom više neprotivrečnih teorija može biti dokazana u nekim slučajevima. U daljem tekstu dajemo nekoliko uslova dovoljnih za neprotivrečnost unije dve neprotivrečne teorije (za dokaze videti [120]). Analogni rezultati jednostavno se proširuju na unije više teorija.

Teorema 2.1 *Neka su* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *neprotivrečne teorije i neka su njihove signature* $\mathcal{L}_1$ *i* $\mathcal{L}_2$ *disjunktne. Njihova unija je neprotivrečna ako i samo ako postoji kardinalni broj* κ *takav da i* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *imaju model kardinalnosti* κ*.*

Teorema 2.2 *Neka su* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *neprotivrečne teorije i neka su njihove signature* $\mathcal{L}_1$ *i* $\mathcal{L}_2$ *redom. Njihova unija je neprotivrečna ako i samo ako postoji model* $\mathcal{M}_1$ *teorije* $\mathcal{T}_1$ *i model* $\mathcal{M}_2$ *teorije* $\mathcal{T}_2$ *takav da su njihove restrikcije na na* $\mathcal{L}_1 \cap \mathcal{L}_2$ *izomorfne.*

Teorema 2.3 (Robinsonova teorema o neprotivrečnosti) *Neka su* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *neprotivrečne teorije i neka su njihove signature* $\mathcal{L}_1$ *i* $\mathcal{L}_2$ *redom. Teorijq* $\mathcal{T}_1 \cup \mathcal{T}_2$ *je neprotivrečna ako i samo ako je* $\mathcal{T}_1 \cap \mathcal{T}_2$ *potpuna teorija za* $\mathcal{L}_1 \cap \mathcal{L}_2$ *rečenice.*

Teorema 2.4 *Neka su* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *neprotivrečne teorije i neka su njihove signature* $\mathcal{L}_1$ *i* $\mathcal{L}_2$ *disjunktne. Tada, ako obe te teorije imaju beskonačne modele, onda je njihova unija neprotivrečna.*

O pregledu svojstava unija teorija čije signature nisu disjunktne videti [119, 121].

Ako dve teorije imaju neprazan presek, ali nijedna od njih nije sadržana u drugoj, sledeći model–teoretski rezultat (videti npr. [23]) govori o njihovim vezama koje mogu biti značajne u korišćenju procedura odlučivanja:

Teorema 2.5 *Neka su* $\mathcal{T}_1$ *i* $\mathcal{T}_2$ *dve teorije jezika* $\mathcal{L}$ *koje imaju modele* $\mathcal{M}_1$ *i* $\mathcal{M}_2$*. Ako je* $\mathcal{M}_2$ *podstruktura strukture* $\mathcal{M}_1$*, onda važi:*

- *ako je $\mathcal{T}_1$ potpuna, onda je svaka egzistencijalna rečenica koja je $\mathcal{T}_2$–teorema istovremeno i $\mathcal{T}_1$–teorema*

- *ako je $\mathcal{T}_2$ potpuna, onda je svaka univerzalna rečenica koja je $\mathcal{T}_1$–teorema istovremeno i $\mathcal{T}_2$–teorema.*

Teorija PRA dopušta model $\langle \mathbf{Q}, 0, +, \leq \rangle$, a teorija PIA model $\langle \mathbf{Z}, 0, +, \leq \rangle$; kako je druga struktura podstruktura prve, na osnovu navedene teoreme sledi da je svaka univerzalna rečenica koja je PRA teorema, istovremeno i PIA teorema. Odatle dalje sledi da proceduru odlučivanja za teoriju PRA možemo da koristimo kao saglasnu (mada ne i kompletnu) proceduru i za PIA.

2.7 Sistemi za prezapisivanje termova

Pravilo prezapisivanja je orijentisana jednakost oblika $l \longrightarrow r$. Pravilo $l \to r$ prezapisuje term t u term s ako postoji podterm t' terma t i supstitucija ϕ takva da je term $l\phi$ jednak termu t' i term s je dobijen na osnovu terma t zamenjivanjem terma t' termom $r\phi$. Sistem za prezapisivanje $\mathcal{R}$ je konačni skup pravila prezapisivanja. Sistem za prezapisivanje $\mathcal{R}$ je zaustavljajući ako i samo ako postoji uređenje svođenja (tj. zaustavljajuće uređenje) $\prec$ takvo da za svako pravilo $l \to r$ važi $r \prec l$. Za parcijalno, dobro zasnovano uređenje $\geq$ skupa kažemo da je zaustavljajuće uređenje (ili uređenje svođenja) ako iz $s > t$ za svaku supstituciju φ sledi $s\varphi > t\varphi$ i $f(t_1, \ldots, s, \ldots, t_n) > f(t_1, \ldots, t, \ldots, t_n)$, gde je $>$ nerefleksivni (strogi) deo uređenja $\geq$. Ako je data relacija prethođenja nad funkcijskim i predikatskim simbolima, ona se može proširiti do uređenja svođenja $\prec$ nad termovima [37]. Uslovno pravilo prezapisivanja je pravilo oblika $l \longrightarrow r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$, gde su uslovi $p_1, p_2, \ldots, p_k$ literali. Pravilo $l \to r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$ prezapisuje term t u drugi term s ako postoji podterm t' terma t i supstitucija ϕ takav da je term $l\phi$ jednak termu t', s je term dobijen od terma t zamenjivanje podterma t' termom $r\phi$ i svaki od uslova $p_i\phi$ se svodi na $\top$ (takođe pravilima prezapisivanja). Sistem $\mathcal{R}$ uslovnih pravila prezapisivanje je zaustavljajući ako postoji uređenje svođenja $\prec$ takvo da za svako pravilo $l \longrightarrow r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$, važi $r \prec l$ i $p_i \prec l$ $(1 \leq i \leq k)$.

Term t je *maksimalan* u formuli f ako za bilo koji drugi term t' u formuli f ne važi $t \prec t'$. Za term t kažemo da je *strani term* u odnosu na teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ ako on nije term unije ovih teorija. Term t je *maksimalan strani term* u formuli f u odnosu na teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ ako je on maksimalan term među stranim termovima u f u odnosu na $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$. Literal l je *maksimalan* u konjunkciji literala f ako za bilo koji drugi literal l' iz formule f ne važi $l \prec l'$. Literal l zovemo *strani literal* u odnosu na $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ ako l nije literal unije ovih teorija.

Za detaljni pregled svojstava sistema za prezapisivanje videti dodatak B.

2.8 Jednakosni sistemi

Jednakosno rezonovanje je često od vitalnog značaja u dokazivanju teorema. Jednakosno izvođenje zasnovano je na aksiomama refleksivnosti, simetričnosti, tranzitivnosti i supstitucije. Teoriju zasnovanu na ovim aksiomama zovemo *čista teorija jednakosti*. U opštem slučaju, problem ispitivanja da li jedna jednakost sledi iz datog skupa jednakosti nije odlučiv. Specijalno, problem ispitivanja da li jedna bazna jednakost sledi iz datog skupa baznih jednakosti je odlučiv. Taj problem može biti rešavan direktnom primenom aksioma jednakosti, ali znatno efikasnije tehnikama prezapisivanja (generisanjem kanonskog sistema na osnovu datog skupa baznih jednakosti) ili algoritmima za određivanje kongruentnog zatvorenja. Postoji više algoritama za određivanje kongruentnog zatvorenja [104, 87], i oni se često koriste kao osnova za efikasno kombinovanje procedura odlučivanja [104, 32, 9, 11].

U dodatku C dat je detaljniji pregled jednakosnih sistema i algoritama za određivanje kongruentnog zatvorenja.

2.9 Prezburgerova aritmetika

U dokazivanju teorema, uloga procedura odlučivanja za delove aritmetike je često od suštinskog značaja zbog njene uloge u problemima verifikacije softvera i hardvera. Cela aritmetika je neodlučiva, ali ona ima značajne odlučive fragmente kao što su Prezburgerova aritmetika i strogo multiplikativna aritmetika [97, 107, 84].

U Prezburgerovoj aritmetici (nad prirodnim brojevima) — PNA sve promenljive su sorte $\mathbf{N}$ i $\Sigma = \{0, s, +\}$, $(0 : \mathbf{N},\ s : \mathbf{N} \to \mathbf{N},\ + : \mathbf{N} \times \mathbf{N} \to \mathbf{N})$, $\Pi = \{<, >, \leq, \geq\}$ (svim predikatima odgovara struktura $\mathbf{N} \times \mathbf{N}$). Umesto $s(0)$ pisaćemo 1, umesto $s(s(0))$ pisaćemo 2 itd. Može se smatrati da i množenje konstantom pripada jeziku PNA: nx je skraćeni zapis za $x+\cdots+x$, gde se x pojavljuje n puta. Aksiome teorije PNA su aksiome Peanove aritmetike bez aksioma koje se odnose na množenje. Slično uvodimo Prezburgerovu aritmetiku nad celim brojevima — PIA i Prezburgerovu aritmetiku nad racionalnim brojevima — PRA. Prezburger je prvi dokazao da je teorija PIA odlučiva [97]. Odlučivost teorije PNA se može dokazati na analogan način. Teorija PRA je takođe odlučiva [72].

Teorija PIA je potpuna i njen skup aksioma je rekurzivan, pa na osnovu teoreme A.1, sledi da je ona odlučiva. Nama će od većeg značaja biti dokazi odlučivosti Prezburgerove aritmetike zasnovani na eliminaciji kvantifikatora, koji indukuju odgovarajuće procedure odlučivanja. Neke od procedura odlučivanja za Prezburgerovu aritmetiku zasnovani na eliminaciji kvantora su Cooperova procedura za PIA [29] i Hodesova za PRA [52] (koja je suštinski zasnovana na Furijeovom metodu za rešavanje linearnih nejednakosti nad racionalnim brojevima [74]). Procedure za PRA se u automatskom rezonovanju često koriste kako bi se izbegla znatno veća kompleksnost procedura odlučivanja za PIA. Na osnovu teoreme 2.5, procedura odlučivanja za PRA daje saglasne odgovore za sve univerzalne PIA rečenice. Dakle, ako je za neku univerzalnu PIA rečenicu procedura

odlučivanja za PRA daje potvdan odgovor, onda je ta rečenica zaista teorema teorije PIA. Međutim, ne važi obratno jer postoje univerzalno kvantifikovana tvrđenja koja su teoreme nad celim brojevima, ali ne i nad racionalnim.

Detaljniji pregled svojstava Prezburgerove aritmetike dat je u dodatku D.

2.10 Background and Notation: Summary

In the rest of the thesis we will (mostly) deal with many–sorted first-order theories and we will use a standard classical logic inference system. In the rest of the thesis we also use standard notions of decidable theories, term rewriting systems, equational systems and Presburger arithmetic.

A theory $\mathcal{T}$ is decidable if there is an algorithm (which we call a *decision procedure*) such that for an input $\mathcal{T}$-sentence F, it returns `true` if and only if $\mathcal{T} \vdash F$ (and returns `false` otherwise). A more detailed account on decidability is given in Appendix A.

A *rewrite rule* is an oriented equation of the form $l \rightarrow r$. The rule $l \rightarrow r$ rewrites a term t to another term s if there is a subterm t' of t and a substitution ϕ such that $l\phi$ is equal to t' and s is t with the subterm t' replaced by $r\phi$. A *rewrite system* $\mathcal{R}$ is a finite set of rewrite rules. A rewrite system $\mathcal{R}$ is terminating if and only if there is a reduction ordering $\prec$ such that $r \prec l$ for each rule $l \rightarrow r$. When a precedence relation on function and predicate symbols is given, it can be extended to a reduction ordering $\prec$ on terms. A term t is *maximal* in a formula f if for any other term t' in formula f it does not hold that $t \prec t'$. A term t is called an *alien term* w.r.t. $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ if it is not a term of these theories. A term t is *maximal alien term* in a formula f w.r.t. $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ if it is maximal term among alien terms in f w.r.t. $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$. A literal l is *maximal* in a conjunction of literals f if for any other literal l' in formula f it does not hold $l \prec l'$. A literal l is called *alien literal* w.r.t. $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ if it is not a literal of these theories. A more detailed account on term rewriting systems is

given in Appendix B.

Special attention in the rest of the thesis will be given to the pure theory of equality, i.e. to the theory of equality with "uninterpreted" functions and predicates, i.e., functions and predicates with no information about them (other than information about their sorts). Its axioms are the axioms of reflexivity, symmetry, transitivity and substitutivity. The problem whether some equality is implied by some set of equalities can be treated by the use of these axioms or, more efficiently, by the ground completion procedure or by congruence closure algorithms. A more detailed account on equality reasoning is given in Appendix C.

In the rest of the paper, special attention will also be given to Presburger arithmetic (according to its significance in verification problems). Presburger Natural arithmetic (PNA) is (roughly speaking) a theory built up from the constant 0, variables, binary $+$, unary s, relations $<$, $>$, $=$, $\leq$, $\geq$ and the standard connectives and quantifiers of first-order predicate calculus. We write 1 in-

stead of $s(0)$, 2 instead of $s(s(0))$ etc. Multiplication by a constant can also be considered as in PNA: nx is treated as $x + \cdots + x$, where x appears n times. The axioms of PNA are those of Peano arithmetic without axioms concerning multiplication. Similarly we introduce Presburger arithmetic over integers — *Presburger Integer Arithmetic* (PIA) and Presburger arithmetic over rationals — and *Presburger Rational Arithmetic* (PRA). It was Presburger who first showed that PIA is decidable [97]. The decidability of PNA can be proved in an analogous way. PRA is also decidable [72]. A more detailed account on Presburger arithmetic is given in Appendix D.

Glava 3

Sheme za ugradnju procedura odlučivanja u dokazivače teorema

Tokom sedamdesetih i početkom osamdesetih godina obnovljeno je interesovanje za procedure odlučivanja za jednostavne teorije i za njihovu upotrebu u dokazivanju teorema. Ovaj interes ojačan je značajnim shemama za kombinovanje procedura odlučivanja predloženim od Nelsona i Oppena [86] i od Shostaka [104]. Drugi značajan pristup — proširenje procedure odlučivanja dodatnim, raspoloživim lemama objavili su Boyer i Moore i uspešno primenili u svom dokazivaču NQTHM [17]. U ovom delu teze dajemo opis ove tri sheme, kao i opis shema primenjenih u sistemu TECTON (Kapur i Nie, [68]), u ograničenom kontekstualnom prezapisivanju (Armando i Ranise, [4]) i u sistemu SVC (Barrett, Dill i Stump, [10]). Ove sheme poslužile su kao motivacija za dva fleksibilna okvira koji se predlažu u ovoj tezi.

Sve metode koje opisujemo u nastavku ove glave koriste se za teorije bez kvantifikatora i to uglavnom u formi dokazivanja nezadovoljivosti zadate (implicitno egzistencijalno zatvorene) formule.

3.1 Nelson/Oppenov algoritam za kombinovanje procedura odlučivanja

Nelson/Oppenova shema [86] jedan je od najranijih i jedan od najopštijih metoda za kombinovanje teorija. U vremenu koje je sledilo objavljivanje ove procedure, najviše istraživanja uloženo je u jednakosne teorije i algoritme unifikacije, dok su pokušaji uopštavanja Nelson/Oppenovog metoda bili ređi (videti [120]).

Nelson/Oppenov metod pokazuje kako je na osnovu procedura odlučivanja za nekoliko teorija prvog reda moguće dobiti i proceduru odlučivanja za njihovu uniju. Osnovna ideja metoda je da se komunikacija između procedura

odlučivanja odvija preko jednakosti odgovarajućih teorija.

Dokaz korektnosti metoda koji je dat u originalnom radu [86] ima grešaka, a u kasnijim opisima metoda [91, 88], isti autori se u dokazima, zbog jednostavnosti, ograničavaju samo na neke slučajeve, ali čemu opštije dokaze nije moguće neposredno izvesti. Opis metoda i tvrđenja na kojima se zasniva njegova korektnost bazirani su na radu [120].

3.1.1 Jednostavna konjuktivna normalna forma

Formula je u *jednostavnoj konjuktivnoj normalnoj formi* (eng. simple Conjunctive Normal Form) ako je ona konjunkcija literala (atomičkih formula ili negacija atomičkih formula). Neka su date teorije prvog reda sa jednakošću $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_n$ čije su signature $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_n$ uzajamno disjunktne[1]. Za svaku teoriju $\mathcal{T}_i$, neka je $sCNF(\mathcal{T}_i)$ klasa formula teorije $\mathcal{T}_i$ koje su u jednostavnoj konjuktivnoj normalnoj formi. Neka je $\mathcal{T}$ unija teorija $\mathcal{T}_i$, tj. deduktivno zatvorenje unije teorija $\mathcal{T}_i$. Neka je $\mathcal{L}$ signatura teorije $\mathcal{T}$ i neka je $sCNF(\mathcal{T})$ klasa formula teorije $\mathcal{T}$ koje su u jednostavnoj konjuktivnoj normalnoj formi.

Ako je za svaku teoriju $\mathcal{T}_i$ raspoloživa procedura odlučivanja za formule klase $sCNF(\mathcal{T}_i)$, onda se može konstruisati procedura odlučivanja za formule klase $sCNF(\mathcal{T})$.

3.1.2 Razdvojena forma

Zbog mogućih "mešanih" termova (tj. termova izgrađenih od simbola iz više signatura), na zadatu formula ϕ iz klase $sCNF(\mathcal{T})$ ne može se, u opštem slučaju, primeniti nijedna od procedura za klase $\mathcal{T}_i$. Te procedure moguće je upotrebiti ako formulu ϕ transformišemo u oblik $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$, koji zovemo *razdvojena forma* (eng. separate form), pri čemu svaka od podformula ϕ_i pripada nekoj od klasa $sCNF(\mathcal{T}_j)$. Ukoliko je neka od podformula nezadovoljiva u odgovarajućoj teoriji (npr. ako je ϕ_i nezadovoljivo u $\mathcal{T}_j$, pri čemu je $\phi_i \in sCNF(\mathcal{T}_j)$), onda je ona (pa i formula ϕ) nezadovoljiva i u $\mathcal{T}$. Dakle, nezadovoljivost formule ϕ moguće je tada dokazati korišćenjem procedura odlučivanja za $sCNF(\mathcal{T}_j)$. Jednostavnosti u izlaganju radi, nećemo praviti razliku između konjunkcije $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$ i multiskupa $\{\phi_1, \phi_2, \ldots, \phi_m\}$.

Ako ϕ nije u razdvojenoj formi, primenjujemo sledeći postupak:

1. **Apstrakcija stranih termova** U formuli ϕ rekurzivno zameni svaki strani term t novom promenljivom z (odgovarajuće sorte) i dodaj (kao konjunkt) jednakost $z = t$ u formulu ϕ. Svaku jednakost oblika $t_1 = t_2$ zameni jednakostima $z = t_1$ i $z = t_2$ (gde je z nova promenljiva odgovarajuće sorte).

2. **Particionisanje** Podeli novodobijeni skup literala u $m \leq n$ blokova koji sadrže samo $\mathcal{T}_i$ literale (jednakosti između promenljivih mogu da pripadnu bilo kojoj grupi literala za koju odgovarajuća teorija dopušta jednakost za sortu tih promenljivih).

[1] Primetimo da simbol jednakosti u signaturi teorija sa jednakošću ne spada u signaturu.

Rezultat navedenog postupka je $sCNF$ formula oblika $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$ gde je svaka od formula ϕ_i formula neke klase $sCNF(\mathcal{T}_j)$ za neko $j \in \{1, 2, \ldots, n\}$. Formula ϕ može da ima više razdvojenih formi, ali su sve one ekvivalentne do na imena varijabli i do na standardna svojstva konjunkcije i jednakosti; sve te forme smatraćemo jednakim, pa se zato može govoriti o razdvojenoj formi formule ϕ, koju ćemo označavati sa $\hat{\phi}$. Zbog notacijske jednostavnosti, možemo da smatramo da je razdvojena forma formule ϕ oblika $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ gde je formula ϕ_i formula klase $sCNF(\mathcal{T}_i)$ (pri tome, neke od formula ϕ_i mogu da budu jednake $\top$).

Jednostavno se dokazuje da je za svaku formulu ϕ ($\phi \in sCNF(\mathcal{T})$) ϕ ekvivalentno formuli $\exists \vec{x} \hat{\phi}$, gde $\vec{x}$ označava nove promenljive uvedene procedurom razdvajanja. Dakle, važi sledeće tvrđenje:

Teorema 3.1 *$sCNF$ formula je zadovoljiva ako i samo ako je zadovoljiva njena razdvojena forma.*

Osnovni problem u ispitivanju da li je formula ϕ zadovoljiva ispitivanjem zadovoljivosti podformula ϕ_i (koje čine njenu razdvojenu formu) je sledeći: svaka od podformula ϕ_i može da bude zadovoljiva, a da pri tom njihova konjunkcija nije zadovoljiva. Dakle, da bi se omogućila ispravna primena pojedinačnih procedura odlučivanja neopohodno je omogućiti njihovu komunikaciju. U Nelson/Oppenovom metodu, ta komunikacija se ostvaruje prosleđivanjem iz jedne u druge procedure svih izvedenih jednakosti između promenljivih formule $\hat{\phi}$.

U opštem slučaju metod je nešto složeniji jer je moguće da u nekom koraku sledi disjunkcija jednakosti (ali ne i bilo koja od tih jednakosti pojedinačno) i tada je neophodno ići na razmatranje slučajeva. To je moguće samo u teorijama koje su *nekonveksne*.

3.1.3 Konveksne i nekonveksne teorije

Definicija 3.1 *Formula f je* nekonveksna *ako f povlači $x_1 = y_1 \vee x_2 = y_2 \vee \ldots x_n = y_n$, ali ni za jedno i između 1 i n formula F ne povlači $x_i = y_i$. Inače, formula f je* konveksna. *Teorija $\mathcal{T}$ je* konveksna *ako je svaka konjunkcija njenih literala konveksna.*

Na primer, konveksne su teorije PRA, teorija jednakosti sa neinterpretiranim funkcijskim simbolima (tj. čista teorija jednakosti) i teorija lista čije su aksiome:

1. $cons(car(x), cdr(x)) = x$
2. $car(cons(x, y)) = x$
3. $cdr(cons(x, y)) = y$

Na primer, nekonveksne su teorije PIA, strogo multiplikativna racionalna aritmetika i teorija nizova sa *store* i *select* čije su aksiome:

1. $select(store(v, i, e), j) = \text{if } i = j \text{ then } e \text{ else } select(v, j)$
2. $store(v, i, select(v, i)) = v$
3. $store(store(v, i, e), i, f) = store(v, i, f)$
4. $i \neq j \rightarrow store(store(v, i, e), j, f) = store(store(v, j, f), i, e)$

Teorija PIA je nekonveksna jer, na primer, formula $x = 1 \wedge y = 2 \wedge 1 \leq z \wedge z \leq 2$ povlači formulu $x = z \vee y = z$, ali ne povlači ni $x = z$ ni $y = z$. Slično, strogo mulitplikativna racionalna aritmetika je nekonveksna, jer na primer, formula $xy = 0 \wedge z = 0$ povlači formulu $x = z \vee y = z$, ali ne povlači ni $x = z$ ni $y = z$.

U radu [91] detaljno se razmatra uticaj konveksnosti na složenost metoda. Razmatranje slučajeva može se izbeći *nedeterminističkom* verzijom Nelson/-Oppenovog metoda.

3.1.4 Deterministička verzija algoritma

Neka su $\mathcal{T}_1$ i $\mathcal{T}_2$ dve teorije čije su signature disjunktne. Neka je $\mathcal{T}$ njihova unija i neka je ϕ formula koja pripada klasi $sCNF(\phi)$. Originalni Nelson/Oppenov algoritam [86] ispituje zadovoljivost formule ϕ na sledeći način:

1. transformiši formulu ϕ u razdvojenu formu (na način opisan u 3.1.2) $\phi_1 \wedge \phi_2$;
2. Ako je ϕ_1 ili ϕ_2 nezadovoljiva, vrati rezultat *nezadovoljiva*;
3. Ako ϕ_1 ili ϕ_2 povlači neku jednakost između promenljvih koju ne povlači druga formula, onda dodaj tu jednakost kao novi literal u formulu koja je ne povlači; idi na korak 2;
4. Ako ϕ_1 ili ϕ_2 povlači neku disjunkciju $u_1 = v_1 \vee u_2 = v_2 \vee \ldots \vee u_k = v_k$, ali pri tom ne povlači nijednu jednakost pojedinačno, onda rekurzivno primeni celu proceduru na formule $\phi_1 \wedge \phi_2 \wedge u_1 = v_1$, ..., $\phi_1 \wedge \phi_2 \wedge u_k = v_k$. Ako je bilo koja od ovih formula zadovoljiva, vrati rezultat *zadovoljiva*. Inače vrati rezultat *nezadovoljiva*.
5. Vrati rezultat *nezadovoljiva*.

Procedura za unije više od dve teorije dobija se jednostavnim uopštavanjem.

Primer 3.1 *Razmotramo uniju teorije* PRA*, teorije jednakosti sa neinterpretiranim funkcijskim i predikatskim simbolima (tj. čiste teorije jednakosti; u oznaci $\mathcal{E}$) i teorije listi (u oznaci $\mathcal{L}$). Treba dokazati da je sledeća formula nezadovoljiva:*

$$x \leq y \ \wedge \ y \leq x + car(cons(0, x)) \ \wedge \ p(h(x) - h(y)) \ \wedge \ \neg p(0)$$

Nakon razdvajanja (tj. nakon apstrakcije stranih termova i particionisanja), dobija se sledeći skup literala:

ϕ_{pra}	$\phi_{\mathcal{E}}$	$\phi_{\mathcal{L}}$
$x \leq y$	$p(v_2) = \top$	$v_1 = car(cons(v_5, x))$
$y \leq x + v_1$	$p(v_5) = \bot$	
$v_2 = v_3 - v_4$	$v_3 = h(x)$	
$v_5 = 0$	$v_4 = h(y)$	

Svaka od tri konjunkcije je pojedinačno zadovoljiva, pa mora da postoji međusobna interakcija — razmena implicitnih jednakosti da bi nezadovoljivost bila detektovana. Konjunkcija $\phi_{\mathcal{L}}$ povlači jednakost $v_1 = v_5$ i ona se prosleđuje. Konjunkcija ϕ_{pra} koristi ovu jednakost i povlači $x = y$. Nakon toga, $\phi_{\mathcal{E}}$ povlači $v_3 = v_4$. Konjunkcija ϕ_{pra} onda povlači $v_2 = v_5$. Konačno, konjunkcija $\phi_{\mathcal{E}}$ sadrži dva protivrečna literala i vraća rezultat nezadovoljiva *(čime je dokazana i nezadovoljivost polazne formule). Naredna tabela prikazuje polazne literale i prosleđene jednakosti u poretku u kojem su one izvođene:*

ϕ_{pra}	$\phi_{\mathcal{E}}$	$\phi_{\mathcal{L}}$
$x \leq y$	$p(v_2) = \top$	$v_1 = car(cons(v_5, x))$
$y \leq x + v_1$	$p(v_5) = \bot$	
$v_2 = v_3 - v_4$	$v_3 = h(x)$	
$v_5 = 0$	$v_4 = h(y)$	
		$v_1 = v_5$
$x = y$		
	$v_3 = v_4$	
$v_2 = v_5$		
	nezadovoljivo	

3.1.5 Nedeterministička verzija algoritma

Umesto pojedinačnog uvođenja jednakosti i razmatranja slučajeva u nekonveksnim teorijama, moguće je nedeterministički unapred pretpostaviti sve jednakosti koje važe između promenljivih iz polazne formule [120]. Pre opisa procedure uvešćemo pojam *rasporeda*.

Neka je P bilo koja particija skupa S i neka je R odgovarajuća relacija ekvivalencije. *Rasporedom* (eng. arrangement) skupa S zadatim particijom P nazivamo skup

$$ar(S) = \{x = y \mid x, y \in S \wedge xRy\} \cup \{x \neq y \mid x, y \in S \wedge \neg(xRy)\}$$

koji sadrži (do na refleksivnost i simetričnost) sve jednakosti između dva ekvivalentna elementa i sve nejednakosti između dva neekvivalentna elementa iz S. Na primer, ako je $S = \{x_1, x_2, x_3, x_4\}$ i $P = \{\{x_1, x_2, x_3\}, \{x_3\}\}$, onda je

$$ar(S) = \{x_0 = x_1, x_0 = x_2, x_1 = x_2, x_0 \neq x_3, x_1 \neq x_3, x_0 \neq x_3\}$$

U nastavku teksta kostićemo rasporede nad promenljivama (pri čemu, naravno, u istoj klasi mogu da budu samo promenljive iste sorte). Za skup promenljivih

X, $ar(X)$ je skup formula, ali ćemo ga, kada je to, pogodno, tretirati i kao konjunkciju formula.

Neka su $\mathcal{T}_1$ i $\mathcal{T}_2$ dve teorije sa disjunktnim signaturama, neka je $\mathcal{T}$ njihova unija i neka je ϕ formula koja pripada klasi $sCNF(\phi)$. Sledeći nedeterministički algoritam [120] ispituje zadovoljivost formule ϕ:

— **Faza dekompozicije**

1. Neka je ϕ transorfmisana u razdvojenu formu i neka je ϕ_i $(i = 1, 2)$ deo te forme koji pripada teoriji $\mathcal{T}_i$
2. Neka je X skup promenljivih koji dele literali u ϕ_1 i ϕ_2; odredi (nedeterministički) jedan raspored $ar(X)$.
3. Predaj par $\langle \phi_1 \cup ar(X), \phi_2 \cup ar(X) \rangle$ sledećoj fazi

— **Faza provere**

1. Ispitaj zadovoljivost $\phi_1 \cup ar(X)$
2. Ispitaj zadovoljivost $\phi_2 \cup ar(X)$
3. Vrati rezultat *zadovoljiva* ako su oba dobijena rezultata potvrdna. Inače, završi rad bez uspeha.

Formula ϕ je nezadovoljiva ako i samo ako ne postoji raspored za koji navedena procedura vraća odgovor *zadovoljivo*. Kako je svaka implementacija nužno deterministička, u implementaciji navedene procedure potrebno je podržati mehanizam ispitivanja svih rasporeda. Opisani algoritam moguće je implementirati i na inkrementalan način:

Neka je L proizvoljna permutacija literala u razdvojenoj formi zadate formule. U jednom koraku, prosleđuje se po jedan literal iz L procedurama odlučivanja za $sCNF(\mathcal{T}_1)$ i $sCNF(\mathcal{T}_2)$ (i to u zavisnosti od toga da li literal pripada teoriji $\mathcal{T}_1$ ili teoriji $\mathcal{T}_2$). Ako neki literal ima deljene promenljive sa nekim od literala koji je već bio prosleđen drugoj proceduri odlučivanja, onda biramo neki raspored tih promenljivih i prosleđujemo ih i jednoj i drugoj proceduri. Izvršavanje procedure se nastavlja dok se ne detektuje nezadovoljivost jednom ili drugom procedurom odlučivanja ili dok se ne obrade svi literali iz L.

Jedina suštinska razlika između originalne Nelson/Oppenove procedure i navedene nedeterminističke je u obimu jednakosti između promenljivih koje se ispituju: u originalnom radu, ispituju se sve moguće jednakosti, dok je kasnije pokazano [120] da je dovoljno ispitivati jednakosti nad promenljivama koje su deljene između različitih komponenti razdvojene forme formule koja se dokazuje.

3.1.6 Saglasnost, kompletnost i zaustavljanje

U ovom delu navodimo osnovne teoreme kojima se dokazuje ispravnost Nelson/Oppenove procedure. Neke od teorema se odnose na nedeterminističku varijantu procedure, ali se jednostavno mogu primeniti i na osnovnu, originalnu varijantu.

U opisu procedura, ograničava se na kombinaciju teorija sa dijsunktnim signaturama. Ukoliko te teorije imaju modele iste kardinalnosti, onda, na osnovu teoreme 2.1 sledi da je njihova unija neprotivrečna, pa problem zadovoljivosti nije trivijalan. Taj uslov, međutim, nije dovoljan i za dokaz korektnost Nelson/Oppenove procedure. Štaviše, može se konstruisati primer teorija sa disjunktnim signaturama koje imaju samo konačne modele i za koje Nelson/Oppenova procedura nije korektna. S druge strane, može se pokazati da je procedura za kombinaciju korektna ako su teorije koje se kombinuju *stabilno–beskonačne*. Moguće je da to nije potreban uslov za korektnost procedure, ali je najslabiji dovoljan uslov utvrđen do sada [120]. O mogućim uopštenjima Nelson/Oppenove procedure na teorije koje nemaju disjunktne signature videti radove [119, 121].

Definicija 3.2 *Neprotivrečna teorija bez kvantifikatora $\mathcal{T}$ sa signaturom $\mathcal{L}$ je* stabilno–beskonačna *ako važi: za svaku formulu ϕ bez kvantifikatora ϕ, ako je $\mathcal{T} \cup \{\phi\}$ neprotivrečno, onda postoji beskonačan model koji zadovoljava $\mathcal{T} \cup \{\phi\}$.*

O tome da je dovoljno razmatrati samo jednakosne dodatne literale govori Craigova interpolaciona lema [103]:

Teorema 3.2 (Craigova interpolaciona lema) *Ako je $\mathcal{T}_1 \cup \mathcal{T}_2 \models \phi_1 \rightarrow \phi_2$, gde je ϕ_1 formula nad jezikom teorije $\mathcal{T}_1$, a ϕ_2 formula nad jezikom teorije $\mathcal{T}_2$, onda postoji formula ψ, čije slobodne promenljive pripadaju preseku slobodnih promenljivih za ϕ_1 i ϕ_2, takva da je $\mathcal{T}_1 \models \phi_1 \rightarrow \psi$ i $\mathcal{T}_2 \models \psi \rightarrow \phi_2$.*

Teorema 3.3 *Neka su $\mathcal{T}_1$ i $\mathcal{T}_2$ dve stabilno–beskonačne teorije sa disjunktnim signaturama, neka je $\phi_1 \in sCNF(\mathcal{T}_1)$ i $\phi_2 \in sCNF(\mathcal{T}_2)$ i neka je V skup zajedničkih slobodnih promenljvih za ϕ_1 i ϕ_2. Ako je formula $\phi_i \wedge ar(V)$ zadovoljiva u $\mathcal{T}_i$ ($i = 1, 2$), onda je formula $\phi_1 \cup \phi_2$ zadovoljiva u $\mathcal{T}_1 \cup \mathcal{T}_2$.*

U svim narednim teoremama pretpostavljamo da su teorije $\mathcal{T}_1$ i $\mathcal{T}_2$ stabilno–beskonačne i sa disjunktnim signaturama i da je $\mathcal{T}$ unija ove dve teorije.

Teorema 3.4 (Saglasnost) *Ako postoji izlazni par $\langle \psi_1, \psi_2 \rangle$ faze dekompozicije Nelson/Oppenove procedure takav da je ψ_i zadovoljiva u $\mathcal{T}_i$ za $i = 1, 2$, onda je polazna formula zadovoljiva u $\mathcal{T}$.*

Teorema 3.5 (Kompletnost) *Ako je formula $\phi \in sCNF(\mathcal{T})$ zadovoljiva u $\mathcal{T}$, onda postoji izlazni par $\langle \psi_1, \psi_2 \rangle$ faze dekompozicije Nelson/Oppenove procedure takav da je ψ_i zadovoljiva u $\mathcal{T}_i$ za $i = 1, 2$*

Primetimo da za navedena tvrđenja nije bilo potrebno pretpostaviti da je problem zadovoljivosti u teorijama komponentama odlučiv niti da su te teorije aksiomatibilne. Ako je zadovoljivost u teorijama–komponentama odlučiva, onda dobijamo sledeći jači rezultat:

Teorema 3.6 *Pretpostavimo da su problemi zadovoljivosti u $\mathcal{T}_i$ formula iz klasa $sCNF(\mathcal{T}_i)$ ($i = 1, 2$) odlučivi. Onda, formula $\phi \in sCNF(\mathcal{T})$ je zadovoljiva u $\mathcal{T}$ ako i samo ako Nelson/Oppenova procedura vraća rezultat* zadovoljivo.

Navedene teoreme dokazuju da je Nelson/Oppenova procedura korektna ako su teorije komponente stabilno–beskonačne i disjunktnih signatura. Rezultat se jednostavno uopštava na kombinacije više od dve teorije. Zaustavljanje Nelson/Oppenove procedure se dokazuje trivijalno (jer ima konačno mnogo promenljivih koje se razmatraju) i ona predstavlja proceduru odlučivanja za kombinacije teorija koje su stabilno–beskonačne i disjunktnih signatura. Kompleksnost Nelson/Oppenove procedure zavisi od kompleksnosti procedura odlučivanja za teorije komponente.

3.1.7 Primene

Nelson/Oppenova metod za kombinovanje procedura odlučivanja našla je primenu u više uspešnih savremenih dokazivača, od kojih su neki Stanford Pascal Verifier [75] i STeP [76], kao i u logičkom programiranju sa ograničenjima (eng. constraint logic programming) [118].

Zbog njene velike opštosti, Nelson/Oppenovu proceduru moguće je primenjivati za kombinacije velikog broja teorija. Najčešće se koriste Prezburgerova aritmetika, teorija listi, teorija nizova, čista teorija jednakosti.

Iako je nedeterministička verzija procedure (opisana u 3.1.5) pogodnija za teorijska razmatranja, osnovna, deterministička verzija i inkrementalna nedeterministička verzija su pododnije za praktične primene. Implementacija osnovne verzije je posebno jednostavna u slučaju konveksnih teorija komponenti.

U determinističkoj verziji procedure, u slučaju konveksnih teorija određivanje implicitnih jednakosti može se uraditi na sledeći način: neka je ϕ_i formula iz klase $sCNF(\mathcal{T}_i)$ i neka su x i y neke dve promenljive koje se pojavljuju u ϕ_i (a u ϕ_i se ne pojavljuje literal $x = y$); ako je $\phi_i \wedge x \neq y$ nezadovoljivo u $\mathcal{T}_i$, onda je $x = y$ implicitna jednakost i kažemo da ϕ_i povlači jednakost $x = y$. U slučaju nekonveksnih teorija komponenti, situacija je kompleksnija, jer je potrebno razmatrati sve moguće konjunkcije nejednakosti nad promenljivim iz formule ϕ. Opisani postupak daje opšti način za određivanje implicitnih jednakosti (odnosno disjunkcija implicitnih jednakosti). U praktičnim primenama, za konkretne teorije moguće je primeniti efikasnije algoritme za njihovo određivanje. Tako, na primer, u originalnoj proceduri, Nelson i Oppen su za PRA koristili simpleks algoritam.

3.2 Shostakov algoritam za kombinovanje procedura odlučivanja

Shostakov algoritam za kombinovanje procedura odlučivanja je primenljiv na skup teorija znatno uži u odnosu na Nelson/Oppenov algoritam, ali je na račun te restrikcije dobijena značajna efikasnost. Shostakov algoritam primenjuje se na *kanonizabilne* i *algebarski rešive* jednakosne teorije bez kvantifikatora. Može se pokazati da je kombinacija kanonizabilnih i algebarski rešivih jednakosnih teorija bez kvantifikatora ponovo kanonizabilna i algebarski rešiva jednakosna teorija bez kvantifikatora. Za rezovovanje o jednakostima koristi se efikasan

algoritam za kongruentno zatvorenje, koji istovremeno služi i kao "lepak" za kombinovanje više teorija komponenti. Suštinska ideja algoritma je u tome da se literali pojednostavljuju "rešavanjem", a onda da se njihova obrada prepusti kongruentnom zatvorenju.

Opis algoritma koji je dat u originalnom radu [104] ima grešaka i neprecizности. Opis algoritma i tvrđenja na kojima se zasnivaju njegova svojstva bazirani su na radu [32].

3.2.1 Kongruentno zatvorenje

Ako je E skup jednakosti, problem $E \models s = t$ može biti rešavan tehnikama prezapisivanja i Knuth/Bendixovom procedurom upotpunjavanja. Najčešće je, međutim, od tog pristupa efikasniji metod *kongruentnog zatvorenja* [87, 104, 8]. Ovaj metod zasnovan je na ideji primenjivanja aksioma jednakosti, ali suženih na one termove koji se pojavljuju u E, uključujući i njihove podtermove.

Navodimo najpre definiciju *kongruentnog zatvorenja* proizvoljne relacije R, u terminima grafova:

Definicija 3.3 *Neka je $G = (V, D)$ označeni usmereni graf za koji su $\lambda(v)$ i $\delta(v)$ redom oznaka i izlazni stepen čvora v (iz skupa V). Neka je $v[i]$ čvor u takav da je (v, u) i-ta grana sa početnim čvorom v. Ako je α neki skup čvorova, neka je $use(\alpha)$ skup $\{v \in V \mid (\exists i : v[i] \in \alpha)\}$. Za datu binarnu relaciju R nad V, ekvivalentno zatvorenje relacije R (označeno sa R^*) je refleksivno, simetrično i tranzitivno zatvorenje relacije R.* Kongruentno zatvorenje $\hat{R}$ *binarne relacije R je najmanje proširenje relacije R^* takvo da za svaka dva čvora u, v ako važi $\lambda(u) = \lambda(v)$, $\delta(u) = \delta(v)$ i $u[i]\hat{R}v[i]$ za $1 \leq i \leq \delta(u)$, onda važi i $u\hat{R}v$.*

Kongruentno zatvorenje za jednakost može biti definisano na sledeći način:

Definicija 3.4 Kongruentno zatvorenje $\simeq_C$ *generisano skupom jednakosti C je minimalna relacija ekvivalencije koja zadovoljava sledeća svojstva:*

- *za bilo koje dve jednakosti $t_1 = t_2$ of C, važi $t_1 \simeq_C t_2$;*
- *ako je $t_1 \simeq_C t_2$, onda je $f(\ldots t_1 \ldots) \simeq_C f(\ldots t_2 \ldots)$.*

Naglasimo da je navedenim definicijama uvedeno zapravo *konstantno* kongruentno zatvorenje, jer se (eventualne) promenljive obrađuju kao konstante (tj. nije dozvoljeno instanciranje promenljivih).

Neka je, na primer, dati skup jednakosti E jednak $\{w = g(f(a)), f(a) = a\}$ i neka je potrebno dokazati da važi $w = g(a)$. Metod kongruentnog zatvorenja (najčešće) konstruiše graf čiji čvorovi odgovaraju termovima i podtermovima termova iz skupa $\{w, g(f(a)), g(a)\}$. Relacija kongruentnog zatvorenja povezuje (smeštanjem u iste klase ekvivalencije) one čvorove koji odgovaraju datim jednakostima tako da su termovi w i $g(f(a))$ identifikovani, kao i termovi $f(a)$ i a. Dalje, bilo koja dva čvora sa identičnom oznakom čiji sledbenici su identični

su takođe identični. Identifikacija termova a i $f(a)$, dakle, indukuje identifikovanje termova $g(a)$ i $g(f(a))$. Kao rezultat, termovi w i $g(a)$ su u istoj klasi ekvivalencije.

Kongruentno zatvorenje indukovano skupom jednakosti E može biti primenjeno na proizvoljan skup termova T i može biti korišćeno za proveravanje zadovoljivosti skupa jednakosti i nejednakosti. To je ilustrovano sledećim primerom.

Primer 3.2 *Neka je* $T = \{a, b, c, f(a,b), f(b,c), g(a), f(g(a),b)\}$ *i neka je* $E = \{a = b, b = c\}$. *Kongruentno zatvorenje* $\mathcal{C}$ *je onda particija*

$$\{\{a,b,c\}, \{f(a,b), f(b,c)\}, \{g(a)\}, \{f(g(a),b)\}\}$$

Konjunkcija

$$a = b \wedge b = c \wedge f(a,b) \neq g(a) \wedge f(b,c) \neq f(g(a),b)$$

je zadovoljiva jer je svaki par termova u nejednakostima u različitim klasama. S druge strane, konjunkcija

$$a = b \wedge b = c \wedge f(a,b) \neq f(b,c) \wedge f(b,c) \neq f(g(a),b)$$

je nezadovoljiva jer termovi $f(a,b)$ *i* $f(b,c)$ *pripadaju istoj klasi u kongruentnom zatvorenju.*

Više o jednakosnom rezonovanju i kongruentnom zatvorenju videti u dodatku C.

3.2.2 Struktura *union–find*

Konstruisanje grafa kongruentnog zatvorenja i određivanje particija klasa ekvivalencije obično se realizuje na bazi primitivne *union–find* strukture. Struktura *union–find* održava se korišćenjem dve funkcije:

find funkcija *find* preslikava elemente iz jedne klase ekvivalencije u jedinstvenog predstavnika te klase; ako je $find(q) = q$, onda kažemo da je q koreni čvor svoje klase.

union funkcija $union(q, r)$ ima za argumente dva korena čvora i spaja njihove klase postavljanjem $find(r) := q$ i slično za sve ostale elemente koji su u istoj klasi kao i r.

O različitim implementacijama *union–find* strukturama videti [115].

3.2.3 σ-teorije

Teorija $\mathcal{T}$ je σ-teorija (ili *kanonizabilna teorija*) ako postoji funkcija σ koji preslikava skup termova u skup termova takva da važi:

1. jednakost $t = u$ je valjana u teoriji $\mathcal{T}$ ako i samo ako je $\sigma(t) = \sigma(u)$.

2. ako je t term koji ne pripada teoriji $\mathcal{T}$, onda je $\sigma(t) = t$

3. $\sigma(\sigma(t)) = \sigma(t)$

4. ako je $\sigma(t) = f(t_1, \ldots, t_n)$ (f je bilo koji funkcijski simbol) za neki term t teorije $\mathcal{T}$, onda je $\sigma(t_i) = t_i$ za $1 \leq i \leq n$.

5. $\sigma(t)$ ne sadrži promenljive koje ne sadrži term t.

Na primer, svaki term realne linearne aritmetike može biti sveden na kanonski oblik $a_1x_1 + a_2x_2 + \ldots a_nx_n + c$ ($n \geq 0$) pri čemu je uveden neki poredak između simbola x_i ($0 \leq i \leq n$).

3.2.4 Algebarski rešive σ-teorije

σ-teorija $\mathcal{T}$ je *algebarski rešiva* ako postoji izračunljiva funkcija *solve* koja za argument jednakost e vraća $\top$, $\bot$ ili konjunkciju jednakosti i ima sledeća svojstva (neka je $solve(e) = E$):

1. E i e su ekvivalentni, tj. svaki model koji zadovoljava E zadovoljva e i svaki model za e može se proširiti do modela za E (jer E može da sadrži nove promenljive koje se ne pojavljuju u e);

2. $E \in \{\top, \bot\}$ ili $E = \bigwedge_i (x_i = t_i)$

3. ako e ne sadrži promenljive, onda $E \in \{\top, \bot\}$

4. ako je $E = \bigwedge_i (x_i = t_i)$ onda važi:

   (a) x_i se pojavljuje u e

   (b) za sve i i j, x_i se ne pojavljuje u t_j

   (c) za sve i i j, $i \neq j$ povlači $x_i \neq x_j$

   (d) $\sigma(t_i) = t_i$

Primetimo da funkcija *solve* može da otkrije nezadovoljivost jednakosti i, time, nezadovoljivost konjunkcije koja se dokazuje. Štaviše, kad god je jednakost ekvivalentna sa $\bot$, *solve* to može da otkrije.

Algebarski rešive σ-teorije su jednakosna realna linearna aritmetika, jednakosna celobrojna linearna aritmetika, konveksna teorija lista, monadička teorija skupova itd. Na primer, funkcija *solve* za realnu linearnu aritmetiku za jednakost oblika $a_1x_1 + \ldots a_nx_n + c =_{\mathbf{R}} b_1x_1 + \ldots b_nx_n + d$ vraća $x_1 =_{\mathbf{R}} ((b_2 - a_2)/(a_1 - b_1))x_2 + \ldots ((b_n - a_n)/(a_1 - b_1))x_n + (d - c)$; funkcija *solve* za celobrojnu linearnu aritmetiku za $17x - 49y = 30$ vraća $x = 49z - 4 \ \wedge \ y = 17z - 2$; funkcija *solve* za teoriju lista za $cons(car(x)), cdr(car(y))) = cdr(cons(y, x))$ vraća $car(x) = car(x) \wedge y = cons(cons(a, cdr(x)), d)$.

3.2.5 Algoritam

U Shostakovom algoritmu, koriste se tri osnovne strukture podataka *use*, $find$ i *sig*. Struktura *use* ne mora da inicijalno oslikava sve termove datih jednakosti, već nove termove i njihove neposredne veze ažurira tokom uvođenja novih termova; struktura $find$ je inicijalizovana tako da je $find(t) = t$ za sve termove, a struktura *sig* se tokom algoritma ažurira tako da uvek važi $sig(f(u_1, \ldots, u_n)) = f(find(u_1), \ldots, find(u_n))$. Algoritam primenjuje *canon* na obe strane jednakosti koja se obrađuje, kako bi bile dobijene njihove normalne forme zasnovane na do tada poznatim jednakostima. U slučaju terma t koji je već bio obrađen, *canonsig* osigurava da je njegova normalna forma upravo $find(t)$.

Shostakov algoritam za kombinovanje procedura odlučivanja je, suštinski, uopštenje Shostakovog algoritma za kongruentno zatvorenje, tj. uopštenje neinterpretiranog slučaja na interpretirani slučaj. Funkcija σ ima u interpretiranom slučaju ulogu semantičkog analogona strukture $find$ u neinterpretiranom slučaju. Dakle, u čistom neinterpretiranom slučaju, $find$ vraća kanonskog predstavnika terma na bazi obrađenih jednakosti. U čisto interpretiranom slučaju, funkcija σ vraća kanonskog predstavnika terma u terminima interpretirane teorije. Shostakov algoritam je, suštinski, metod za kombinovanje te dve kanonske forme.

Shostakov algoritam u svom osnovnom delu, na bazi datih jednakosti, ažurira sve strukture podataka i time definiše funkciju *canon*. Ukoliko pri tome nije detektovana kontradikcija, pristupa se obradi nejednakosti i ispituju se vrednosti *canon* funkcije za levu i desnu stranu nejednakosti. Ukoliko su one jednake, nejednakost (pa i polazna konjunkcija literala) je nezadovoljiva. Ukoliko se ne detektuje kontradikcija, zadata konjunkcija literala je zadovoljiva.

Pseudokôd Shostakovog algoritma (odnosno osnovnog njegovog dela) dat je u nastavku.

```
Sh(T)=
  CASES T OF
    nil : RETURN,
    [a=b, T'] :
      Sh(T');
      Process1(solve(canon(a)=canon(b)))
  ENDCASES

Process1(S)=
  FOR e in S DO
    CASES e OF
      false : RETURN unsatisfiable,
      a=b : Merge(canonsig(a), canonsig(b))
    ENDCASES

Merge(a,b)=
```

```
  UNLESS a=b DO
    union(a,b);
    FOR u IN use(a) DO
      IF u is uninterpreted
        THEN
          replace a with b in the argument list of sig(u);
          FOR v IN use(b) WHEN sig(v)=sig(u) DO
            Process1(solve(find(u)=find(v)));
          use(b) := use(b) ∪ { u }
        ELSIF find(u)=u
          THEN
            u' := u with b for a in its argument list;
            Merge(u, canonsig(σ(u')));
      ENDIF


canon(t)=canonsig(signature(t))

canonsig(w)=
  IF w is atomic
    THEN
      RETURN find(w);
    ELSE
      LET f(w1, w2, ..., wn) = w IN
        IF w is interpreted, replace each wi with canonsig(wi);
        IF w=sig(u) for some u ∈ use(w1)
          THEN
            RETURN find(u)
          ELSE
            FOR i FROM 1 TO n DO add w to use(wi);
            sig(w) := w;
            use(w) := { };
            RETURN w
        ENDIF
  ENDIF

signature(t)=
  IF t is a constant
    THEN
      RETURN t
    ELSE
      RETURN σ(f(canon(t1),...,canon(tn)))
        where t=f(t1,t2,...,tn)
  ENDIF
```

Primer 3.1 *Slika 3.1 ilustruje različite faze u primeni procedure na formulu*

$$z = f(x - y) \ \wedge \ x = z + y \ \wedge \ -y \neq -(x - f(f(z)))$$

koja pripada uniji realni linearne aritmetike i čiste teorije jednakosti. Puna linija na dijagramima predstavlja union − find strukturu (npr. $find(z) = f(x + -1 \cdot y)$ na slici 3.1(a). Isprekidana linija predstavlja use strukturu: isprekidana strelica od t do u označava da je u član skupa $use(t)$.

Procedura najpre poziva `Process1` *za jednakosti iz konjunkcije. Izabira se jednakost $z = f(x - y)$, kanonizuje (dajući $z = f(x + -1 \cdot y)$) i predaje funkciji* `solve`*. Kanonizovana jednakost predstavlja svoje rešenje, pa se poziva* `Merge` *za $(z, f(x + -1 \cdot y))$. Slika 3.1(a) ilustruje stanje struktura nakon ovog poziva. Veze označene isprekidanim linijama proističu iz poziva funkcije* `canon` *za $f(x - y)$. Sledeća jednakost koja se obrađuje je $x = z + y$ koja ima kanonizovan oblik $x = f(x + -1 \cdot y) + y$. Pozvana funkcija* `solve` *nema efekta i poziva se funkcija* `Merge` *za $(x, f(x + -1 \cdot y) + y)$. Unutar ove funkcije, poziva se i funkcija* `canonsig` *za $x + -1 \cdot y$ dajući $f(x + -1 \cdot y)$. Nakon toga, rešava se dodatna jednakost $x + -1 \cdot y = f(x + -1 \cdot y)$ (slika 3.1(b) ilustruje stanje struktura nakon ovog poziva) i zatim, poziva funkcija* `Merge` *za $(x + -1 \cdot y, f(x + -1 \cdot y))$. (slika 3.1(c)). Primetimo da je term $f(x + -1 \cdot y)$ dodat u svoju sopstvenu* `use` *listu i da je vrednost* `sig`*$(f(x + -1 \cdot y)$ ažurirana tako da oslikava novu vrednost za* `signature`*$(f(x + -1 \cdot y))$ koja je jednaka $f(f(x + -1 \cdot y))$. Nema novih parova jednakosti, pa je obrada jednakosti završena.*

Funkcija `canon` *i za $-y$ i za $-(x - f(f(z)))$ vraća $-1 \cdot y$, pa je zadata konjunkcija nezadovoljiva.*

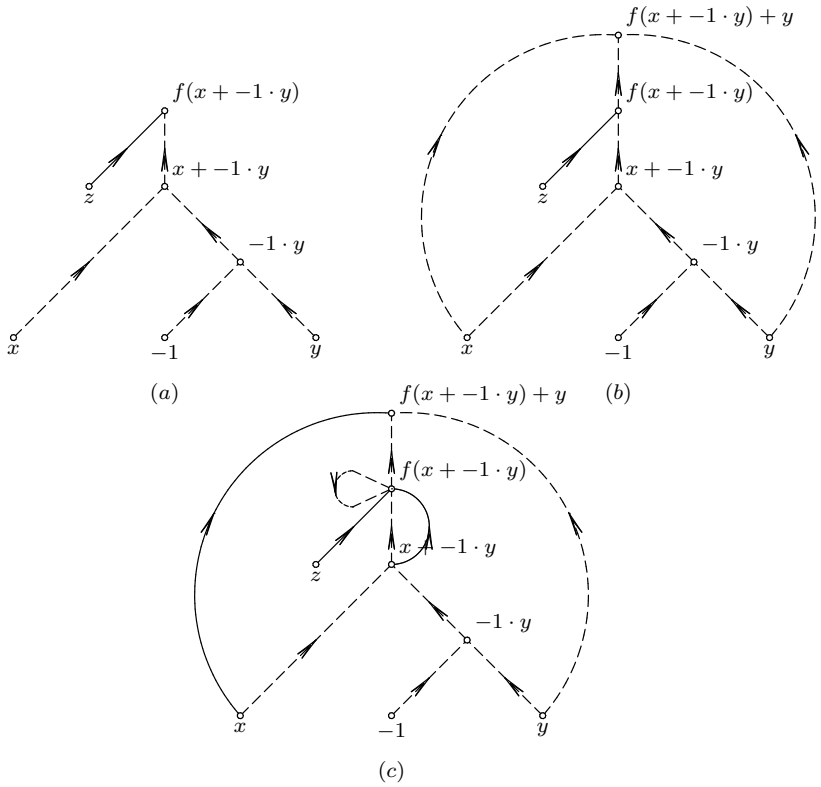
3.2.6 Saglasnost i kompletnost

Neka $canon^T(t)$ označava vrednost funkcije *canon* za term t nakon izvršenja programa $Sh(T)$. Neka je $canon*^T(t)$ isto što i $canon^T(t)$ ali bez bočnih efekata izmene *sig* i *use* struktura.

Saglasnost Shostakovog algoritma potdvrđuje naredne teorema [32]:

Teorema 3.7 *Ako je $canon*^T(a) = canon*^T(b)$ onda je $T \models a = b$.*

Nekoliko radova o Shostakovom algoritmu sadrži opise njegovih varijanti, dok neki sadrže i dokaze da se algoritam zaustavlja i da je potpun [32, 9, 11]. Međutim, Rueß i Shankar nedavno su pokazali [100] da su Shostakov algoritam kao i sve njegove varijante nepotpuni i, štaviše, da se ne zaustavljaju uvek. Naime, konjunkcija $f(v - 1) - 1 = v + 1 \wedge f(u) + 1 = u - 1 \wedge u + 1 = v$ je nezadovoljiva, ali to Shostakov algoritam ne može da detektuje. Dodatno, Shostakov algoritam se ne zaustavlja za sledeću jednostavnu formulu: $f(v) = v \wedge f(u) = u - 1 \wedge u = v$. Nedavni radovi Rueß i Shankara [100] i Barretta, Dilla i Stumpa [10] su usmereni ka prevazilaženju ovih nedostataka Shostakovog algoritma.



Slika 3.1: Ilustracija primene Shostakovog algoritma

3.2.7 Primene

Shostakov algoritam je zbog svoje efikasnosti naišao na široku popularnost. Interesovanje za ovaj algoritam dodatno je pojačano nakon objavljivanja njegove stroge analize [32]. Shostakov metod koristi se u sistemima od kojih su neki PVS [92], EHDM [43] i STeP [76, 11].

U poređenju sa Nelson/Oppenovim algoritmom, Shostakov pristup daje značajno bolje rezultate i eksperimenti govore da je vreme utrošeno od strane Shostakove procedure za red veličine manje od vremena utrošenog od strane Nelson/Oppenove [32]. S druge strane, domen primene Shostakov algoritma je znatno manji nego za Nelson/Oppenov.

U uspešnim dokazivačima teorema koriste se i razne varijante osnovnog Shostakovog algoritma [9, 11].

3.3 Boyer/Mooreova procedura za linearnu aritmetiku

Boyer/Moorova procedura za linearnu aritmetiku, za razliku od Nelson/Oppenove i Shostakove procedure, ne bavi se problemom kombinovanja više procedura odlučivanja, već problemom proširivanja domena jedne procedure odlučivanja dodatnim lemama. Procedura je uspešno implementirana u okviru Boyer/Mooreovog dokazivača NQTHM, a opis procedure objavljen je prvi put u radu [17].

Osnovna ideja u ovoj proceduri je korišćenje nove heuristike — *augmentacije* (eng. augmentation) koja proširuje skup informacija dostupnih proceduri odlučivanja tvrđenjima o funkcijama o kojima sâma procedura odlučivanja ne zna ništa. Na osnovu Boyer/Moorevih analiza značaj ove heuristike je suštinski za uspešnu primenu: sâma procedura odlučivanja nije bitno uticala na moć dokazivača, a zajedno sa heuristikom unapređenje je bilo značajno (kako u većoj efikasnosti, većem domenu, tako i u manjoj potrebi za akcijama korisnika).

Problem sa heuristikom augmentacije je to što ona u velikoj meri komplikuje integrisanje procedure odlučivanja u dokazivač i zahteva sofisticirane kontrolne mehanizme. Rezultujuća integracione shema je izuzetno komplikovana, a sam opis dug i teško razumljiv (rad [17] slovi za jedan od radova najtežih za razumevanje u čitavoj oblasti). U sâmom radu detalji implementacije su pomešani sa opisom osnovnih algoritama i delovi algoritama opisani su u terminima korišćenih struktura podataka a ne u terminima njihovog logičkog značenja. Sve to uticalo je na obeshrabrivanje od daljih pokušaja mnoge koji su želeli da naprave drugu praktičnu implementaciju Boyer/Mooreove procedure. (Uprkos tome, rad [17] izvršio je snažan uticaj u oblasti dokazivanja teorema i to ne samo u podoblasti korišćenja procedura odlučivanja, već i u prezapisivanju, jednakosnom rezonovanju itd.)

U nastavku, pojednostavljenom opisu procedure prethodi kratak opis Boyer/-Mooreovog dokazivača NQTHM.[2]

3.3.1 Boyer i Mooreov dokazivač NQTHM

Boyer i Mooreov NQTHM [15, 16] je dokazivač teorema koji koristi strateške informacije u izvođenju induktivnih dokaza. Namenjen je verifikaciji programa, pa je procedura odlučivanja za teoriju PIA (tj. za linearnu aritmetiku, kako teoriju PIA zovu Boyer i Moore) dodata kako bi povećala efikasnost sistema. Ta procedura komunicira sa dva osnovna modula dokazivača — *pojednostavljivačem* (eng. simplifier) i *prezapisivačem* (eng. rewriter).

Dokazivač NQTHM radi u jeziku prvog reda bez kvantifikatora. Tvrđenja oblika $\phi_1 \wedge \phi_2 \wedge \phi_n \rightarrow \psi$ (gde su ϕ_i i ψ atomičke formule) dokazuju se svođenjem formule $\phi_1 \wedge \phi_2 \wedge \phi_n \wedge \neg\psi$ na kontradikciju.

Višesortna logika je implicitno reprezentovana na osnovu tzv. principa školjke (eng. shell principle), koji za svaku novu sortu zahteva:

[2] Opis je zasnovan na [17, 23].

- konstruktor funkciju
- destruktor funkcije
- individualnu bazičnu konstantu
- dobro-zasnovano uređenje
- predikat koji prepoznaje termove date sorte

Prirodni brojevi su uvedene principom školjke na način prikazan u tabeli 3.1. Odgovarajuće aksiome navedene su u tabeli 3.2.

Element školjke	simbol	Svojstvo/značenje
konstruktor	s	funkcija sledbenik
desruktor	pr	funkcija prethodnik ($pr(s(v) = v$)
bazična konstanta	0	prirodan broj 0
dobro zasnovana relacija	$prec$	$prec(u, v) \equiv v = s(u)$
prepoznavač sorte	nat	$nat(v)$ ako i samo ako je v prirodan broj

Tabela 3.1: Sintaktički opis prirodnih brojeva u sistemu NQTHM

A_1 $nat(0) \wedge nat(s(v)$
A_2 $s(v) \neq 0$
A_3 $(nat(v) \wedge v \neq 0) \rightarrow s(pr(v)) = v$
A_4 $nat(v) \rightarrow pr(s(v)) = v$
A_5 $pr(0) = 0 \wedge (\neg nat(v) \rightarrow pr(v) = 0$
A_6 $(nat(v) \wedge v \neq 0 \wedge u = pr(v)) \equiv prec(u, v)$
A_7 $[(\neg nat(v) \rightarrow \phi[v]) \wedge \phi[0] \wedge ((nat(v) \wedge v \neq 0 \wedge \phi\{pr(v) \mapsto v\}) \rightarrow \phi[v])] \rightarrow \phi[v]$

Tabela 3.2: Aksiome prirodnih brojeva u sistemu NQTHM

Jezik sistema NQTHM sadrži simbole koje zahteva princip školjke za svaku od definisanih sorti. Teorija sistema NQTHM je određena unijom aksiomatskih sistema za definisane sorte. Za svaku sortu postoji dobro zasnovano uređenje i dokaz indukcijom je prioritet sistema. Data formulu ϕ bez kvantifikatora se, pre pokušaja da bude dokazana indukcijom, prosleđuje pojednostavljivaču i prezapisivaču koji nastoje da je transformišu u logičku konstantu $\perp$ ili $\top$. Kada ta transformacija ne može da se ostvari, induktivni dokaz se primenjuje na rezultujuću formulu (ili rezultujuće formule).

Pojednostavljivanje je prva dokazivačka strategija koja se primenjuje na zadatu formulu. Pojednostavljivač je procedura čiji je ulaz formula (u konjuktivnoj normalnoj formi), a izlaz skup formula koje su u nekom smislu jednostavnije od ulazne formule. Da bi se dokazalo da je zadata formula $\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ teorema, potrebno je dokazati da je teorema svaka od formula ϕ_i $(1 \leq i \leq n)$. Pojednostavljivanje disjunkcije literala odvija se na sledeći način: za svaki literal pretpostavi se da važe negacije preostalih literala i prezapisuje se u tom

kontekstu (radi se, dakle, o jednoj formi kontekstualnog prezapisivanja [124]). Cilj je da se bar jedan literal prezapiše u logičku konstantu $\top$.

Prezapisivač je procedura koja zadati term prezapisuje u skladu sa određenom smenom i zadatim kontekstom. Ova procedura koristi uslovna pravila prezapisivanja oblika $\delta \longrightarrow \delta'$ **if** $h_1 \wedge h_2 \wedge \ldots \wedge h_n$. U kombinaciji sa pojednostavljivačem, prezapisivač radi u prisustvu pretpostavki, odnostno konteksta. Ako formula ϕ treba da bude dokazana na osnovu pretpostavki $\Phi = \{\phi_1, \ldots, \phi_n\}$ i uslovnog pravila prezapisivanja R (oblika $\delta \longrightarrow \delta'$ **if** $h_1 \wedge h_2 \wedge \ldots \wedge h_n$), sistem najpre pokušava da utvrdi da li postoji instanca terma/formule δ koja je sintaktički identična nekom podizrazu formule koja se dokazuje. Ako je to tačno, odgovarajuća supstitucija σ se primenjuje na uslove pravila prezapisivanja koji tada treba da budu dokazani (na osnovu konteksta). Podciljevi $h_i\sigma$ se dokazuju rekurzivno dok ne budu prezapisani u $\top$ (ako je to moguće).

Kontrolni mehanizam dokazivača NQTHM uključuje sofisticirane heuristike za izbor pogodnih pravila prezapisivanja, zaustavljanje neproduktivnog grananja, kontrole ekspanzije izraza zasnovane na definicijama itd. Sistem NQTH dokazao je mnošto netrivijalnih teorema, odnostno verifikacijskih tvrđenja, od kojih su neka ispravnost procedure odlučivanja za iskazni račun, ispravnost algoritma za pretraživanje stringova, egzistencija i jedinstvenost faktorizacije prirodnih brojeva [17].

3.3.2 Linearna aritmetika

Zbog značaja koji aritmetika ima u verifikacijskim tvrđenjima u sistemu NQTHM, postoji potreba za procedurom odlučivanja koja efikasno razrešava aritmetička tvrđenja (ili bar neka od njih). Kako je cela aritmetika neodlučiva, neophodno je izabrati neki njen odlučiv deo — npr. strogo multiplikativnu aritmetiku ili Prezburgerovu aritmetiku. Boyer i Moore su se odlučili za Prezburgerovu celobrojnu aritmetiku, s tim što koriste proceduru odlučivanja za PRA (tj. za teoriju gusto uređenih Abelovih grupa bez krajnjih tačaka). Na osnovu teoreme 2.5, procedura odlučivanja za PRA daje saglasne odgovore za sve univerzalne PIA rečenice. Dakle, ako je za neku univerzalnu PIA rečenicu procedura daje potvdan odgovor, onda je ona zaista teorema teorije PIA, Međutim, ne važi obratno i postoje tvrđenja koja su teoreme nad celim brojevima, ali ne i nad racionalnim, na primer:

$$0 < k \wedge 0 < l \wedge 2 \cdot k + 1 \leq 2 \rightarrow 2 \cdot k \leq 2 \cdot l$$

Navedeno tvrđenje dokazivač NQTHM dokazuje indukcijom, a njegovi autori kažu da su takva tvrđenja retka u verifikacijskim problemima (pozivajući se na [86]).

Kao procedura odlučivanja za PRA koristi se Hodesov algoritam [52] (videti poglavlje D.1), tj. Hodesov metod eliminacije varijabli (to je, suštinski, Fourierov metod). Boyer i Moore naglašavaju da je sama ta procedura veoma mali deo ukupne integracione sheme i kažu: "konačna verzija naše procedure za linearnu aritmetiku je implementacija tog algoritma [Hodesovog] u istoj meri u kojoj je softver za Spejs šatl implementacija Njutnovih zakona" [17]. S druge strane, Boyer/Moorova integracion shema je strogo prilagođena Hodesovoj proceduri i

ona ne može biti lako zamenjena nekom drugom.

Formula koja se opovrgava reprezentovana je kao baza polinoma. Svaka nejednakost čuva se u normalizovanom, *linearizovanom* obliku $a_0 + a_1x_1 + \ldots a_nx_n$ (transformacija u normalizovani oblik realizuje se iscrpnom primenom potrebnih pravila prezapisivanja (videti i [22, 23] kao i poglavlje D.5). U jednom koraku primene algoritma svaki par oblika $\alpha + \beta \leq 0$, $\gamma - \beta \leq 0$, zamenjuje se formulom $\alpha + \gamma \leq 0$. U ovom opisu nećemo se upuštati u detalje implementacije ovih osnovnih funkcija.

3.3.3 Prosta integraciona strategija

U prvoj instanci, u dokazivač NQTHM je bila ugrađena proceduru za linearnu aritmetiku kao "crna kutija", dakle kao procedura koja sa ostatkom dokazivača može da komunicira samo preko jednog ulaza (formula linearne aritmetike) i preko jednog izlaza (potvrdnog ili odrečnog odgovora). Očekivalo se da će ovako ugrađena procedura moći da da ubrza rezonovanje o aritmetičkim vezama, o tranzitivnosti nejednakosti (umesto korišćenja pravila prezapisivanja itd). Ipak, pokazalo se da ova, prosta integraciona strategija dokazivaču donosi samo zanemarljivo unapređenje. Naime, praktično jedina tvrđenja koja su dokazana efikasnije nego bez procedure za linearnu aritmetiku bila su tvrđenja sâme linearne aritmetike. Većina tvrđenja iz verifikacijskih problema uključivala je funkcije koji ne pripadaju linearnoj aritmetici, te je, u tim slučajevima, prosta integraciona strategija bila neupotrebljiva. Dakle, bilo je potrebno proširiti je modulima koji se odnose na "strane termove" — na funkcije koji ne pripadaju linearnoj aritmetici.

3.3.4 Augmentacija

Heuristika augmentacije eliminiše "najteži strani term" formule koja se opovrgava, na osnovu dodatnog raspoloživog tvrđenja, pravila odnosno leme. Relacija poretka "težine" termova u Boyer/Mooreovoj proceduri je totalno uređenje nad termovima i definiše se na sledeći način: term t_1 je *teži* od t_2 ako i samo ako je ispunjen jedan od sledećih uslova:

- broj varijabli u t_1 veći je nego broj varijabli u t_2;
- broj varijabli u t_1 i t_2 jednak, ali je "veličina" terma t_1 veća nego "veličina" terma t_2;
- broj varijabli u t_1 i t_2 jednak i "veličine" t_1 i t_2 su jednake, ali term t_1 dolazi iza t_1 u alfabetskom poretku termova.

Pod *veličinom* se smatra broj otvorenih zagrada u neskraćenom zapisu terma.

Ilustrujmo osnovnu ideju pravila augmentacije primerom. Pretpostavimo da treba dokazati tvrđenje

$$l \leq min(a) \wedge 0 < k \rightarrow l < max(a) + k$$

Navedeno tvrđenje nije PRA, te procedura odlučivanja za PRA ne može da bude primenjena. Pretpostavimo i da je raspoloživa lema

$$min(x) \leq max(x)$$

Najteži strani term u formuli $l \leq min(a) \wedge 0 < k \rightarrow l < max(a) + k$ je $min(a)$, a u formuli $min(x) \leq max(x)$ term $min(x)$. Term $min(x)$ instanciran supstitucijom $\{x \mapsto a\}$ postaje jednak termu $min(a)$. Kada mu se doda lema instancirana supstitucijom $\{x \mapsto a\}$, polazno tvrđenje postaje

$$l \leq min(a) \wedge 0 < k \wedge min(a) \leq max(a) \rightarrow l < max(a) + k$$

Navedeno tvrđenje ne pripada teoriji PRA, ali ako se strani termovi $min(a)$ i $max(a)$ abstrahuju novim varijablama $mina$ i $maxa$, tvrđenje postaje

$$l \leq mina \wedge 0 < k \wedge mina \leq maxa \rightarrow l < maxa + k \ ,$$

ono pripada teoriji PRA i jeste njegova teorema. (U proceduri za linearnu aritmetiku apstrakcija se ne primenjuje eksplicitno, već se dozvoljava obrada stranih termova kao da su promenljive.)

Heuristika augmentacije je kruto integrisana sa Hodesovim metodom eliminacije varijabli, pa zbog toga nije moguće praviti jedonstavne izmene jednog ili drugog dela integracione sheme.

3.3.5 Moduli procedure

Boyer/Moorova procedura za linearnu aritmetiku u velikoj meri kooperira sa modulima za pojednostavljivanje i prezapisivanje, a sama je sačinjena od sledećih specifičnih modula:

Modul za linearizaciju Ako formula ϕ pripada linearnoj aritmetici, onda

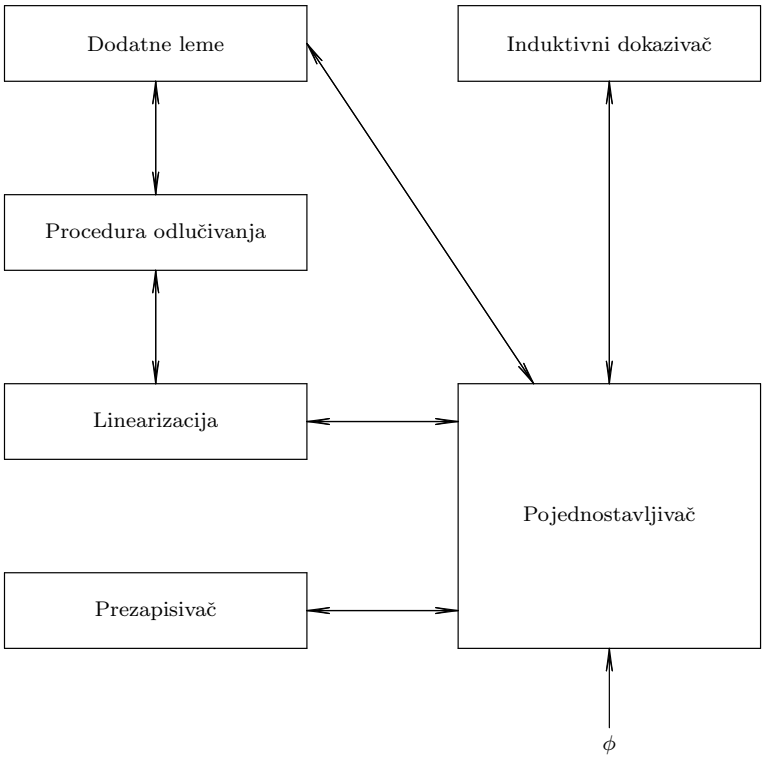
- termovi se normalizuju
- dodaju se uslovi nenegativnosti za prirodne brojeve
- tako proširena formula se negira i njene atomičke formule se linearizuju
- rezultujuća formula transformiše se u disjunktivnu normalnu formu i zatim prosleđuje proceduri odlučivanja za PRA.

Procedura odlučivanja Ako je $\psi_1 \wedge \ldots \wedge \psi_m$ linearizovana formula linearne aritmetike i ako se svodi na $\bot$ procedurom odlučivanja za PRA, proces se zaustavlja. Inače, ako se svodi na $\top$ i ako bar jedan od literala sadrži neki strani term, onda se $\psi_1 \wedge \ldots \wedge \psi_m$ prosleđuje mehanizmu za korišćenje dodatnih hipoteza/lema.

Mehanizam za korišćenje dodatnih hipoteza/lema Ako je $\psi_1 \wedge \ldots \wedge \psi_m$ formula čiji je najteži term t traži se raspoloživa lema ψ koja uključuje term koji može biti instanciran u t. Ako takva lema nije pronađena, zadata formula se vraća nepromenjena. Inače:

- ako je ψ literal ili konjunkcija literala, onda se ona spaja sa $\psi_1 \wedge \ldots \wedge \psi_m$ i tako proširena formula se prosleđuje proceduri odlučivanja
- ako je ψ oblika $\phi_1 \wedge \ldots \phi_n \rightarrow \phi$, formuli $\psi_1 \wedge \ldots \wedge \psi_m$ se dodaje literal ϕ i tako proširena formula se prosleđuje proceduri odlučivanja. Ako se detektuje nezadovoljivost, uslovi ϕ_i se pojedinačno prosleđuju pojednostavljivaču. Ako su svi uslovi ϕ_i transformisani u $\top$, onda se $\psi_1 \wedge \ldots \wedge \psi_m$ prezapisuje u $\bot$.

Komunikacija različitih modula u sistemu NQTHM ilustrovana je na slici 3.2.



Slika 3.2: Komunikacija modula procedure za linearnu aritmetiku sa osnovnim modulima sistema NQTHM

Primer 3.3 *Neka je zadato tvrđenje ϕ:*

$$l \leq min(x) \wedge 0 < k \rightarrow l < max(x) + k$$

Formula $\neg\phi$ u normalizovanom obliku jednaka je (zbog jednostavnosti, izostavljena su ograničenja nenegativnosti za promenljive):

$$l - min(x) \leq 0 \wedge 1 - k \leq 0 \wedge max(x) + k - l \leq 0$$

Navedena formula prosleđuje se proceduri odlučivanja za PRA *i, kako je ona zadovoljiva, procedura pokušava da nađe upotrebljivu lemu. Najteži strani term u formuli* $l-min(x) \leq 0 \wedge 1-k \leq 0 \wedge max(x)+k-l \leq 0$ *je* $min(x)$*. Pretpostavimo da je raspoloživa lema*

$$min(v) \leq max(v)$$

koja može biti instancirana i linearizovana tako da daje

$$min(x) - max(x) \leq 0$$

Dobijena formula se dodaje tekućoj formuli kao konjunkt:

$$min(x) - max(x) \leq 0 \wedge l - min(x) \leq 0 \wedge 1 - k \leq 0 \wedge max(x) + k - l \leq 0$$

Proširena formula se prosleđuje proceduri odlučivanja za PRA *i eliminiše se najpre* $min(x)$*:*

$$l - max(x) \leq 0 \wedge 1 - k \leq 0 \wedge max(x) + k - l \leq 0$$

a zatim i $min(x)$ *dajući*

$$l + k - l \leq 0 \wedge 1 - k \leq 0$$

Eliminisanjem promenljive k *dobija se formula* $1 \leq 0$ *koja nije zadovoljiva u* PRA*. Dakle, formula* ϕ *je teorema u* PRA*.*

3.3.6 Primene

Boyer/Mooreova procedura za linearnu aritmetiku uspešno je primenjena u njihovom dokazivaču NQTHM (opis implementacije i eksperimentalni rezultati dati su u radu [17]). Međutim, zbog komplikovanog opisa i detalja implementacije pomešanih sa idejama visokog nivoa, procedura u svom osnovnom obliku nije naišla na popularnost — u svom osnovnom obliku ona je implementirana jedino u dokazivaču NQTHM. S druge strane, ova procedura i rad u kojem je prvi put opisana unela je dosta značajnih novina u oblast i smatra se jednim od najznačajnijih referenci. Između ostalog, na ovoj proceduri zasniva se rad sa brojevima u sistemu TECTON, kao i najveći deo ograničenog kontekstualnog prezapisivanja.

3.4 Rezonovanje o brojevima u sistemu TECTON

U sistem TECTON ugrađena je Kapur/Nieova procedura za rezonovanje o brojevima koja je slična Boyer/Mooorovoj proceduri. Osnovni principi na kojima je zasnovana ta procedura objavljeni su u radovima [68] i [69]. Pored kontektualnog prezapisivanja i augmentacije koji se zajedno sa procedurom odlučivanja za Prezburgerovu aritmetiku koriste u sistemu NQTHM, dokazivač TECTON raspolaže i moćnim mehanizmom za rad sa jednakostima i implicitnim jednakostima. Sistem TECTON je izgrađen nad prezapisivačkim dokazivačem RRL i sve jednakosti (kako eksplicitne, tako i implicitne) su od ključnog značaja za njegovu efikasnost (jer se na bazi njih generišu dodatna pravila prezapisivanja).

3.4.1 Dokazivač RRL i sistem TECTON

Dokazivač RRL zasnovan je na prezapisivanju i prezapisivačkim tehnikama, ali je ojačan i brojnim dodatnim heuristikama [66, 69]. RRL podržava jednakosno i induktivno rezonovanje korišćenjem prezapisivačkih tehnika. Specifikacijski jezik dokazivača RRL je jednakosni, sa podrškom definisanju apstraktnih tipova podataka korišćenjem konstruktora. RRL transformiše ulaznu formulu u jednakosti i uslovne jednakosti koje su implicitno univerzalno kvantifikovane. RRL se razlikuje u svom dizajnu od mnogih sistema kao što su *PVS*, *HOL*, *Isabelle*, *Nuprl*, po tome što on pokušava da do svih potrebnih izvođenja dođe bez navođenja korisnika (i u tom smislu je sličan sistemima *Otter* i *EQP*). RRL ima ugrađene sledeće heuristike:

1. orijentisanje jednakosti u zaustavljajuća pravila prezapisivanja;
2. određivanje sledećeg pravila prezapisivanja koje će da bude primenjeno, i u vezi s tim, pogodno instanciranje slobodnih promenljivih i potvrđivanje uslova pravila prezapisivanja (ukoliko ih ima);
3. pozivanje procedura odlučivanja za brojeve (Prezburgerova aritmetika bez kvantifikatora), bitove, tipove podataka sa slobodnih konstruktorima i iskaznu logiku;
4. izbor sledećeg pravila izvođenja;
5. automatsko generisanje analize slučajeva;
6. izbor između više indukcijskih shema zasnovan na definicijama funkcijskih simbola koji se pojavljuju u tvrđenju;
7. generisanje lema;
8. automatski bektreking u pokušaju da se nađe dokaz, ako je jedan pokušaj propao.

Slično kao u sistemu NQTHM, sve heuristike se primenjuju u istom redu korišćenjem jedne strategije. Korisnik ne treba da odlučuje o tome kojim redom pravila izvođenja treba da se primenjuju, kako pravila prezapisivanja treba da budu instancirana i slično.

Svako pravilo prezapisivanja koje koristi RRL mora biti zaustavljajuće, što se obezbeđuje algoritmom koji implementira dobro-zasnovano uređenje svođenja i to, preciznije, uređenje leksikografske staze (videti poglavlje B.3.4).

Pojednostavljivanje u odnosu na kontekst (kontekstualno prezapisivanje) je osnovni mehanizam izvođenja. Algoritam za pojednostavljivanje automatski određuje koje je pravilo prezapisivanja primenljivo na dato tvrđenje. Ovo se postiže najpre određivanjem mogućih instancijacija za promenljive u pravilu prezapisivanja i, nakon toga, proveravanjem da su uslovi pravila (ukoliko ih ima) zadovoljeni. Uslovi pravila se proveravaju uzimajući u obzir kontekst formule koja se dokazuje i ostala pravila (kao i procedure odlučivanja) koja su čvrsto integrisana u sistemu.

Ako neko tvrđenje ne može biti dokazano pojednostavljivanjem, primenjuje se indukcija. Promenljive na koje se primenjuje indukcija se automatski biraju korišćenjem tzv. *cover set* metoda.

Ako pokušaj dokaza indukcijom ne uspe a i ne dovede do kontraprimera, primenjuje se bektreking koji vodi do sledeće primenljive indukcijske sheme.

RRL ima više heuristika za generisanje lema na osnovu formula generisanih tokom pokušaja dokaza. Neke od njih su generalizacija (apstrahovanjem nekih podizraza koji se pojavljuju u tvrđenju) i slabljenje uslova (u uslovnim tvrđenjima). Potrebne leme koje ne mogu biti automatski generisane moraju da budu stavljene na raspolaganje od strane korisnika (to je mesto na kojem RRL zahteva asistenciju).

Kada pokušaj dokaza ne uspe, transkript pokušaja može da sugeriše razlog neuspeha, kao i alternativne pokušaje dokaza (npr. modifikovanjem tvrđenja, dodavanjem leme itd).

TECTON je verifikacijski i specifikacijski sistem izgrađen nad dokazivačem *Rewrite Rule Laboratory* (RRL). Njegova osnovna namena je podrška konstruisanju velikih i kompleksnih dokaza kakvi su obično potrebni u rezonovanju o specifikaciji i opisu generičkih komponenti koje se koriste u dizajnu softvera i hardvera.

3.4.2 Fourierov algoritam

Fourierov algoritam za linearne nejednakosti nad racionalnim brojevima[3] [74] je suštinski jednak Hodesovom algoritmu [52] i zasniva se na sledećoj ideji: neka je $\mathcal{P}$ skup nejednakosti i neka je x promenljiva koja se pojavljuje u njemu; neka je C_i nejednakost iz $\mathcal{P}$ sa negativnim koeficijentom $-c_i$ uz x ($c_i > 0$) i neka je C_j nejednakost iz $\mathcal{P}$ sa pozitivnim koeficijentom c_j uz x ($c_j > 0$); tada je $c_iC_j + c_jC_i$ nejednakost koja ne sadrži x i svako rešenje za zajedničko za C_i i C_j je istovremeno rešenje i za $c_iC_j + c_jC_i$; dodatno, svako rešenje za $c_iC_j + c_jC_i$ može da se proširi do rešenja za C_i i C_j. Egzistencijalno kvantifikovana promenljiva može da se eliminiše iz konjunkcije nejednakosti transformacijom svih parova nejednakosti na opisani način. Time je određena i procedura odlučivanja za racionalne nejednakosti zasnovana na eliminaciji kvantora.

Opisani postupak je saglasan i potpun za teoriju PRA, ali nije potpun za PIA i PNA. Za cele i prirodne brojeve nejednakost C_i sa negativnim koeficijentom $-c_i$ uz x ($c_i > 0$) i nejednakost C_j sa pozitivnim koeficijentom c_j uz x ($c_j > 0$) moguće je zameniti nejednakošću $c_iC_j + c_jC_i$ ili nejednakošću $(nzd(c_i, c_j)/c_j)C_j + (nzd(c_i, c_j)/c_i)C_i$. Svako rešenje polaznih nejednakosti je istovremeno i rešenje nove nejednakosti, ali ne važi obratno. Na primer, nejednakosti $4x \leq 7$ i $-6x \leq -8$ se transfomišu u nejednakost $3 \cdot 4x - 2 \cdot 6x \leq 3 \cdot 7 - 2 \cdot 8$ što je ekvivalentno sa $0 \leq 5$; nejednakost $0 \leq 5$ je zadovoljiva, ali ne postoji vrednost x koja zadovoljava polazne nejednakosti, odnosno nejednakosti $12x \leq 21$ i $-12x \leq -16$ (jer ne postoji vrednost x za koju je zadovoljeno

[3] U literaturi (pa i u radu [68]) na Fourier/Motzkinov algoritam za linearne nejednakosti često se refereše kraće kao na Fourierov algoritam. U daljem tekstu, pod terminom Fourierov algoritam i mi ćemo podrazumevati Fourier/Motzkinov algoritam za linearne nejednakosti.

$16 \leq 12x \leq 21$). Navedeni primer ilustruje kako Fourierov metod nije potpun za dokazivanje nezadovoljivosti konjuncija PNA literala ili PIA literala. Za Kapur/Nieovo proširenje Fourierovog metoda koje obezbeđuje potpunost potrebno je (u svakoj iteraciji) sledeće: ako je C_i oblika $A \leq c_i x$, a C_j oblika $c_j x \leq B$, onda pored nejednakosti $(nzd(c_i, c_j)/c_j)C_j + (nzd(c_i, c_j)/c_i)C_i$ treba uvesti i ograničenja $(nzd(c_i, c_j)/c_j)B < (nzd(c_i, c_j)/c_i)A + nzd(c_i, c_j) - d_i - d_j$. Tek kada su eliminisane sve promenljive, ako je dobijena tačna nejednakost, proverava se da li su zadovoljena sva ograničenja. Opisani postupak određuje proceduru odlučivanja za PNA i PIA, ali ona više ne spada u grupu procedura sa eliminacijom kvantora.

Vredno je naglasiti da, kao što autori sistema kažu, nije neophodno imati tri nezavisne implementacije Fourierovog algoritma (po jednu za racionalne, cele i prirodne brojeve). Dovoljno je imati osnovni Fourierov algoritam, pri čemu se za cele i prirodne brojeve, nakon eliminacije varijabli, vrši provera dodatnih uslova koji su generisani tokom primene procedure.

Detaljni opis Fourierovog metoda i njegovih proširenja i svojstava dat je u [68].

3.4.3 Implicitne jednakosti

U dokazivaču RRL, kao u svakom sistemu zasnovanom na prezapisivanju, u formuli koja se dokazuje poseban značaj imaju jednakosti, kako eksplicitne, tako i implicitne. Jednakosti se koriste za uvođenje novih pravila prezapisivanja i za pojednostavljivanje formule koja se dokazuje.

Ako je C nejednakost $\sum_{j=1}^{n} a_i x_i \leq b$, sa $C^{=}$ označavamo jednakost $\sum_{j=1}^{n} a_i x_i = b$. Implicitna jednakost u skupu jednakosti i nejednakosti $\mathcal{P}$ je jednakost $C^{=}$, gde nejednakost C pripada skupu $\mathcal{P}$ i važi $\mathcal{P} \models C^{=}$ u zadatom domenu. Za racionalne brojeve važe sledeća dva važna tvrđenja:

Teorema 3.8 *Ako za skup nejednakosti C_k $(1 \leq k \leq m)$ važi da je nejednakost $\sum_{i=1}^{m} \alpha_i C_i$ ekvivalentna nejednakosti $0 \leq 0$, onda je svaka od nejednakosti C_k $(1 \leq k \leq m)$ implicitna jednakost.*

Teorema 3.9 *Nejednakost C_i u skupu neprotivrečnih nejednakosti $\mathcal{P}$ je implicitna nejednakost ako i samo ako Fourierov algoritam daje nejednakost $0 \leq 0$ koristeći nejednakost C_i.*

Dakle, u slučaju racionalnih brojeva, moguće je detektovati sve implicitne jednakosti, ali to nije tako u slučaju prirodnih i celih brojeva: naime, u slučaju prirodnih i celih brojeva postoje skupovi nejednakosti za koje nije moguće odrediti sve implicitne jednakosti (na način opisan za racionalne brojeve). Sve implicitne jednakosti moguće je, međutim, odrediti na manje efikasan način, u stilu Nelson/Oppenove procedure (videti 3.1).

3.4.4 Linearna aritmetika sa neinterpretiranim funkcijskim simbolima

Procedura čiji opis sledi odnosi se se Prezburgerovu aritmetiku obogaćenu neinterpretiranim funkcijskim i predikatskim simbolima (tj. obogaćenu čistom teorijom jednakosti). Opisom su obuhvaćene teorije PRA, PIA i PNA, pri čemu se podrazumevano koriste varijante Fourierovog metoda prilagođene ovim teorijama.

Za datu formulu Prezburgerove aritmetike G sa neinterpretiranim funkcijskim simbolima, generiše se formula G_a koja je konjunkcija $GE \wedge G'$ takva da je G' formula čiste Prezburgerove aritmetike (bez neinterpretiranih simbola), a GE je konjunkcija jednakosti koje povezuju strane termove iz G sa novim simbolima (tzv. apstrakcijskim konstantama). GE ne uključuje pojavu nijedne aritmetičke funkcije ili relacije (osim $=$). Formula G_a je nezadovoljiva ako i samo ako je G nezadovoljiva.

Jednakosti iz G' se koriste da eliminišu promenljive iz G' kao i iz GE. Rezultujuće jednakosti iz GE mogu da budu obrađene korišćenjem algoritma za kongruentno zatvorenje (videti C.3) ili Knuth/Bendixovom procedurom upotpunjavanja (videti B.4) za bazne termove. U sistemu RRL koristi se Knuth/Bendixova procedura, koja, u vidu sistema za prezapisivanje, daje proceduru odlučivanja za bazne termove. Dobijena pravila prezapisivanja se onda koriste za normalizovanje formule G'.

Formula G' se dalje obrađuje primenom Fourierovog algoritma. Svaka jednakost koja je izvedena iz G' se dodaje formuli GE i generiše se novi kanonski sistem za prezapisivanje. Ovo može da rezulutuje dodatnim jednakostima koje povezuju strane termove i apstrakcijske konstante.

Opisana interakcija između procedure odlučivanja za bazne termove se uvek zaustavlja. Saglasnost procedure sledi iz saglasnosti baznog upotpunjavanja i iz saglasnosti Fourierovog algoritma. Dakle, ako je opisanom procedurom detektovana nezadovoljivost, onda je polazna formula G nezadovoljiva. Inače, ako se procedura zaustavi dajući zadovoljavajuću interpretaciju za G_a, onda se može pokazati da je formula G zadovoljiva pravljenjem modela za nju ako su sve implicitne jednakosti iz GE i G' eksplicirane Fourieorovim algoritmom i baznim upotpunjavanjem. Pod pretpostavkom da postoji način da se Fourierov algoritam proširi tako da daje sve implicitne jednakosti, opisana procedura predstavlja proceduru odlučivanja za Prezburgerovu aritmetiku sa neinterpretiranim funkcijskim simbolima. Za racionalne brojeve moguće je na opisani način proširiti Fourierovu proceduru tako da detektuje sve implicitne jednakosti, a za cele i prirodne brojeve to je moguće uraditi, na primer, u stilu Nelson/Oppenove procedure (videti 3.1).

3.4.5 Linearna aritmetika sa dopustivim sistemom za prezapisivanje

Ako je $\mathcal{R}$ sistem za prezapisivanje takav da za svaki konačan skup baznih jednakosti E izraženih simbolima iz $\mathcal{R}$ i konstantama, i ako se upotpunjavanje zaus-

tavlja dajući kanonski sistem za prezapisivanje $\mathcal{R}'$, onda $\mathcal{R}'$ može biti upotrebljen za ispitivanje da li neka jednakost sledi iz skupa jednakosti E. Kanonski sistem $\mathcal{R}$ koji ispunjava taj uslov zovemo *dopustivim* (eng. admissible) sistemom. Za dopustiv sistem $\mathcal{R}$, njegova teorija bez kvantifikatora je odlučiva korišćenjem upotpunjavanja. Nije, međutim, dovoljno normalizovati tvrđenje primenom dopustivog sistema, već je potrebno pravila superponirati sa jednakostima iz datog tvrđenja. Na primer, može se pokazati da je sistem za prezapisivanje $\{f(f(x)) \rightarrow x\}$ dopustiv. Neka je $f(a) = f(b) \wedge a \neq b$ formula koju treba opovrgnuti i neka je jednakost $f(a) = f(b)$ orijentisana na sledeći način: $f(a) \rightarrow f(b)$. U tom slučaju $f(a)$ se superponira sa $\{f(f(x)) \rightarrow x\}$ dajući novu jednakost $a = f(f(b))$ sa normalizovanim oblikom $a = b$. Nakon toga trivijalno se pokazuje da je formula $a \neq b \wedge a = b$ nezadovoljiva.

Opisana procedura odlučivanja za teoriju dopustivog sistema bez kvantifikatora može biti kombinovana sa procedurom odlučivanja za Prezburgerovu aritmetiku sa neinterpretiranim funkcijskim simbolima. Slično kao i u slučaju Prezburgerove aritmetike sa neinterpretiranim funkcijskim simbolima, strani funkcijski termovi se apstrahuju uvođenjem novih konstanti. Ove i druge bazne jednakosti se zatim upotpunjavaju korišćenjem procedure za bazno upotpunjavanje, a dodatne jednakosti sa dobijaju superponiranjem baznih jednakosti sa pravilima iz $\mathcal{R}$. Nove jednakosti se koriste za normalizovanje linearnih nejednakosti, iz kojih se izvode dodatne implicitne jednakosti. Ovako povezane komponente daju proceduru odlučivanja za slučaj kada su interpretirani simboli aksiomatizovani dopustivim sistemom za prezapisivanje.

Naglasimo da postoje jednakosne teorije koje imaju odlučive teorije bez kvantifikatora, ali nemaju dopustive sisteme za prezapisivanje. Štaviše, postoje i jednakosne teorije sa odlučivim teorijama bez kvantifikatora koje imaju kanonske sisteme za prezapisivanje, ali oni nisu dopustivi [68].

3.4.6 Integrisanje Fourierovog algoritma u sistem za prezapisivanje

U slučaju interpretiranih funkcija često je neophodno korišćenje dodatnih lema, odnosno pravila prezapisivanja. U sistemu TECTON implementirana je i sa uspehom se primenjuje proširena Fourierova procedura u kombinaciji sa baznim upotpunjavanjem i mehanizmom za korišćenje lema. Procedura može biti opisana na sledeći način: neka je G formula bez kvantifikatora koju treba dokazati; razmatra se njena negacija i ona se skolemizuje. Potrebno je pokazati da dobijena formula nije zadovoljiva. Dobijena formula deli se na više formula deljenjem na pozicijama gde postoji disjunkcija. Za svaki od podciljeva koji je oblika $L_1 \wedge ... \wedge L_k$ (L_i su literali) treba pokazati da je nezadovoljiv. Iz te konjunkcije izdvaja se najpre skup GE baznih jednakosti (uključujući literale kao što je $p(a) = true$ ili $p(a) = false$). Nakon toga primenjuju se sledeći koraci:

- Bazno upotpunjavanje se primenjuje da dâ kanonski sistem za prezapisavanje za bazne jednakosti. Ako je kontradikcija otkrivena, podcilj je nezadovoljiv. Inače, dobijeni kanonski sistem se koristi za normalizovanje

nejednakosti.

- Normalizovane nejednakosti se prosleđuju Fourierovom algoritmu za eliminaciju.

  (a) Kad god je implicitna jednakost generisana, ide se na korak 1.

  (b) Ako je otkrivena kontradikcija, podcilje je nezadovoljiv

  Ako nakon koraka 1 i 2 nije napravljen progres, idi na sledeći korak

- Bira se maksimalni literal/term između svih literala/termova i traži se lema koja može da bude instancirana tako da uključuje taj literal/term (analogno heuristici augmentacije, videti 3.3.4). Ukoliko je lema uslovna, njeni uslovi se proveravaju rekurzivno i to korišćenjem kontekstualnog prezapisivanja (videti B.5 i [124]).

Opisana procedura je očito saglasna i zaustavlja se (jer se bazno upotpunjavanje uvek zaustavlja i jer se koristi dobro zasnovano uređenje termova). Procedura nije potpuna, čak i ako raspoložive leme indukuju zadato tvrđenje: na primer, neka su raspoložive sledeće dve leme: $q \Rightarrow \neg p(a, x)$ i $\neg q \Rightarrow \neg p(x, b)$. Tada važi $\neg p(a, b)$, ali nezadovoljivost literala $p(a, b)$ ne može biti dokazana niti korišćenjem prvog ni drugog pravila [124]. Nepotpunost opisane procedure, ne smatramo, međutim, njenim ozbiljnim nedostatkom: cilj opisane procedure je proširivanje domena procedure odlučivanja za Prezburgerovu aritmetiku i to na delove koji su neodlučivi.

3.5 Ograničeno kontekstualno prezapisivanje

Ograničeno kontekstualno prezapisivanje (eng. constraint contextual rewriting) Armanda i Ranisea je uopštenje kontektualnog prezapisivanja koje uključuje funkcionalnosti procedure odlučivanja za neku teoriju [2, 4]. Uloga procedure odlučivanja se karakteriše apstraktno i oznaka CCR(X) se koristi da bi se naglasila nezavisnost sistema CCR od teorije koju odlučuje procedura odlučivanja. Ograničeno kontekstualno prezapisivanje odnosi se na problem ugradnje jedne procedure odlučivanja u dokazivač teorema (i time, na proširivanje njenog domena), ali ne i na problem kombinovanja više procedura odlučivanja). Može se pokazati da su integracione sheme za procedure odlučivanja primenjene u sistemima NQTHM i TECTON instance sistema CCR(X).

Na bazi sistema CCR razvijen je dokazivač RDL (Rewrite and Decision procedures Laboratory) [1].

3.5.1 Kontekstualno prezapisivanje

Kontekstualno prezapisivanje (eng. contextual rewriting) je uopštenje uslovnog prezapisivanje i predstavlja moćno pravilo izvođenja. Da bi se pojednostavio term t kontekstualnim prezapisivanjem, pored skupa uslovnih pravila prezapisivanja R, koristi se i skup jednakosti C koji zovemo *kontekst* (eng. context)

terma t. Ako je kontekst prazan, kontekstualno prezapisivanje se svodi na uslovno prezapisivanje. U automatskom dokazivanju teorema, kada se dokazuje nezadovoljivost konjunkcije litetala, u prezapisivanju jednog literala kao kontekst koriste se preostali literali; u direktnom dokazu, kada se dokazuje da važi disjunkcija literala, u prezapisivanju jednog literala kao kontekst koriste se negacije preostalih literala. U nastavku dajemo formalnu definiciju kontekstualnog prezapisivanja (detaljan opis kontekstualnog prezapisivanja i njegovih različitih varijanti dat je u radu [124]).

Definicija 3.5 Konstantno kongruentno zatvorenje $\simeq_C$ *generisano skupom jednakosti* C *je minimalna relacija ekvivalencije koja zadovoljava sledeće uslove:*

- *za svaku jednakost* $t_1 = t_2$ *of* C*, važi* $t_1 \simeq_C t_2$*;*
- *ako je* $t_1 \simeq_C t_2$*, onda važi* $f(\ldots t_1 \ldots) \simeq_C f(\ldots t_2 \ldots)$*.*

Definicija 3.6 *Za dati skup jednakosti* C *i za skup uslovnih pravila prezapisivanja* R*, prezapisivanje u kontekstu* C *se definiše rekurzivno na sledeći način: term* t *se prezapisuje u term* t' *kontekstualnim prezapisivanjem korišćenjem skupova* R *i* C *(i pišemo* $t \longrightarrow_{C,R} t'$*) ako važi:*

- $t \simeq_C t'$ *ili*
- *postoji podterm* t'' *terma* t*, pravilo prezapisivanja* $l \longrightarrow r$ **if** $\{p_1, p_2, \ldots, p_k\}$ *u skupu* R *i supstitucija* σ *takvi da je*
  - $l\sigma = t''$
  - *za svako* p_i*,* $1 \leq i \leq k$*,* $p_i\sigma \rightarrow^*_{C,R} true$
  - *term* t' *jednak je termu* t *u kojem je podterm* t'' *zamenjen termom* $r\sigma$*.*

  Kažemo da je pravilo $l \longrightarrow r$ *if* $\{p_1, p_2, \ldots, p_k\}$ primenljivo *ako su zadovoljena pravila (1) i (2).*

Kontekstualno prezapisivanje može da se koristi i u deduktivnom i u induktivnom dokazivanju teorema. U deduktivnom dokazivanju, kontekstualno prezapisivanje je moćno pravilo koje strogo uključuje nekoliko pravila pojednostavljivanja kao što su brisanje tautologija, uključivanje, demodulacija, ali i pravila kao što je augmentacija (ili njegove varijante).

3.5.2 Dokazivački specijalisti

Procedura odlučivanja za neku teoriju koja samo daje odgovore "da" i "ne" najčešće nije adekvatna u praktičnim primenama, već je potrebno da ona može da radi inkrementalno, tj. da obrađuje delove tekućeg problema, kao i da može da "normalizuje" zadate izraze u odnosu na trenutno raspoložive informacije. Ta znanja, specifična za neku teoriju, su u ograničenom kontekstualnom prezapisivanju sadržana u tzv. dokazivačkim specijalistima (eng. reasoning specialist).

Dokazivački specijalista je procedura zasnovana na stanjima — tzv. bazama ograničenja (eng. constraint stores) — koja su konačni skupovi literala reprezentovanih u nekoj internoj formi i čija funkcionalnost je apstraktno karakterisana na sledeći način (T_c je teorija odlučiva teorija za koju je raspoloživa procedura odlučivanja; T_j je proširenje teorije T_c):

Inicijalizacija baze ograničenja `cs-init`(C) važi ako i samo ako je baza ograničenja "prazna", tj. ako je skup ograničenja C valjan u teoriji T_c;

Otkrivanje protivrečnosti `cs-unsat`(C) važi ako i samo ako je skup ograničenja C T_c–protivrečan.

Pojednostavljivanje baza ograničenja Pojednostavljivanje baza ograničenja je transformisanje jedne baze ograničenja u drugu, pri čemu se postavlja uslov saglasnost i uslov da je dobijena baza ograničenja manja od polazne u uređenju svođenja koje se koristi.

Normalizacija Normalizacija je proces izračunavanja normalne forme formula/-termova u odnosu na ograničenja koja sadrži baza C. Normalizacija čuva jednakost i dobijena formula/term mora da bude manja od polazne u uređenju svođenja koje se koristi.

3.5.3 Pojednostavljivanje klauza, prezapisivanje i augmentacija

U ograničenom kontekstualnom prezapisivanju, osnovno kontekstualno prezapisivanje je prošireno specifičnim funkcionalnostima procedure odlučivanja (odnosno dokazivačkim specijalistima), a ostale forme prezapisivanja se dodatno specifikuju novim pravilima (npr. pravilom `cl-true` koje prezapisuje $E \cup \{true\}$ u $\{true\}$).

Augmentacija je zasnovana na idejama primenjenim u sistemima NQTHM i TECTON, kao i u kontekstualnom prezapisivanju.

3.5.4 Svojstva ograničenog kontekstualnog prezapisivanja

Shema CCR može biti instancirana na različite integracione metode (uključujući Boyer/Mooreov, Kapur/Nieov metod i Zhangovo kontekstualno prezapisivanje).

U radu [4] dokazani su saglasnost i zaustavljanje ograničenog kontekstualnog prezapisivanja za apstraktnu shemu. Ti dokazi se ne zasnivaju na nekim specifičnim svojstvima konkretnih teorija, već na skupu opštih uslova i važe za CCR(X) instanciran bilo kojom teorijom X koja zadovoljava te opšte uslove. Svi relevantni uslovi sadržani su u specifikaciji dokazičkih specijalista i drugih pravila izvođenja. Oni su zasnovani, između ostalog, na nekom uređenju svođenja koje služi kao mera stanja procedure. Tokom procesa prezapisivanja, u tom uređenju, niz formula opada i time je obezbeđeno zaustavljanje.

Razmatranje kompletnosti sistema CCR(X) za neke teorije X zahtevalo bi dodatno specifikovanje dokazivačkih specijalista. Naime, dokazivački specijalisti su preslabo specifikovani i, za trenutnu verziju sistema CCR, nemoguće je

dokazati potpunost za osnovnu teoriju (ili za osnovnu teoriju sa čistom teorijom jednakosti); ova slaba specifikacija čini CCR veoma opštom shemom, ali u isto vreme onemogućava dokazivanje bilo koje forme potpunosti (i čini se da je najslavija tačka sheme CCR).

3.6 Barrett/Dill/Stumpov okvir za kombinovanje procedura odlučivanja

U ovom poglavlju dajemo kratak pregled osnovnih ideja iz nedavno objavljenog rada Barretta, Dilla i Stumpa u kojem se predlaže jedan fleksibilni okvir za kombinovanje procedura odlučivanja[4] [10]. Okvir je motivisan, pre svega, potrebom da se Nelson/Oppenova i Shostakova shema implementiraju na uniforman način, da se kombinuju i da se o njima može formalno rezonovati. U tekućoj verziji shema se odnosi na disjunktne konveksne stabilno–beskonačne teorije prvog reda bez kvantifikatora. Okvir se oslanja na specifične strukture podataka odnosno atribute (za ažuriranje vrednosti `find`) i po tome je sličan drugim shemama za kombinovanje procedura odlučivanja.

3.6.1 Osnovni okvir

U Barrett/Dill/Stumpov okviru svaki izraz ima atribut `find` koji oslikava klase ekvivalencije između izraza; taj atribut indukuje relaciju $\sim$ nad izrazima koja je relacija ekvivalencije skupa obrađenih izraza i on odgovara $find$ funkciji u Shostakovom algoritmu. Dokazivanje teoreme se realizuje kroz proveravanje zadovoljivosti skupova literala, pri čemu je skup literala zadovoljiv ako logičke konstante $true$ i $false$ nisu u istoj klasi ekvivalencije indukovanoj relacijom $\sim$. Okvir ima interfejs sa komunikaciju sa spoljašnjim programima koji čine sledeće tri funkcije: `AddFormula` (dodaje literal skupu $\mathcal{A}$ koji zovemo *istorija pretpostavki* (eng. assumption history)); `Satisfiable` (vraća $false$ ako i samo ako je $T \cup \mathcal{A} \models false$); `Simplify` transformiše izraz u novi izraz koji je ekvivalentan moduo $T \cup \mathcal{A}$. Dodatno, postoje funkcije koje su specifične za konkretne teorije, kao što su `TheorySetup`, `TheoryRewrite` i `PropagateEqualities`. Okvir je definisan ovim, kao i pomoćnim funkcijama za određivanje i ažuriranje klasa ekvivalencije relacije $\sim$. Za svaku od osnovnih funkcija okvir sadrži preduslove i postuslove (izražene na pseudokodu). Da bi se govorilo o korektnosti okvira i njegovih instanci, specifikacija se zadaje u terminima preduslova i postuslova. Često je potrebno govoriti o *stanju* (eng. state) programa. Svako izvršavanje programa se smatra nizom stanja, pri čemu stanje uključuje vrednosti promenljivih koje se koriste i lokaciju u kôdu. Ukoliko su preduslovi i postuslovi zadovoljeni za svaku funkciju primenjenu do određene lokacije u programu, kaže se da je program u *nenarušenom* (eng. uncorrupted) stanju. Takođe, ako je $true \not\sim false$, kaže se da je program u *konzistentnom* (eng. consistent) stanju. Procedura za

[4] Radovi [10] i [59] sa analognim imenima (prezentovani na dve uzastopne konferencije CADE) i slična po duhu, su paralelna u dva istraživačka pravca — u kombinovanju i integrisanju procedura odlučivanja.

ispitivanje zadovoljivosti u osnovnom okviru Barretta, Dilla i Stumpa je *saglasna* ako važi $T \cup \mathcal{A} \models false$ uvek kada je program u nekonzistentnom stanju. Može se dokazati da ako je program u nenarušenom stanju, onda je procedura zadovoljivosti saglasna. Procedura zaustavljanja je *potpuna* ako je $T \cup \mathcal{A}$ zadovoljivo uvek kada je program u konzistentnom stanju. Može se dokazati da je procedura potpuna ako je ona uvek u nenarušenom stanju i ako određena svojstva važe za svaku pojedinačnu teoriju na koju se ona odnosi. Uz određene pretpostavke svaka funkcija, pa i čitava procedura se zaustavlja.

3.6.2 Primeri instanciranih procedura i njihova svojstva

Funkcije opšteg okvira Barretta, Dilla i Stumpa mogu se instancirati tako da okvir odgovara Nelson/Oppenovoj ili Shostakovoj proceduri za kombinovanje procedura odlučivanja. Za tako instancirani okvir, tj. za tako dobijene procedure može se dokazati da su zadovoljeni uslovi za potpunost i zaustavljanje. Dodavanje algebarski rešivih teorija Nelson/Oppenovoj proceduri (tj. kombinovanje Shostakove i Nelson/Oppenove procedure) je takođe moguće, pri čemu se mora voditi računa o tome da uslovi potrebni za potpunost i zaustavljanje ostanu da važe.

3.7 Schemes for Combination and Integration of Decision Procedures into Theorem Provers: Summary

In the late 70s and early 80s, interest in decision procedures for simple theories and for their use in theorem proving was revived. This interest was supported by influential schemes for combinations of decision procedures reported by Nelson and Oppen [86] and by Shostak [104]. Along the line of research on incorporating decision procedures into heuristic theorem provers, one of the most influential approaches was reported in the late 80s by Boyer and Moore [17] and successfully applied in their NQTHM. In this chapter we briefly describe these and three other schemes which will be also discussed in the rest of the thesis. The macro inference rules presented in the following chapter are motivated by these approaches.

Nelson and Oppen's algorithm for combination of theories. This approach [86] gives a decision procedure for a combination of (decidable) quantifier–free theories which do not share logical symbols other than equality (provided there are decision procedures for these theories). Alien terms in literals are abstracted and literals making the conjunction being refuted are then divided into groups according to theories to which they belong; after that, each decision procedure tries to deduce some new equality and to propagate it to other theories. In this approach, deducing of a new equality E in some theory can be based on a decision procedure for that theory (i.e. on checking unsatisfiability of the conjunction

of literals augmented by $\neg E$), but more efficient algorithms for specific theories can also be used for that purpose. Nelson/Oppen's algorithm is a decision procedure for a combination of stably–infinite component theories [120] (A consistent, quantifier free theory $\mathcal{T}$ with signature Σ is called stably–infinite iff any quantifier-free Σ-formula is satisfiable in $\mathcal{T}$ iff it is satisfiable in an infinite model of $\mathcal{T}$ [91].). An analysis of Nelson/Oppen's procedure and its theoretical background can be found in [120].

Shostak's algorithm for combination of theories. Shostak's approach [104] gives a procedure for a combination of *canonizable* and *algebraically solvable* quantifier–free equational theories. It is based on *solvers* for specific theories which are tightly combined by an efficient underlying congruence closure algorithm for ground equalities. This algorithm deals merely with equational theories and, for instance, cannot deal with linear arithmetic inequalities. (In Shostak's original implementation linear arithmetic inequalities were treated by an external decision procedure, in the Nelson/Oppen style). A rigorous analysis of Shostak's algorithm can be found in [32]. Recently, Rueß and Shankar [100] showed that Shostak's procedure and all its published variants [32, 9, 11] are incomplete and even can be non-terminating; they also present one new variant of Shostak's procedure and prove its termination, soundness and completeness. There is also another recent variant of Shostak's algorithm reported by Barrett, Dill and Stump [10].

As for Nelson/Oppen algorithm, Shostak's algorithm in its basic form does not address the problem of using additional rewrite rules, lemmas etc.

Boyer/Moore's linear arithmetic procedure. Boyer and Moore's algorithm [17] extends a realm of one decision procedure and does not combine decision procedures for different theories. In their approach, a decision procedure for Presburger arithmetic (based on Hodes' algorithm [52]) is combined with a heuristic *augmentation* which provides information about alien functions (i.e. provides lemmas) and with a technique for using the remaining literals in the clause being proved as the context to simplify one literal. In this approach, an underlying decision procedure for Presburger arithmetic can't be changed by another one, which makes the system highly inflexible. Despite the successful use of this algorithm in NQTHM and its general influence (not only on the integration of decision procedures into heuristic provers) it hasn't gained wider popularity and there are very few systems which use it. One of the reasons for it is probably the complicated original description which is often given in terms of special data-structures rather than in terms of their logical meaning and is often given intermixed with different optimisations. All that also makes any theoretical analysis of Boyer and Moore's algorithm a very difficult task.

Reasoning about numbers in TECTON. Kapur and Nie's algorithm for reasoning about numbers [68] is based on Boyer and Moore's approach, but

improves it in several ways. It is designed so that the underlying reasoning about Presburger formulae can be handled by different procedures. It makes the system much more flexible than Boyer and Moore's procedure. The original description is based on a form of Fourier's algorithm for Presburger arithmetic; as a decision procedure it is essentially the same as one due to Hodes' [52] used by Boyer and Moore, but in its extended form [74], it also provides a mechanism for deducing implicit Presburger equalities (which are efficiently used for further rewriting of remaining parts of a formula being proved). This scheme uses literals in the formula being proved as the context to simplify one literal, similarly as in Boyer and Moore's procedure, but described more precisely via congruence closure and contextual rewriting [124]. This approach uses rewriting techniques and is used in the rewrite based prover TECTON.

Constraint contextual rewriting. Armando and Ranise's constraint contextual rewriting [2, 4] is a formal system motivated by the ideas from Boyer and Moore's system. Constraint contextual rewriting is a extended form of contextual rewriting [124] which incorporates the functionalities provided by some decision procedure. It provides a flexible framework which can be instantiated by a specific decision procedure X for some theory and it can formally cover approaches used by Boyer and Moore and by Kapur and Nie (or, at least, their essential parts). The soundness and termination of the constraint contextual rewriting is proved formally for the abstract scheme when certain requirements on the rewriting mechanism and the decision procedure are satisfied [4].

As the previous two approaches, constraint contextual rewriting does not address the problem of combining decision procedures, but only the problem of extending the realm of one decision procedure by additional rewrite rules.

Barrett/Dill/Stump's framework for combining decision procedures. Barrett, Dill and Stump's recent work [10] gives a flexible framework for cooperating decision procedures. This framework uses several general functions (such as `AddFormula`, `Satisfiable` and `Simplify`) and some theory specific functions (such as functions `TheorySetup`, `TheoryRewrite` and `PropagateEqualities`). The framework uses a specific `find` attribute on expressions (which corresponds to Shostak's $find$ function) which implies an equivalence relation $\sim$. A processed expression is satisfiable if it does not imply $true \sim false$. There are abstract requirements concerning soundness, completeness and termination for the abstract framework. The framework can be instantiated to Nelson/Oppen style or to Shostak style procedure. For these instances of the framework it can be proved that the needed requirements are fulfiled and, hence, they are sound, complete and terminating procedures. In addition, the framework enables combining Nelson/Oppen and Shostak style procedures.

Glava 4

Fleksibilno proširivanje procedura odlučivanja

U ovom delu teze dajemo opis fleksibilnog okvira, tj. apstraktne sheme EPM (extended proof method) za integrisanje procedure odlučivanja u dokazivač teorema. EPM shema zasnovana je na idejama Boyer/Moorove procedure za linearnu aritmetiku, pri čemu ona predstavlja njeno značajno uopštenje i značajnu formalizaciju. EPM shema omogućava proširivanje domena procedure odlučivanja A za teoriju $\mathcal{T}$ (npr. za Prezburgerovu aritmetiku) korišćenjem raspoloživih lema. Predložena shema može biti primenjena u različitim dokazivačima, za različite teorije i za različite procedure odlučivanja za te teorije. Nova procedura odlučivanja može jednostavno biti dodata u sistem samo ako pruža ograničeni skup funkcionalnosti kao što je proveravanje zadovoljivosti i eliminacija jedne promenljive. Zaustavljanje i saglasnost EPM sheme mogu se dokazati za apstrakni oblik (podrazumevajući da je on instanciran konkretnom teorijom koja zadovoljava date uslove). Shema se ne odnosi na kombinovanje više procedura odlučivanja, već samo na problem proširivanja domena jedne procedure odlučivanja dodatnim pravilima prezapisivanja tj. lemama. EPM shema (i neke njene instance) implementirana je u okviru sistema *Clam*, baziranom na paradigmi planiranja dokaza. Opis te implementacije dat je u delu 6.

U nastavku teksta podrazumeva se da se EPM shema odnosi na odlučivu teoriju $\mathcal{T}$ (nad jezikom $\mathcal{L}$) koja dopušta eliminaciju kvantora i na neko njeno (u opštem slučaju neodlučivo) konzervativno proširenje $\mathcal{T}^e$ nad jezikom $\mathcal{L}^e$

4.1 Najteži ne-$\mathcal{T}$-term

Tvrđenje koje treba dokazati nekom konkretnom procedurom odlučivanja za neku teoriju $\mathcal{T}$ često pada "tik izvan njenog domena", tj. formula F koju treba dokazati često je $\mathcal{T}^e$-formula, ali ne i $\mathcal{T}$-formula (za neke teorije $\mathcal{T}$ i $\mathcal{T}^e$). U tom slučaju, i ako je skup $\Pi^e \setminus \Pi$ prazan, to znači da F sadrži *ne-$\mathcal{T}$-termove* (tj. termove strane u odnosu na teoriju $\mathcal{T}$). U daljem tekstu podrazumevaćemo,

jednostavnosti radi, ako nije drugačije naglašeno, da je $\Pi^e \setminus \Pi = \emptyset$. Pretpostavljamo i da svakom funkcijskom simbolu iz Σ^e pridružena torka oblika $\langle \tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}, \tau \rangle$ (tj. oblika $\tau_{i_1} \times \tau_{i_2} \times \cdots \times \tau_{i_k} \to \tau$) gde je $\tau \in T$.

Definicija 4.1 *Za $\mathcal{T}^e$-termove, definišemo skupove* ne-$\mathcal{T}$-termova *na sledeći način:*

- *ako je t $\mathcal{T}$-term, onda je njegov skup ne-$\mathcal{T}$-termova prazan;*
- *u $f(t_1, t_2, \ldots, t_n)$, gde je $f \notin \Sigma$, skup ne-$\mathcal{T}$-termova je $\{f(t_1, t_2, \ldots, t_n)\}$;*
- *u $f(t_1, t_2, \ldots, t_n)$, gde je $f \in \Sigma$ (i $n \geq 0$), skup ne-$\mathcal{T}$-termova je unija skupova ne-$\mathcal{T}$-termova u termovima t_j ($j = 1, 2, \ldots, n$).*

Definicija 4.2 *Za $\mathcal{T}^e$-formulu i $\mathcal{T}^e$-atomičku formulu, definišemo skupove* ne-$\mathcal{T}$-termova *na sledeći način:*

- *skup ne-$\mathcal{T}$-termova u $\forall x\ F$, $\exists x\ F$, $\neg F$ jednak je skupu ne-$\mathcal{T}$-termova u F;*
- *skup ne-$\mathcal{T}$-termova u $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \to F_2)$ jednak je uniji skupova ne-$\mathcal{T}$-termova u F_1 i u F_2;*
- *u $p(t_1, t_2, \ldots, t_n)$, gde je $p \in \Pi$, skup ne-$\mathcal{T}$-termova jednak je uniji skupova ne-$\mathcal{T}$-termova u termovima t_j, ($j = 1, 2, \ldots, n$);*
- *u $t_1 = t_2$, skup ne-$\mathcal{T}$-termova jednak je uniji skupova ne-$\mathcal{T}$-termova u termovima t_j, ($j = 1, 2$);*
- *u $\top$ i u $\bot$ skupovi ne-$\mathcal{T}$-termova su prazni.*

U ispitivanju da li je $\mathcal{T}^e$-formula F teorema, opravdano je pokušati sa korišćenjem procedure odlučivanja za teoriju $\mathcal{T}$ (ili korišćenjem same procedure odlučivanja ili u kombinaciji sa pogodno odabranim lemama). Dakle, potrebno je transformisati formulu F u neku odgovarajuću $\mathcal{T}$-formulu. To možemo da postignemo generalizacijom: u $\mathcal{T}^e$-formuli F, generalizujemo ne-$\mathcal{T}$-termove (u smeru od spolja ka unutra) novim promenljivama na sledeći način: zamenjujemo svaki ne-$\mathcal{T}$-term sorte τ novom promenljivom iste sorte i zatim prelazimo na univerzalno zatvorenje formule F. Kada svi funkcijski simboli iz Σ^e imaju ili sortu τ ili sortu $\tau_{i_1} \times \tau_{i_2} \times \cdots \times \tau_{i_k} \to \tau$, gde je $\tau \in T$, onda je formula F' dobijena na opisani način $\mathcal{T}$-formula (sa mogućim izuzetkom koji čine neki redudantni kvantifikatori čije sorte nisu u T). Na primer, formula proširene Prezburgerove aritmetike

$$\forall \alpha\ (min(\alpha) \leq max(\alpha))$$

(gde je α sorte `list of pnats`) može biti transformisana u

$$\forall min \forall max \forall \alpha\ (min \leq max)$$

(gde su *min* i *max* sorte **N**). U ispitivanju da li je F' teorema, koristimo proceduru odlučivanja za $\mathcal{T}$ najpre eliminišući sve nove promenljive (ili korišćenjem same procedure odlučivanja ili u kombinaciji sa lemama). Pogodno je odložiti (koliko je to moguće) upotrebu lema jer one potencijalno uvode nove strane termove. Dodatno, poželjno je eliminisati promenljive dobijene od najsloženijih termova. Zato je potrebno uvesti neko totalno uređenje nad $\mathcal{T}^e$-termovima. Najpre definišimo funkciju $|\ .\ |$ koja preslikava skup $\mathcal{T}^e$-termova u skup prirodnih brojeva (vrednost te funkcije odgovara *veličini* terma).

Definicija 4.3 *Ako je $\mathcal{T}^e$-term t funkcijski simbol sorte τ ($\tau \in T$) ili je promenljiva, onda je $|t| = 1$. Ako je $\mathcal{T}^e$-term t term oblika $f(t_1, t_2, \ldots, t_n)$, onda je $|t| = 1 + \sum_{i=1}^{n} |t_i|$.*

Definicija 4.4 *$\mathcal{T}^e$-term t_1 je* teži *od $\mathcal{T}^e$-terma t_2 ako i samo ako važi:*

- *$|t_1| > |t_2|$ ili*
- *$|t_1| = |t_2|$ i dominantan simbol terma t_1 u alfabetskom poretku dolazi iza dominantnog simbola terma t_2 ili*
- *$|t_1| = |t_2|$, $t_1 = f(t'_1, t'_2, \ldots, t'_n)$, $t_2 = f(t''_1, t''_2, \ldots, t''_n)$ i postoji vrednost k $(1 \leq k \leq n)$ takva da su t'_i i t''_i identični termovi (za $i < k$) i t'_k je teži od t''_k.*

Definicija 4.5 *Ako su termovi t_1 i t_2 identični ili je t_2 je teži od t_1, onda $t_1 \preceq t_2$.*

Definicija 4.6 *$\mathcal{T}^e$-term t je* najteži ne-$\mathcal{T}$-term *u nekom skupu S $\mathcal{T}^e$-termova, ako je t ne-$\mathcal{T}$-term i za svaki ne-$\mathcal{T}$-term t' iz skupa S važi $t' \preceq t$.*

Relacija $\preceq$ predstavlja totalno uređenje u skupu $\mathcal{T}^e$–termova i ovo uređenje ispunjava sledeći uslov: za konačno mnogo promenljivih i funkcijskih simbola, za svaki term t postoji konačno mnogo termova t' takvih da je $t' \preceq t$ (što je ključni uslov za zaustavljanje). Relacija $\preceq$ za uređenje termova (zasnovana na složenosti termova i alfabetskom poretku) je analogna onoj koja se koristi u Boyer/Moorovoj proceduri, a umesto nje moguće je koristiti i neko uređenje svođenja, kao što je rekurzivno uređenje sa statusom (videti dodatak B).

4.2 Procedure pojednostavljivanja

U ovom poglavlju opisujemo četiri procedure za pojednostavljivanje (i za smanjivanje broja kvantifikatora) $\mathcal{T}^e$-formule koja se dokazuje. Svaka od njih može biti upotrebljena u različitim kontekstima pojednostavljivanja formula, u kombinaciji sa drugim komponentama dokazivača, a one zajedno mogu da formiraju proširenje procedure odlučivanja za teoriju $\mathcal{T}$. Sve četiri procedure primenljive su samo na $\mathcal{T}^e$-formule.

Pretpostavimo da je za teoriju $\mathcal{T}$ raspoloživa procedura odlučivanja zasnovana na eliminaciji kvantifikatora; pretpostavimo da su raspoložive dve procedure[1]: primenljive na $\mathcal{T}$-formule — $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ i $\texttt{DpGround}_{\mathcal{T}}$:

- neka je $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ procedura koja datu $\mathcal{T}$-formulu F sa kvantifikatorima[2] transformiše u preneks normalnu formu i eliminiše unutrašnji kvantifikator (i odgovarajuću promenljivu), tj. vraća formulu F' takvu da važi $\mathcal{T} \vdash F$ ako i samo ako $\mathcal{T} \vdash F'$, F' je u preneks normalnoj formi i broj kvantifikatora u formuli F' je manji (bar za jedan) nego broj kvantifikatora u formuli F.

- neka je $\texttt{DpGround}_{\mathcal{T}}$ procedura koja za datu baznu $\mathcal{T}$-formulu F vraća $\top$ ako je $\mathcal{T} \vdash F$ i vraća $\bot$ ako je $\mathcal{T} \nvdash F$.

Za mnoge odlučive teorije postoje procedure koje zadovoljvaju navedene uslove, tj. procedure zasnovane na eliminaciji kvantifikatora (videti A.2.1 i [72]).

Primenu narednih procedura razmatraćemo samo za univerzalno zatvorena tvrđenja i leme.

4.2.1 Procedura odlučivanja za $\mathcal{T}$

Neka je procedura $\texttt{Dp}_{\mathcal{T},A}$ definisana na sledeći način: ako je $\mathcal{T}$–formula koja se dokazuje bazna, onda primeni $\texttt{DpGround}_{\mathcal{T}}$; inače (tj. ako $\mathcal{T}$–formula koja se dokazuje nije bazna) primeni $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$.

Primetimo da je $\texttt{Dp}_{\mathcal{T},A}$ procedura odlučivanja za teoriju $\mathcal{T}$, tj. $\texttt{Dp}_{\mathcal{T},A}$ transformiše $\mathcal{T}$-formulu F u $\top$ ako je $\mathcal{T} \vdash F$ i u $\bot$, inače.

Procedura odlučivanja za teoriju $\mathcal{T}$ definisana na ovaj način je fleksibilnija, ali obično nešto sporija od kompaktnijih procedura iz kojih su izbačena nepotrebna ponavljanja nekih algoritamskih koraka (obično nekih nepotrebnih normalizacija). Izložena proširena procedura odlučivanja može da koristi i neku drugu proceduru odlučivanja za teoriju $\mathcal{T}$ umesto $\texttt{Dp}_{\mathcal{T},A}$.

4.2.2 Eliminacija redudantnog kvantifikatora

Procedura `ElimRedundantQuantifier` za eliminaciju redudantnog kvantifikatora je veoma jednostavna: ako postoji redudantan kvantifikator (nevažno koje sorte) u $\mathcal{T}^e$-formuli F koja se dokazuje, eliminiši ga.

Dobro je da su raspoloživa dodatna pravila pojednostavljivanja za teoriju $\mathcal{T}$ takva da ih je moguće primeniti pre `ElimRedundantQuantifier` (ali to nije neophodno). Na primer, u PNA, moguće je svaku atomičku formulu svesti na oblik u kojem se svaki term pojavljuje najviše jednom (na primer, $k \leq max(a) + k$ može biti svedeno na $0 \leq max(a)$).

[1] Često i ove procedure mogu biti implementirane na fleksibilan način: kao primene serija pravila prezapisivanja (videti poglavlje D.5 i [22])

[2] Formula F može da sadrži redudantne kvantifikatore sorti koje nisu u T. Ova procedura ignoriše te kvantifikatore (ali ih ne eliminiše).

4.2.3 Eliminacija $\mathcal{T}$-promenljive

Neka je procedura $\texttt{ElimOneVar}_{\mathcal{T},A}$ definisana na sledeći način: ako je F $\mathcal{T}^e$-formula i ako postoji promenljiva v koja se ne pojavljuje u ne-$\mathcal{T}$-termovima formule F, onda generalizuj sve njene ne-$\mathcal{T}$-termove (u smeru od spolja ka unutra), zatim upotrebi proceduru $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ da eliminišeš promenljivu v (i odgovarajući kvantifikator) i onda zameni novouvedene promenljive polaznim, generalizovanim termovima[3]. (Primetimo da je ova procedura primenljiva i na $\mathcal{T}$-formulae, ali je razumno primenjivati je samo na formule koje su $\mathcal{T}^e$-formule ali ne i $\mathcal{T}$-formule.)

4.2.4 Eliminacija generalisanog ne-$\mathcal{T}$-terma korišćenjem lema

Procedura $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ definisana je samo za univerzalno kvantifikovane $\mathcal{T}^e$-formule. Ona je definisana na sledeći način:

- Ako je formula F koja se dokazuje $\mathcal{T}^e$-formula a nije $\mathcal{T}$-formula, odredi njen najteži ne-$\mathcal{T}$-term t;
- odredi skup $\mathcal{A}$ svih (različitih) atomičkih formula (sa njihovim polaritetima) formule F u kojima se t pojavljuje;
- Za svaki član f skupa $\mathcal{A}$, pokušaj da pronađeš raspoloživu lemu[4] $\forall x_1 \forall x_2 \ldots \forall x_n L_i$ takvu da je:[5]
  - (i) ako je t' najteži ne-$\mathcal{T}$-term[6] u L_i, i ako se on pojavljuje u atomičkoj formuli l, postoji (najopštija) supstitucija ϕ_i takva da su dominantni predikatski simboli formula f i l jednaki, f i l imaju isti polaritet (u F i u L_i respektivno), termovi t i t' pojavljuju se na istoj poziciji argumenata (u f i u l respektivno) i $t = t'\phi_i$.
  - (ii) sve promenljive u L_i su zamenjene nekim $\mathcal{T}$-termovima u F (sa odgovarajućim sortama) supstitucijom ϕ_i;

[3] Naglasimo da razmatramo samo univerzalno zatvorene formule, pa možemo slobodno promeniti poredak kvantifikatora u formuli koja je u preneks normalnoj formi; stoga procedura $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ može biti upotrebljena za eliminaciju bilo kog kvantifikatora. U opštem slučaju, $\texttt{ElimOneVar}_{\mathcal{T},A}$ mogla bi da bude definisana na sledeći način: ako je F $\mathcal{T}^e$-formula u preneks normalnoj formi i ako postoji promenljiva v sa odgovarajućim kvantifikatorom u *unutrašnjem bloku kvantifikatora* koja se ne pojavljuje u ne-$\mathcal{T}$-termovima formule F, onda generalizuj sve njene ne-$\mathcal{T}$-termove, upotrebi proceduru $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ da eliminišeš promenljivu v i onda zameni novouvedene promenljive polaznim, generalizovanim termovima.

[4] Možemo koristiti instance aksioma supstitucije kao leme, ali ih možemo koristiti i na sledeći način: ako je f dominantan funkcijski simbol terma t, za svaki par $f(t_1, t_2, \ldots, t_m)$ i $f(u_1, u_2, \ldots, u_m)$ različitih termova koji se pojavljuju u formuli koja se dokazuje koristimo formulu $t_1 = u_2 \wedge t_2 = u_2 \wedge \cdots \wedge t_m = u_m \rightarrow f(t_1, t_2, \ldots, t_m) = f(u_1, u_2, \ldots, u_m)$ kao lemu. Ovaj pristup ima širi doseg, ali je verovatno manje efikasan.

[5] Motivacija ovog koraka je u mogućnosti da takva lema sadrži informaciju dovoljnu za dokazivanje datog tvrđenja (mada nema garancija tog tipa).

[6] Razumno je ispitivati samo leme koje su $\mathcal{T}^e$-formule a nisu $\mathcal{T}$-formule; svaka lema koja je $\mathcal{T}$-formula ne može da sadrži nijednu informaciju koja ne može biti izvedena procedurom $\texttt{Dp}_{\mathcal{T},A}$; stoga, pretražujemo samo leme sa ne-$\mathcal{T}$-termovima.

(iii) najteži ne-$\mathcal{T}$-term u $L_i\phi_i$ je $t'\phi_i$.

- Novo tekuće tvrđenje koje treba dokazati je formula[7] $F \vee \neg L_1\phi_1 \vee \cdots \vee \neg L_j\phi_j$; generalizuj sve ne-$\mathcal{T}$-termove u njoj i onda upotrebi proceduru $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ da eliminišeš promenljivu v koja odgovara termu t; onda zameni novouvedene promenljive polaznim, generalizovanim termovima i vrati rezultujuću formulu kao novo tekuće tvrđenje koje treba dokazati.

4.3 Proširenje procedure odlučivanja — EPM shema

Razmatranje u ovom poglavlju i opisana shema za proširenje procedure odlučivanja (EPM shema) biće usmereno samo na univerzalno kvantifikovana tvrđenja i leme. Shema EPM definisana je na sledeći način:

(1) ako je moguće (tj. ako postoji redudantan kvantifikator), primeni proceduru `ElimRedundantQuantifier` i idi na korak (1); inače, idi na korak (2).

(2) ako je moguće (tj. ako je formula koja se dokazuje $\mathcal{T}$-formula), primeni proceduru $\texttt{Dp}_{\mathcal{T},A}$:

- ako ona vrati $\top$, onda je polazna formula teorema,
- ako ona vrati $\bot$ i
  - korak (4) nije bio primenjen, onda polazna formula nije teorema;
  - korak (4) jeste bio primenjen, onda se vrati na tu tačku primene algoritma i pokušaj da primeniš $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ na neki drugi način; ako to nije moguće, EPM se zaustavlja ne uspevši da dokaže ili opovrgne dato tvrđenje.

inače, idi na korak (3).

(3) ako je moguće (tj. ako je formula F koja se dokazuje $\mathcal{T}^e$-formula i ako postoji promenljiva v koja se ne pojavljuje u ne-$\mathcal{T}$-termovima formule F), primeni proceduru $\texttt{ElimOneVar}_{\mathcal{T},A}$ i idi na korak (1); inače, idi na korak (4).

(4) ako je moguće (tj. ako je formula koja se dokazuje $\mathcal{T}^e$-formula i nije $\mathcal{T}$-formula), primeni proceduru $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ i idi na korak (1) (ako procedura $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ može biti primenjena na više od jednog načina, onda zadrži ovu poziciju za mogući bektreking).

[7] Primetimo da vrednost j ne mora da bude jednaka broju elemenata skupa $\mathcal{A}$: za neka $\mathcal{T}^e$ tvrđenja da bi bila dokazana leme nisu potrebne; na primer, tvrđenje $\forall\alpha\ \forall k\ \ (max(\alpha) \leq k \vee max(\alpha) > k)$ može biti dokazano bez korišćenja lema. Dakle, čak i ako neke od lema sa zadatim svojstvima nisu pronađene, još uvek je simsleno pokušati sa dokazivanjem dobijenog tvrđenja uz pomoć ostalih lema i same teorije $\mathcal{T}$.

4.4 Verifikacijski primer

U ovom poglavlju primenom instancirane EPM sheme biće detaljno urađen jedan od primera iz [17]. Koristimo PNA kao teoriju $\mathcal{T}$ i Cooperov algoritam [29] kao algoritam A (videti poglavlja D.2 i D.4). Neka je $\{i, j, k, l, \ldots\}$ skup promenljivih sorte $\mathbf{N}$ i neka je $\{\alpha, \beta, \gamma, \ldots, \}$ skup promenljvih tipa `list of pnats`. Dokazujemo sledeće tvrđenje:

$$\forall l\ \forall \alpha\ \forall k\quad (l \leq min(\alpha)\ \wedge\ 0 < k\ \ \rightarrow\ \ l < max(\alpha) + k)\ .$$

(1a) Nema redudantnih kvantifikatora u datom tvrđenju, pa se ide na korak (2);

(2a) Tvrđenje nije PNA-formula, pa se ide na korak (3);

(3a) Postoji promenljiva (k) koja se ne pojavljuje u ne-PNA-termovima ($min(\alpha)$ i $max(\alpha)$); generalizujemo $min(\alpha)$ sa min, $max(\alpha)$ sa max i zatim koristimo proceduru $\texttt{ElimOneVar}_{pna,Cooper}$ da eliminišemo promenljivu k iz formule $\forall min \forall max \forall l \forall \alpha \forall k\ (l \leq min \wedge 0 < k\ \rightarrow\ l < max + k)$. Dobijamo formulu:

$$\forall min \forall max \forall l \forall \alpha\ (1 + min \leq l \vee l \leq max)\ ,$$

i nakon zamenjivanja max sa $max(\alpha)$ i min sa $min(\alpha)$ dobijamo:

$$\forall l \forall \alpha\ (1 + min(\alpha) \leq l \vee l \leq max(\alpha))\ .$$

(1b) Nema redudantnih kvantifikatora u datom tvrđenju, pa se ide na korak (2);

(2b) Tvrđenje nije PNA-formula, pa se ide na korak (3);

(3b) Postoji promenljiva (l) koja se ne pojavljuje u ne-PNA-termovima ($min(\alpha)$ i $max(\alpha)$); generalizujemo $min(\alpha)$ sa min, $max(\alpha)$ sa max i zatim koristimo proceduru $\texttt{ElimOneVar}_{pna,Cooper}$ da eliminišemo promenljivu l iz formule $\forall min \forall max \forall \alpha \forall l\ (1 + min \leq l \vee l \leq max)$. Dobijamo

$$\forall min \forall max \forall \alpha\ (min \leq max)\ ,$$

i nakon zamenjivanja max sa $max(\alpha)$ i min sa $min(\alpha)$ dobijamo

$$\forall \alpha\ (min(\alpha) \leq max(\alpha))\ .$$

(1c) Nema redudantnih kvantifikatora u datom tvrđenju, pa se ide na korak (2);

(2c) Tvrđenje nije PNA-formula, pa se ide na korak (3);

(3c) Ne postoji promenljiva koja se ne pojavljuje u ne-PNA-termovima, pa se ide na korak (4);

(4c) Najteži ne-PNA-term i $\forall\alpha\ (min(\alpha) \leq max(\alpha))$ je $min(\alpha)$. Pretpostavimo da je raspoloživa lema $L \equiv \forall\xi\ (min(\xi) \leq max(\xi))$. Postoji supstitucija $\phi = \{\xi \mapsto \alpha\}$ takva da je $(min(\alpha)) = (min(\xi))\phi$. Svi preduslovi procedure $\texttt{ElimGeneralisedTerm}_{pna,Cooper}$ su ispunjeni, pa generalizujemo sve ne-PNA-termove u formuli $\forall\alpha\ min(\alpha) \leq max(\alpha) \vee \neg(min(\alpha) \leq max(\alpha))$ — generalizujemo $min(\alpha)$ sa min i $max(\alpha)$ sa max i onda koristimo proceduru $\texttt{DpElimOneQuantifier}_{pna,Cooper}$ da eliminišemo promenljivu min. Dobijamo

$$\forall max\ \forall\alpha(0 \leq 0)\ ,$$

i nakon zamenjivanja max sa $max(\alpha)$ dobijamo $\forall\alpha\ (0 \leq 0)$.

(1d) Može se eliminisati (redudantan) kvantifikator $\forall\alpha$ jer se α ne pojavljuje u $0 \leq 0$ (korišćenjem `ElimRedundantQuantifier`) i dobijamo $0 \leq 0$.

(2d) $0 \leq 0$ je PNA-formula, pa možemo upotrebiti proceduru $\texttt{Dp}_{pna,Cooper}$ koja vraća $\top$. Dakle, dato tvrđenje je teorema.

4.5 Svojstva EPM sheme

Zaustavljanje

Teorema 4.1 *Svaka procedura koja je instanca* EPM *sheme se zaustavlja.*

Dokaz. Svaka od procedura pojednostavljivanja vraća ili formulu sa manje kvantifikatora nego ulazna formula ili je broj kvantifikatora isti, ali je najteži ne-$\mathcal{T}$-term u ulaznoj formuli teži nego najteži ne-$\mathcal{T}$-term u rezultujućoj formuli. Nijedna od procedura pojednostavljivanja ne uvodi nove promenljive. Kako ima konačno mnogo promenljivih u formuli koja se dokazuje i kako za svaki ne-$\mathcal{T}$-term postoji konačno mnogo ne-$\mathcal{T}$-termova nad tim skupom od kojih je on teži, procedure pojednostavljivanja mogu biti primenjene samo konačan broj puta (za konačan skup raspoloživih lema[8]). Dakle, svaka procedura koja je instanca EPM sheme se zaustavlja.

Saglasnost

Teorema 4.2 *Svaka procedura koja je instanca* EPM *sheme je saglasna (tj. ako za zadato tvrđenje F instanca* EPM *sheme za teoriju $\mathcal{T}$ vrati odgovor da je F teorema, onda važi $\mathcal{T}^e \vdash F$, gde je $\mathcal{T}^e$ proširenje teorije $\mathcal{T}$).*

Dokaz. Pretpostavljamo da su raspoložive procedure $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ i $\texttt{DpGround}_{\mathcal{T}}$ potpune i saglasne. Dodatno, ako je pokazano da je formula sa ne-$\mathcal{T}$-termovima generalizovanim sa promenljivama teorema, onda je i ulazna formula teorema. Procedura `ElimRedundantQuantifier` je očito saglasna. Dakle, procedure $\texttt{Dp}_{\mathcal{T},A}$, `ElimRedundantQuantifier` i $\texttt{ElimOneVar}_{\mathcal{T},A}$ su saglasne.

[8] U instancama EPM sheme moguć je bektreking, ali za konačan broj raspoloživih lema, procedura $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ (koja jedina može da se iznova primenjuje u istoj tački) može biti primenjena samo na konačno mnogo načina.

Dokažimo da je saglasna i procedura $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$. Neka je $(\forall\vec{x})F$ formula koja se dokazuje, neka su $(\forall\vec{u})L$ lema i ϕ supstitucija koje zadobvoljavaju uslove procedure $\texttt{ElimGeneralisedTerm}_{\mathcal{T},A}$ (razmatramo samo slučaj kada je samo jedna lema iskorišćena; u opštem slučaju dokaz je analogan). Iz $\mathcal{T}^e \vdash (\forall\vec{u})L$ sledi[9] $\mathcal{T}^e \vdash (\forall\vec{x})L\phi$. Pretpostavimo da smo dokazali $\forall\vec{x}(F \vee \neg L\phi)$. Onda je $\mathcal{T}^e \vdash (\forall\vec{x})L\phi$ i $\mathcal{T}^e \vdash \forall\vec{x}(F \vee \neg L\phi)$ implicira $\mathcal{T}^e \vdash \forall\vec{x}(L\phi \wedge (F \vee \neg L\phi))$ i, dalje, $\mathcal{T}^e \vdash \forall\vec{x}(L\phi \wedge F)$. Konačno, važi $\mathcal{T}^e \vdash \forall\vec{x}F$, tj. formula $(\forall\vec{x})F$ je teorema, što je i trebalo dokazati.

(Za dokaz saglasnost instanci EPM sheme dovoljno je dokazati da su procedure $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$ i $\texttt{DpGround}_{\mathcal{T}}$ saglasne (tj. nije nužno da one budu potpune). Na primer, Hodesov algoritam može biti upotrebljen za proceduru za PNA koja bi bila saglasna (ali nepotpuna) za univerzalno zatvorene formule.)

(Ne)potpunost Za instance EPM sheme ne može se garantovati potpunost; to se, međutim, ne može smatrati nedostatkom ove procedure jer je ona zamišljena za korišćenje u neodlučivim teorijama — ona predstavlja pokušaj da se izgradi metod koji je primenljiv i uspešan i izvan dosega procedure odlučivanja (u neodlučivim fragmentima).

U određenom smislu, svaka instanca EPM sheme jeste potpuna jer je ona strogo proširenje bazične procedure odlučivanja A za teoriju $\mathcal{T}$: sve formule koje pripadaju dosegu te procedure biće razrešene korektno (jer je $\mathcal{T}^e$ konzervativno proširenje teorije $\mathcal{T}$). Dakle, ako formula pripada jeziku teorije $\mathcal{T}$ biće dokazano da ona jeste ili nije teorema te teorije (i teorije $\mathcal{T}^e$). Ako formula F pripada jeziku teorije $\mathcal{T}^e$, ali ne i teorije $\mathcal{T}$, onda u slučaju $\mathcal{T}^e \vdash F$, procedura EPM može da je dokaže (ali ne uvek; videti [124] i primer 5.10), a u slučaju $\mathcal{T}^e \nvdash F$, procedura EPM zaustavlja rad neuspešno.

Efikasnost U EPM shemi postoje određena nepotrebna ponavljanja nekih koraka što proističe iz fleksibilnog kombinovanja nezavisnih modula. Međutim, ovi koraci (generalizacija ne-$\mathcal{T}$-termova, supstitucije, određene normalizacije itd) mogu obično biti izvršeni u linearnom vremenu u odnosu na veličinu tekućeg tvrđenja i stoga ne utiču značajno na ukupnu efikasnost sistema. Štaviše, obično svi koraci EPM sheme (oni koji ne zavise od specifične teorije) mogu biti izvršeni u linearnom vremenu u odnosu na veličinu formule koja se dokazuje. Dakle, složenošću instanci EPM sheme dominira složenost procedura $\texttt{DpGround}_{\mathcal{T}}$ i $\texttt{DpElimOneQuantifier}_{\mathcal{T},A}$.

Iako su neki uslovi u EPM shemi relativno komplikovani, ona je generalno namenjena jednostavnim tvrđenjima (koja se pojavljuju u tipičnim problemima verifikacije softvera), pa bi njeno izvršavanje bilo jednostavno i brzo. Pored toga, primenu EPM sheme možemo svesti samo na tvrđenja za koje je izgledno da će onda biti uspešna (za ovu restrikciju možemo da koristimo pogodne heurističke ili stohastičke skorove [78]), a koristiti druge tehnike (npr. indukciju) u drugim slučajevima.

[9] Pretpostavljamo da je (nekim metodom) dokazano da je $(\forall\vec{u})L$ teorema teorije $\mathcal{T}^e$.

Fleksibilnost Kao i obično, postoji balans između opštosti i efikasnosti u izgradnji proširenja procedure odlučivanja. Shema EPM je izgrađena kao kombinacija nezavisnih apstraktnih modula što smanjuje njenu efikasnost. Na drugoj strani, shema EPM ima veliki stepen opštosti i fleksibilnosti: ona može na jedinstven način biti korišćena u različitim dokazivačima teorema, za različite teorije i za različite bazične procedure odlučivanja. Dodatno, shema EPM ne zahteva nikakve specifične strukture podataka (npr. baze polinoma ili baze ograničenja) niti neke dodatne eksterne mehanizme (npr. eksterno uređenje nad termovima). U svakom trenutku tekuća formula može biti prepuštena nekoj drugoj dokazivačkoj strategiji. Verujemo da ove prednosti dominiraju gubicima u efikasnosti.

4.6 Moguća unapređenja

Predložena EPM shema može biti uopštena na sledeće načine:

- tako da pokriva i predikate iz skupa $\Pi^e \setminus \Pi$ (kada on nije prazan); to može biti postignuto korišćenjem lema i generalizovanjem atomičkih formula čiji je dominantan predikatski simbol iz $\Pi^e \setminus \Pi$;
- korišćenjem aksioma supstitucije ne samo kao lema (već kao ugrađenih pravila prezapisivanja) ili korišćenjem nekog efikasnijeg mehanizma za jednakosno rezonovanje (npr. algoritma za kongruentno zatvorenje);
- proširivanjem domena tako da se ne odnosi samo na univerzalne formule;
- korišćenjem definicija ili lema o $\mathcal{T}^e$ funkcijama i predikatima iskazanim u terminima teorije $\mathcal{T}$ (npr. uvek je moguće prezapisati $double(x)$ u $2x$ korišćenjem definicije $\forall x\ (double(x) =_{\mathbf{N}} 2x)$ i uvek je moguće prezapisati $x \neq y$ u $x < y \vee y < x$ korišćenjem leme $\forall x\ \forall y\ (x \neq y \leftrightarrow x < y \vee y < x)$);

Efikasnost instanci EPM sheme može biti popravljena na sledeće načine:

- pojačavanjem uslova (i) u 4.2.4 — tako da se pokušava unifikacija čitave atomičke formule f sa l (a ne samo unifikacija najtežih ne-$\mathcal{T}$-termova); ovaj pristup može biti efikasniji u nekim tvrđenjima, ali smanjuje opseg sheme: sa ovakvim ograničenjem, na primer, nije moguće dokazati tvrđenje $\forall\alpha\forall k\ min(\alpha) \leq max(\alpha) + k$ korišćenjem leme $\forall\xi(min(x) \leq max(x))$;
- korišćenjem samo lema određene strukture (npr. lema oblika $\forall\vec{x}(H_1 \wedge \cdots \wedge H_k \rightarrow C)$);
- korišćenjem uređenja svođenja nad termovima i preprocesiranjem svih lema.

Između navedenih unapređenja u smeru uopštavanja sheme ili u smeru povećavanja efikasnosti postoji balans i pojedina rešenja mogu biti primenjena u zavisnosti od konteksta.

4.7 Poređenje sa drugim integracionim shemama

EPM shema, kao i druge integracione sheme, je prvenstveno usmerena ka primeni u neodlučivim proširenjima odlučivih teorija, dok su kombinacione sheme prvenstveno usmerene ka primeni u odlučivim unijama odlučivih teorija. Zbog ove razlike, u ovom poglavlju poredimo EPM shemu samo sa sličnim, integracionim shemama (a ne i sa kombinacionim shemama). Poredimo EPM shemu sa Boyer/Mooreovom procedurom za linearnu aritmetiku (NQTHM [17]), Kapur/Nieovom TECTON [68]) i Armando/Raniseovom (CCR [4]) shemom. Sve one usmerene su ka neodlučivim proširenjima odlučivih teorija, koriste dodatne leme za pokrivanje tih proširenja i odnose se samo na univerzalno kvantifikovana tvrđenja.

Nivo apstrakcije Boyer/Mooreova procedura za linearnu aritmetiku ugrađena u NQTHM je opisana samo za PIA kao osnovnu teoriju i za Hodesovu proceduru kao odgovarajuću proceduru odlučivanja; ona ne može biti prilagođena za neku drugu teoriju ili neku drugu proceduru odlučivanja. Dodatno, opis sheme je dat najčešće u terminima korišćenih struktura podataka (često pomešanih sa opisima različitih optimizacija) a ne u terminima njihovog logičkog značenja. Sve ovo čini ovu shemu i sve njene detalje veoma teškim za potpuno razumevanje, za dosledno implementiranje i za formalno rezonovanje. Kapur/Nieova shema ugrađena u sistem TECTON je opisana na znatno apstraktniji način; ona može biti veoma dosledno implementirana i, u principu, ona može biti prilagođena za druge osnovne teorije ili druge osnovne procedure odlučivanja. Ipak, u opisu ove sheme postoje i detalji koji nisu potpuno opisani (kao što je simplifikacija Prezburgerovih atomičkih formula; niti je ona precizno opisana niti je njena uloga sasvim jasna (da li je ona neophodna ili ne)). Ovi slabo opisani detalji otežavaju formalno rezonovanje i u originalnom radu dokazi zaustavljanja, saglasnosti i potpunosti (za Prezburgerovu aritmetiku sa čistom teorijom jednakosti) samo su skicirani. CCR shema je opisana na čisto formalan i apstraktan način. Ona može biti instancirana na različite integracione metode (uključujući Boyer/Mooreov i Kapur/Nieov). Nivo apstrakcije u opisu ove sheme je suprotnost Boyer/Mooreovoj shemi; naime, opis CCR je potpuno apstraktan, zasnovan na novoj i prilično komplikovanoj notaciji i često veoma težak za razumevanje. EPM shema je formalno opisana, ali u terminima samo nekoliko primitivnih procedura (kao što je eliminacija promenljivih). Notacija je jednostavna i shema može biti dosledno implementirana na osnovu opisa. EPM shema je opisana na apstraktan način i može biti instancirana bilo kojom teorijom koja dopušta eliminaciju kvantifikatora.

Osnovna teorija Kao što je već rečeno, Boyer/Mooreova shema je razvijena samo za PIA i Hodesova procedura za PRA (kao saglasna, iako nepotpuna procedura za PIA) je kruto ugrađena u nju i ne može biti zamenjena nekom drugom. TECTON shema je opisana za PIA, PNA i PNA; procedure odlučivanja za ove teorije su zasnovane na Fourierovom metodu, ali mogu biti zamenjene i

nekim drugim metodama; dodatno, shema, u principu, može biti formulisana i za neke druge teorije po analogiji. CCR shema može biti instancirana za bilo koju teoriju za koju postoji ograničen skup (slabo specifikovanih) dokazivačkih specijalista. EPM shema može biti instancirana bilo kojom odlučivom teorijom $\mathcal{T}$ koja dopušta eliminaciju kvantifikatora i bilo kojom procedurom za eliminaciju kvantifikatora A za tu teoriju (npr. Hodesovom procedurom za PRA, Cooperovom proceduroma za PIA). Bilo koja procedura za eliminaciju kvantifikatora može da služi kao osnova za dokazivačke specijaliste potrebne za CCR shemu, pa je što se tiče osnovne teorije, EPM shema opštija od NQTHM i od (osnovne verzije) sheme TECTON, ali manje opšta od sheme CCR.

Jednakosno rezonovanje TECTON koristi kongruentno zatvorenje, odnosno bazno upotpunjavanje kao snažnu podršku jednakosnom rezonovanju. CCR takođe može da koristi ovaj oblik rezonovanja. Boyer/Mooreova i EPM shema imaju slabo jednakosno rezonovanje. U Boyer/Mooreovoj shemi neki oblici jednakosnog rezonovanja podržani su konkretnim pravilima pojednostavljivanja. U EPM shemi aksiome supstitucije mogu da se koriste kao leme.

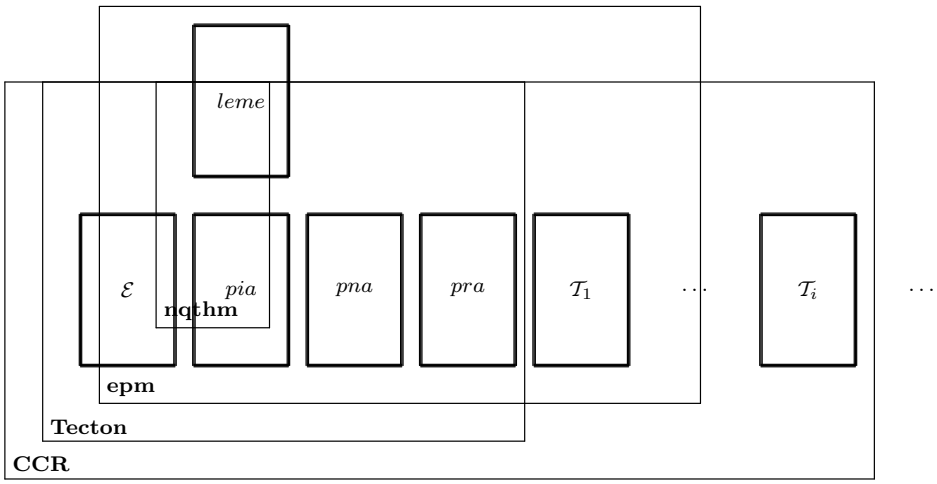
Struktura lema Sheme NQTHM, TECTON i CCR koriste samo leme oblika $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$ (gde su p_i i ρ literali), dok se EPM shema odnosi na tvrđenja i leme proizvoljne strukture; navedene prve tri sheme koriste fiksnu strukturu lema i daju dokaze sa strukturom stabla, dok su dokazi u EPM shemi linearni; ovo čini prve tri sheme efikasnijim, a EPM shemu opštijom.

Doseg Što se tiče strukture lema, EPM shema je opštija od NQTHM, TECTON i CCR; što se tiče osnovne teorije, EPM shema je opštija od sheme NQTHM i od (osnovne verzije) sheme TECTON, ali manje opšta od CCR; što se tiče jednakosnog rezonovanja, Boyer/Mooreova i EPM shema su slabije od shema TECTON i CCR. Ovi dosezi ilustrovani su na slici 4.1.

Naglasimo da su ove sheme *usmerene* ka nekim domenima, ali se od njih ne očekuje da budu *potpune* u tim domenima. Na primer, čak iako je neko tvrđenje logička posledica teorije PRA i nekih raspoloživih lema, ono ne mora biti dokazivo ovim shemama (o ovom problemu diskutuje se u radu [124]).

Uređenje nad termovima Kao i Boyer/Mooreova shema, EPM shema koristi uređenje nad termovima bazirano na veličini i alfabetskom uređenju, dok TECTON i CCR koriste sofisticiranija uređenja svođenja (kao što je uređenje rekurzivne staze). S druge strane, EPM shema nije ograničena na uređenje zasnovano na veličini i ona može biti prilagođena nekom drugom uređenju (što bi zahtevalo i određene izmene u mehanizmu za pozivanje lema).

Zaustavljanje Za Boyer/Mooreovu shemu zaustavljanje nije dokazano i verovatno objašnjenje je u njenom krajnje neformalnom opisu nepodesnom za formalno rezonovanje. Za shemu TECTON zaustavljanje je dokazano neformalno [68]. Zaustavljanje sheme CCR je dokazano formalno za apstraktnu shemu



Slika 4.1: Dosezi shema NQTHM, TECTON, CCR i EPM ($\mathcal{E}$ označava čistu teoriju jednakosti)

kada su ispunjeni određeni zahtevi vezani za prezapisivački mehanizam i procedure odlučivanja [4]. Zaustavljanje EPM sheme je dokazano formalno za apstraktnu shemu.

Saglasnost Za Boyer/Mooreovu i TECTON shemu saglasnost je diskutovana i dokazana samo neformalno. Za CCR i EPM saglasnost je dokazana formalno za apstraktne sheme.

Potpunost Boyer/Mooreova shema koristi Hodesovu proceduru za PIA, pa ona nije potpuna čak ni za PIA (s druge strane, s obzirom da je ona optimizovana za PIA, ona nije potpuna ni za PRA). Shema TECTON je potpuna za PIA (ili PNA, PRA) sa čistom teorijom jednakosti. U shemi CCR, dokazivački specijalisti su preslabo specifikovani i, za trenutnu verziju, nemoguće je dokazati potpunost za osnovnu teoriju (ili za osnovnu teoriju sa čistom teorijom jednakosti); ova slaba specifikacija čini CCR veoma opštom shemom, ali u isto vreme onemogućava dokazivanje bilo koje forme potpunosti (i čini se da je najslavija tačka sheme CCR). Za EPM shemu potpunost za osnovnu teoriju može biti dokazana (tj. ona je strogo proširenje osnovne procedure odlučivanja za teoriju koja dopušta eliminaciju kvantifikatora).

Sve od navedenih shema su usmerene ka neodlučivim proširenjima odlučivih teorija i za nijednu od njih potpunost za njihov doseg ne može biti dokazana.

Dodatni mehanizmi i strukture podataka Za razliku od drugih opisanih integracionih shema, shema EPM ne koristi nikakve posebne strukture podataka, baze podataka, baze ograničenja i sl. Dodatno, ona koristi ugrađeno uređenje nad termovima umesto nekog eksternog uređenja. Zbog toga, u tekućem cilju je sadržana potpuna informacija o stanju dokaza i tekući cilj može biti prosleđen nekoj drugoj dokazivačkoj strategiji nakon bilo kojeg koraka procedure.

Efikasnost Zbog složenih osnovnih oblika rezonovanja, teško je izračunati ukupnu složenost opisanih integracionih shema (veoma je teško izračunati i složenost izolovane Hodesove, Cooperove ili Fourierova eliminacije promenljivih). S druge strane, kako bi bili dobijeni relevantni rezultati uporedne eksperimentalne analize, potrebno je ne samo da sve sheme budu raspoložive na istoj platformi, već i da one dele što je moguće više kôda (npr. Hodesovu eliminaciju promenljivih). Opšti okvir za kombinovanje i integrisanje procedura odlučivanja koji se predlaže u narednom delu pruža mogućnosti za takvo relevantno poređenje i deo 6 sadrži neke eksperimentalne rezultate poređenja varijanti TECTON i EPM sheme (ti rezultati potvrđuju da shema TECTON ima širi doseg zasnovan na jednakosnom rezonovanju, ali i da EPM shema može biti efikasnija za tvrđenje u kojima jednakosno rezonovanje nije suštinsko).

Generalno, linearni dokazi i uređenje nad termovima (koje kontroliše mehanizam lema) mogu da čine EPM shemu manje efikasnom od navedenih shema; s druge strane, iako slabo jednakosno rezonovanje sužava doseg EPM sheme ono je čini efikasnijom, nego što je, na primer, TECTON za tvrđenja u kojima jednakosno rezonovanje nije potrebno ili nije korisno.

Zaključci: prednosti EPM sheme EPM shema koristi leme proizvoljne strukture i, dakle, što se tiče lema, ima širi doseg nego druge sheme. EPM shema je izgrađena na fleksibilan način, od nezavisnih modula i ona može biti instancirana za različite teorije, za različite procedure odlučivanja za te teorije i može da koristi različite uređenja nad termovima. U EPM shemi ne koriste se nikakve dodatne strukture podataka, uvek postoji samo jedan tekući cilj i on može biti prosleđen nekoj drugoj dokazivačkoj strategiji nakon bilo kojeg koraka procedure. Verujemo da shema EPM predstavlja dobar balans između veoma neformalnih (kao što je Boyer/Mooreov) i veoma formalnih (kao što je CCR) pristupa: ona koristi jednostavnu notaciju, jasan opis primitivnih procedura i formulisana je na precizan i kratak način koji omogućava formalno rezonovanje o njenim svojstvima (pa saglasnost i zaustavljanje mogu da budu dokazani u opštem slučaju). EPM shema pokriva širok skup teorija i, za razliku od ostalih pristupa, postoji jasna karakterizacija klase teorija za koje je ona strogo proširenje odguvarajućih procedura odlučivanja. EPM shema je potpuna za sve te teorije i to može biti lako dokazano u opštem slučaju (dok je dokazivanje bilo koje vrste potpunosti u sistemu CCR veoma teško ako ne i nemoguće).

Sva navedena poređenja između EPM sheme i ostalih pristupa uticala su na opšti okvir opisan u narednom delu i na izbor specifičnih makro pravila

izvođenja. EPM shema opisana u tom opštem okviru zadržava većinu prednosti originalne verzije i koristi neke od prednosti rivalskih tehnika.

4.8 Flexible Extension of Decision Procedures: Summary

The scheme EPM for the flexible integration of a certain class of decision procedures into theorem provers is partly based on Boyer/Moore's linear arithmetic procedure [17], but is more general, abstract and flexible. This scheme provides a flexible framework for extending a realm of a decision procedure A for a theory $\mathcal{T}$ (say, for Presburger arithmetic) by the use of available lemmas. The scheme can be used in different theorem provers, for different theories and for different decision procedures for these theories. Specific knowledge is encapsulated in smaller submodules and decision procedures can be simply 'plugged-in' to the system. We assume that the theory $\mathcal{T}$ we deal with admits quantifier elimination. EPM scheme is based on the following primitive procedures ($\mathcal{T}^e$ is some extension of the theory $\mathcal{T}$ and A is some quantifier elimination algorithm for the theory $\mathcal{T}$, e.g. Cooper's algorithm for Presburger arithmetic):

theory specific procedures:

- `DpElimOneQuantifier`$_{\mathcal{T},A}$**:** reduces the number of quantifiers in the $\mathcal{T}$–formula being proved;
- `DpGround`$_{\mathcal{T}}$**:** a decision procedure for ground $\mathcal{T}$-formulae;
- `Dp`$_{\mathcal{T},A}$**:** a decision procedure for a theory $\mathcal{T}$; induced by the procedures `DpElimOneQuantifier`$_{\mathcal{T},A}$ and `DpGround`$_{\mathcal{T}}$;

general procedures:

- `ElimRedundantQuantifier`**:** eliminates a redundant quantifier;
- `ElimOneVar`$_{\mathcal{T},A}$**:** eliminates a variable which does not appear in non-$\mathcal{T}$-terms of the $\mathcal{T}^e$ formula being proved;
- `ElimGeneralisedTerm`$_{\mathcal{T},A}$**:** eliminates the heaviest non-$\mathcal{T}$-term (i.e. the maximal alien term) by using generalisation and lemmas.

The simplified extended proof method is as follows:

(1) try to apply `ElimRedundantQuantifier` and go to (1);

(2) try to apply `Dp`$_{\mathcal{T},A}$;

(3) try to apply `ElimOneVar`$_{\mathcal{T},A}$ and go to (1);

(4) apply `ElimGeneralisedTerm`$_{\mathcal{T},A}$ and go to (1).

The EPM scheme uses lemmas of arbitrary structure and therefore, concerning lemmas, has a wider scope than rival integration schemes. The EPM scheme is built in a flexible manner, of independent modules, and it can be instantiated for different theories and for different decision procedures for these theories (if they provide needed functionalities) and it can use different term orderings. We believe that the EPM scheme is a good balance between highly informal (like Boyer/Moore's) and highly formal (like CCR) approaches: it uses simple notation and clear description of primitive procedures and it is formulated in a precise and short way which enables formal reasoning about its properties (so soundness and termination can be proved in a general way). It covers a wide scope of theories and, unlike in the other schemes, there is a clear characterisation of a class of theories for which it is a strict extension of corresponding decision procedures. The EPM scheme is complete for all these theories and that can be easily proved in a general manner (while proving any kind of completeness in CCR is very difficult if not impossible).

The EPM scheme is well-suited to the proof-planning paradigm. We have made an implementation of the EPM scheme (and some of its instances) within the *Clam* system which is described in chapter 6.

Glava 5

Opšti okvir za kombinovanje i integrisanje procedura odlučivanja

U ovom delu teze dajemo opis opšteg okvira za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema. Opšti okvir zasnovan je na makro pravilima izvođenja koja su motivisana izvođenjima primenjivanim u najznačajnijim pristupima koji se bave ovim problemima. Neka od njih su apstrakcija, kongruentno zatvorenje, korišćenje lema itd. Korišćenjem tih pravila moguće je opisati raznorodne sheme i za kombinovanje i za integrisanje procedura odlučivanja u dokazivače teorema, uključujući Nelson/Oppenovu, Shostakovu i shemu koja se koristi u sistemu TECTON. Ti opisi ne sadrže sve aspekte i detalje pomenutih pristupa, ali sadrže njihove ključne ideje i imaju njihove domene. Opšti okvir pruža mogućnosti za opisivanje jedne sheme za kombinovanje/integrisanje procedura odlučivanja i čini različite sheme uporedivim. Sva makro pravila su saglasna i, prema tome, sve sheme implementirane na osnovu njih su saglasne. Sva makro pravila izvođenja se zaustavljaju, ali zaustavljanje pojedinačnih shema mora da bude dokazano posebno. Za neke od implementiranih shema može se dokazati da su potpune (npr. za Nelson/Oppenovu). Sheme implementirane primenom predloženog opšteg okvira su najčešće manje efikasne od izvornih procedura (u kojima su često primenjene specifične optimizacije), ali verujemo da dobici na osnovu fleksibilnosti i formalnog opisa prevazilaze te gubitke u efikasnosti. Predloženi opšti okvir i neke sheme implementirani su i opis te implementacije sa rezultatima dat je u delu 6.

Neke od definicija i pojmova uvedenih u delu 2 biće ponovljene u nastavku, u novom kontekstu pojedinih makro pravila izvođenja.

5.1 Makro pravila izvođenja

Postoji nekoliko ključnih koraka u različitim sistemima za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema, uključujući apstrakciju, kongruentno zatvorenje, korišćenje lema. Umesto da se usredsredimo na specifične instance ovih koraka (što je najčešći slučaj u opisima sistema koji se odnose na ovu problematiku), daćemo opšti okvir za kombinovanje različitih instanci ovih koraka. Njihov opis daćemo u formi makro pravila izvođenja.

Koristimo dokazivanje pobijanjem, pa, ako je F formula bez kvantifikatora koju dokazujemo, treba da pokažemo da je $\neg F$ nezadovoljivo u $\mathcal{T}$, tj. treba da pokažemo da je $\mathcal{T} \cup \neg F$ protivrečno. U daljem tekstu, smatraćemo da je teorija $\mathcal{T}$ sa kojom radimo fiksirana u određenom kontekstu i pod "$\neg F$ je nezadovoljivo" mislićemo "$\neg F$ je nezadovoljivo u $\mathcal{T}$" itd.

Makro pravila izvođenja označavamo sa $f_1 \Rightarrow_X f_2$, gde je X ime određenog pravila. Ako je pravilo parametrizovano parametrima P, onda njegovo ime označavamo sa $X(P)$. Saglasnost svakog makro pravila izvođenja mora da bude osigurana; tj. za sva pravila $f_1 \Rightarrow_X f_2$, mora da važi sledeće: ako je f_2 nezadovoljivo, onda je i f_1 nezadovoljivo. Ako pravilo izvođenja X čuva (ne)zadovoljivost, onda ga označavamo $\Leftrightarrow_X$. Niz $f_1 \Rightarrow_{X_{i_1}} f_2$, $f_2 \Rightarrow_{X_{i_2}} f_3$, ..., $f_{n-1} \Rightarrow_{X_{i_{n-1}}} f_n$ označavamo kraće $f_1 \Rightarrow^* f_n$. Niz $f_1 \Leftrightarrow_{X_{i_1}} f_2$, $f_2 \Leftrightarrow_{X_{i_2}} f_3$, ..., $f_{n-1} \Leftrightarrow_{X_{i_{n-1}}} f_n$ označavamo kraće $f_1 \Leftrightarrow^* f_n$. Ako važi $\neg F \Rightarrow^* \bot$ onda je $\neg F$ nezadovoljivo, pa je polazna formula F teorema. Ako važi $\neg F \Leftrightarrow^* \top$ onda postoji implicitno konstruisan kontramodel za F, pa ona nije teorema. U heurističkom dokazivaču teorema, negativni rezultati takođe mogu biti korisni (na primer, u kontrolisanju generalizacije i drugih heuristika koje ne čuvaju zadovoljivost). Ako važi $f \Leftrightarrow^* f'$ gde je $f' \in \{\top, \bot\}$ onda kažemo da je ta grana dokaza zatvorene i f' se vraća kao rezultat procesa izvođenja. Dodatno, nakon primene bilo kojeg pravila izvođenja, ako važi $f \Leftrightarrow^* f'$, formula f' može biti prepuštena nekoj drugoj komponenti dokazivača i neka druga dokazivačka strategija (npr. indukcija) može da bude primenjena.

Pretpostavljamo da mogu da budu raspoloživa neka dodatna pravila prezapisivanja, definicijske jednakosti i leme. Razmatraćemo uslovna pravila prezapisivanja (tj. pravila oblika $l \to r$ **if** $p_1, p_2, \ldots, p_k$ i oblika $\rho \to \top$ **if** $p_1, p_2, \ldots, p_k$). Razmatraćemo samo definicijske jednakosti i leme bez kvantifikatora koje su oblika $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow l =_s r$ i $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$ i, stoga, tretiraćemo ih na isti način kao i uslovna pravila prezapisivanja.

Pretpostavljamo da je raspoloživo uređenje nad funkcijskim i predikatskim simbolima koje indukuje uređenje svođenja $\prec$ nad termovima.

Jednostavnosti radi, u nastavku teksta, umesto simbola jednakosti sa oznakom sorte, pisaćemo samo $=$ kada je sorta jasna iz konteksta ili kada ona nije važna.

5.1.1 Disjunktivna normalna forma

Ako je f negacija formule F koju treba dokazati, možemo da je transformišemo u disjunktivnu normalnu formu i da pokušamo da pobijemo svaki od disjunkata.

Ovu transformaciju[1] označavamo na sledeći način:

$$f \Rightarrow_{dnf} (f_1 \vee f_2 \vee \ldots \vee f_n)$$

Kako transformisanje u disjunktivnu normalnu formu čuva (ne)zadovoljivost, označavamo ga i na sledeći način:

$$f \Leftrightarrow_{dnf} (f_1 \vee f_2 \vee \ldots \vee f_n)$$

5.1.2 Dokazivanje disjunkata

Ako je formula f oblika $(f_1 \vee f_2 \vee \ldots \vee f_n)$ (gde je svaka od formula f_i konjunkcija literala), da bismo pobili f, treba da pobijemo svaki od f_i. Dakle,

$$f \Leftrightarrow_{disj} \bot \text{ ako } f_i \Rightarrow^* \bot \text{ za sve } i,\ 1 \leq i \leq n$$

Dodatno, važi:

$$f \Leftrightarrow_{disj} \top \text{ ako } f_i \Leftrightarrow^* \top \text{ za neko } i,\ 1 \leq i \leq n$$

U nastavku ćemo uglavnom raditi sa konjunkcijama literala (tj. sa konjunkcijama atomičkih formula i negacijama atomičkih formula). Nećemo praviti razliku između konjunkcije $l_1 \wedge l_2 \wedge \ldots \wedge l_n$ i mulitskupa $\{l_1, l_2, \ldots, l_n\}$.

5.1.3 Pojednostavljivanje

Pojednostavljivanje je zasnovano na sledećim pravilima:

$$\begin{array}{rcl}
f \cup \{l\} \cup \{l\} & \longrightarrow & f \cup \{l\} \\
f \cup \{l\} \cup \{\neg l\} & \longrightarrow & \bot \\
f \cup \{\bot\} & \longrightarrow & \bot \\
f \cup \{\top\} & \longrightarrow & f \\
f \cup \{\neg(a = a)\} & \longrightarrow & \bot \\
f \cup \{a = a\} & \longrightarrow & f \\
f \cup \{x = y\} & \longrightarrow & f
\end{array}$$

gde je x promenljiva koja se ne pojavljuje u y i f, ili je y promenljiva koja se ne pojavljuje u x i f.

Izvođenje koje iscrpno primenjuje navedena pravila (nad formulom f i dajući formulu f') označavamo sa

$$f \Leftrightarrow_{simpl} f'$$

Dodatno, postoje pojednostavljivanja specifična za pojedine teorije. Na primer, literal $x \leq (y + z) - (y - x)$ treba da bude prezapisan u literal $0 \leq z$ pojednostavljivanjem za PRA. Takođe, pojednostavljivač za teoriju $\mathcal{T}_i$ treba da

[1] Transformisanje u disjunktivnu normalnu formu može se realizovati korišćenjem zaustavljajućeg sistema za prezapisivanje. Međutim, ne postoji odgovarajući kanonski sistem za prezapisivanje.

detektuje valjane i protivrečne bazne $\mathcal{T}_i$ literale i da ih prezapisuje u $\top$ i $\bot$, redom. Pojednostavljivanje za pojedine teorije $\mathcal{T}_i$ se primenjuje samo pojedinačno na $\mathcal{T}_i$ literale. Radeći sa kombinacijama teorija $\mathcal{T}_i$ $(i = 1, 2, \ldots, k)$, iterativno primenjujemo pojednostavljivače za teorije $\mathcal{T}_i$ $(i = 1, 2, \ldots, k)$ i označavamo ovo pojednostavljivanje (prošireni oblik pravila $\Leftrightarrow_{simpl}$) sa

$$f \Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f'$$

Pretpostavljamo da $\Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f'$ ne uvodi nove promenljive ili nove strane termove u odnosu na $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$. Takođe, pretpostavljamo da iz veza $f \Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f'$ i $f' \Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f''$ sledi da su f' i f'' identične formule.

5.1.4 Apstrahovanje

U zavisnosti od dominantnog funkcijskog ili predikatskog simbola, možemo uslovno da smatramo da neka atomička formula pripada teoriji $\mathcal{T}_i$ i da apstrahujemo njene strane termove (novim, *apstrakcijskim promenljivama*) kako bismo dobili atomičku formulu teorije $\mathcal{T}_i$. Na primer, atomička formula $x \leq y + f(x + y)$ može biti apstrahovana u atomičku formulu Prezburgerove aritmetike $x \leq y + t$, gde je $t = f(x + y)$.

Apstrahovanje možemo da primenimo na više teorija istovremeno $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_k$ u jednom koraku (tako da svaki od apstrahovanih literala pripada jednoj od teorija $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_k$). Originalne literale možemo zameniti odgovarajućim apstrakcijskim promenljivama (što ne čuva nezadovoljivost):

$$f \Rightarrow_{abs(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f'$$

Novodobijenoj formuli možemo dodati skup C jednakosti koje definišu uvedene apstrakcijske promenljive ($\{t = f(x + y)\}$ u navedenom primeru), što označavamo sa:

$$f \Rightarrow_{abs^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \cup C$$

ili, alternativno (jer pravilo čuva nezadovoljivost), sa

$$f \Leftrightarrow_{abs^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \cup C$$

Apstrahovanje može biti propagirano ako podtermovi apstrakcijskih termova sadrže strane termove u odnosu na $\mathcal{T}_1$, $\mathcal{T}_2$,...,$\mathcal{T}_k$. Na primer, nakon propagiranja apstrahovanja (u odnosu na PNA i na čistu teoriju jednakosti) $x \leq y + f(x + y)$ postaje $x \leq y + t_1 \wedge t_1 = f(t_2) \wedge t_2 = x + y$. Ovu forma apstrahovanja (propagirano apstrahovanje) označavamo sa *absp* i ono uvek dodaje jednakosti c koje definišu novouvedene promenljive. Odgovarajuće izvođenje označavamo sa

$$f \Rightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \cup C$$

ili, alternativno (jer se čuva nezadovoljivost), sa

$$f \Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \cup C$$

5.1.5 Zamena

Ako je tekuća formula f konjunkcija literala i ako postoji literal $x = t$, gde je x promenljiva i t je term koji ne sadrži x, onda možemo zameniti sva pojavljivanja promenljive x u f termom t i izbrisati literal $x = t$. Ovu operaciju možemo izvršiti za sve promenljive x koje zadovoljavaju navedeni uslov i to u iteracijama za po jednu promenljivu[2] dajući formulu f'. Ovo izvođenje označavamo sa:

$$f \Rightarrow_{repl} f'$$

ili, alternativno, sa

$$f \Leftrightarrow_{repl} f'$$

Ako f ne sadrži literale koji zadovoljavaju navedene uslove, primetimo da važi:

$$f \Leftrightarrow_{abs^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \Leftrightarrow_{repl} f$$

i

$$f \Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)} f' \Leftrightarrow_{repl} f$$

Navedeni primer ilustruje činjenicu da treba biti obazriv sa primenom pravila zamene jer ono može da poništi efekte primene pravila apstrahovanja.

Druga varijanta pravila zamene je svedena samo na zamenu promenljivih promenenljivim (tj. samo na slučaj kada postoje jednakosti oblika $x = y$); takva zamena eliminiše jednu promenljivu i ne uvodi nove strane termove. Označavamo je sa

$$f \Rightarrow_{repl=} f'$$

ili, alternativno, sa

$$f \Leftrightarrow_{repl=} f'$$

5.1.6 Nezadovoljivost

Pretpostavimo da je raspoloživa procedura koja može da detektuje da li je konjunkcija nekih literala $\{l_{i_1}, l_{i_2}, \ldots, l_{i_k}\}$ $(i_k < n)$ iz formule f nezadovoljiva u teoriji $\mathcal{T}_i$, gde je f konjunkcija literala $l_1, l_2, \ldots, l_n$. Ako je konjunkcija $\{l_{i_1}, l_{i_2}, \ldots, l_{i_k}\}$ $(i_k < n)$ nezadovoljiva, onda je i f nezadovoljivo. Ovo izvođenje tada zapisujemo na sledeći način:

$$f \Leftrightarrow_{unsat(\mathcal{T}_i)} \bot$$

5.1.7 Povlačenje

Neka je f unija dva disjunktna skupa literala — A i B. Pretpostavimo da literali $B = \{l_{i_1}, l_{i_2}, \ldots, l_{i_k}\}$ povlače (u teoriji $\mathcal{T}_i$) neku konjunkciju ili disjunkciju literala C (koja se ne pojavljuje u $A \cup B$). Pretpostavljamo da je saglasnost tog

[2] U opštem slučaju, efekat zamene može da zavisi od poretka promenljivih na koje se primenjuje; zbog jednostavnosti, mi o tome nećemo detaljnije diskutovati.

povlačenja obezbeđena njegovom specifikacijom (koja je označena sa P za $\mathcal{T}_i$). Ako zadržavamo literale iz B, to zapisujemo

$$A \cup B \Rightarrow_{entail^+(P,\mathcal{T}_i)} A \cup B \cup C$$

ili, alternativno,

$$A \cup B \Leftrightarrow_{entail^+(P,\mathcal{T}_i)} A \cup B \cup C$$

Ako ne zadržavamo literale iz B, označavamo to sa

$$A \cup B \Rightarrow_{entail(P,\mathcal{T}_i)} A \cup C$$

dok u nekim slučajevima važi i

$$A \cup B \Leftrightarrow_{entail(P,\mathcal{T}_i)} A \cup C$$

Pravila $\Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$, $\Leftrightarrow_{unsat(\mathcal{T}_i)}$ i $\Leftrightarrow_{entail(P,\mathcal{T}_i)}$ su jedina pravila izvođenja u predloženom opštem okviru koja koriste znanje specifično za domen, tj. znanje specifično za teorije $\mathcal{T}_i$. Neka druga pravila (kao apstrahovanje) koriste informacije o ovim teorijama, ali samo informacije o njihovom jeziku, tj. o njihovoj signaturi.

Primer 5.1 *Nelson/Oppenova procedura za kombinovanje teorija bez kvantifikatora [86, 120] je zasnovana na propagiranju implicitnih jednakosti između različitih teorija.*

Formula F je nekonveksna *(eng. nonconvex) ako F povlači $x_1 = y_1 \vee x_2 = y_2 \vee \ldots x_n = y_n$, ali ni za jedno i između 1 i n formula F ne povlači $x_i = y_i$. Inače, F je konveksna (eng. convex). Teorija je konveksna ako je svaka konjunkcija njenih literala konveksna. Na primer, konkveksne su* PRA*, teorija lista sa* `car`*,* `cdr`*,* `cons` *i* `atom`*, čista teorija jednakosti. Teorija nizova sa* `store` *i* `select`*,* PIA *i strogo multiplikativna aritmetika su primeri nekonveksnih teorija.*

Suštinski, za povlačenje jednakosti, dovoljno je za datu teoriju imati proceduru koja proverava nezadovoljivost datog skupa literala. U Nelson/Oppenovom algoritmu, te procedure mogu biti tretirane kao eksterne funkcije. Ako postoji n promenljivih (iste sorte), treba ispiti sve nejednakosti između njih[3] *i detektovati kao implicitnu jednakost $x = y$ ako je tekući skup literala nezadovoljiv sa $\neg(x = y)$ (drugim rečin, izvođenje $f \Leftrightarrow_{entail^+(no,\mathcal{T}_i)} f \cup \{x = y\}$ je ekvivalentno sa $f \cup \{\neg(x = y)\} \Leftrightarrow_{unsat(\mathcal{T}_i)} \bot$). Ako je teorija nekonveksna, potrebno je razmatrati ne samo nejednakosti, već i konjunkcije nejednakosti. Kada je kao protivrečna detektovana konjunkcija nejednakosti, onda je njena negacija (disjunkcija jednakosti) indukovana tekućom formulom i dalje vodi u razmatranje više slučajeva (koji odgovaraju disjunktima).*

Za neke teorije postoje efikasni algoritmi za detektovanje implicitnih jednakosti i Nelson i Oppen navode neke od njih u svom originalnom radu (npr. simpleks algoritam za PRA*).*

[3] Razmatranje parova promenljivih može biti svedeno na promenljive deljene od strane više teorija [120]. Mi, međutim, nećemo razmatrati ovu optimizaciju originalne Nelson/Oppenove procedure.

Označavamo sa NO povlačenje za teoriju $\mathcal{T}_i$, takvo da ono povlači jednakost E (ili disjunkciju jednakosti E ako je teorija $\mathcal{T}_i$ nekonveksna) na osnovu skupa L svih $\mathcal{T}_i$–literala. Izvedene jednakosti (ili disjunkcije jednakosti) E se dodaju tekućem skupu literala i ovo izvođenje označavamo sa

$$A \cup L \Rightarrow_{entail^+(NO,\mathcal{T}_i)} A \cup L \cup E$$

ili, alternativno, sa

$$A \cup L \Leftrightarrow_{entail^+(NO,\mathcal{T}_i)} A \cup L \cup E$$

Primer 5.2 *Shostakova procedura [104, 32] za kombinovanje teorija bez kvantifikatora odnosi se na algebarski rešive σ–teorije. Teorija T je σ–teorija ako postoji funkcija kanonizator σ iz skupa termova u skup termova takva da važe sledeći uslovi:*

1. *jednakost $t = u$ je valjana u teoriji ako i samo ako važi $\sigma(t) = \sigma(u)$.*
2. *ako je t term koji ne pripada teoriji, onda je $\sigma(t) = t$*
3. *$\sigma(\sigma(t)) = \sigma(t)$*
4. *ako je $\sigma(t) = f(t_1, \ldots, t_n)$ (f je bilo koji funkcijski simbol) za term t koji pripada teoriji, onda je $\sigma(t_i) = t_i$ za $1 \leq i \leq n$.*
5. *$\sigma(t)$ ne sadrži promenljive koje nisu sadržane u t.*

Na primer, svaki term koji pripada realnoj linearnoj aritmetici može da bude sveden u kanonizovani oblik $a_1x_1 + a_2x_2 + \ldots a_nx_n + c$ ($n \geq 0$) pri čemu postoji neko uređenje nad simbolima x_i ($0 \leq i \leq n$).

σ–teorija $\mathcal{T}$ je algebarski rešiva ako postoji izračunljiva funkcija, solve koja za jednakost e vraća ili $\top$ ili $\bot$ ili konjunkciju jednakosti i ima sledeća svojstva (neka je $solve(e) = E$):

1. *E i e su ekvivalentni, tj. svaki model koji zadovoljava E zadovoljava e i svaki model koji zadovoljava e može da se proširi do modela koji zadovoljava E (jer E može da sadrži promenljive koje se ne pojavljuju u e);*
2. *$E \in \{\top, \bot\}$ ili $E = \bigwedge_i (x_i = t_i)$*
3. *ako e ne sadrži promenljive, onda je $E \in \{\top, \bot\}$*
4. *ako je $E = \bigwedge_i (x_i = t_i)$ onda važi:*
   (a) *x_i se pojavljuje u e*
   (b) *za svako i i j, x_i se ne pojavljuje u t_j*
   (c) *za svako i i j, $i \neq j$ povlači $x_i \neq x_j$*
   (d) *$\sigma(t_i) = t_i$*

Naglasimo da funkcija solve može detektovati nezadovoljivost jednakosti pa, odatle, i nezadovoljivost formule koja se dokazuje. Štaviše, uvek kada je jednakost ekvivalentna sa $\bot$, funkcija solve to može da detektuje.

Među algebarski rešivim σ teorijama su jednakosna realna linearna aritmetika, jednakosna celobrojna linearna aritmetika, konveksna teorija lista, monadička teorija skupova itd. Na primer, funkcija solve za realnu linearnu aritmetiku za jednakost oblika $a_1x_1 + \ldots a_nx_n + c = b_1x_1 + \ldots b_nx_n + d$ vraća jednakost $x_1 = ((b_2 - a_2)/(a_1 - b_1))x_2 + \ldots ((b_n - a_n)/(a_1 - b_1))x_n + (d - c)$; funkcija solve za celobrojnu linearnu aritmetiku za jednakost $17x - 49y = 30$ vraća $x = 49z - 4 \wedge y = 17z - 2$; funkcija solve za teoriju lista za jednakost $cons(car(x)), cdr(car(y))) = cdr(cons(y, x))$ vraća $car(x) = car(x) \wedge y = cons(cons(a, cdr(x)), d)$.

Ako je $solve(e) \equiv E$, onda ovu vrstu izvođenja označavamo sa

$$A \cup \{e\} \Rightarrow_{entail(solve,\mathcal{T})} A \cup E$$

ili, alternativno, sa

$$A \cup \{e\} \Leftrightarrow_{entail(solve,\mathcal{T})} A \cup E$$

Ovo pravilo rešava samo jednu jednakost u jednoj primeni (tj. ne rešava iscrpno više ili sve nerešene jednakosti iz date formule) U ovom pravilo e može biti i nejednakost, tj. e može biti oblika $\neg d$ (gde je d nejednakosti); u tom slučaju, E je jednako $\neg D$, gde je $solve(d) \equiv D$. Kažemo da je jednakost (ili nejednakost) e rešena *ako je $e \equiv solve(e)$. Ako su sve jednakosti i nejednakosti u formuli koja se dokazuje rešene, onda je pravilo $\Leftrightarrow_{entail(solve,\mathcal{T})}$ neuspešno i nema nikakvog efekta.*

Primer 5.3 *U radu [68] opisana je procedura (inspirisana Fourierovim metodom [74]) za detektovanje implicitnih jednakosti E iz skupa* PRA *nejednakosti I (videti isti rad u vezi sa ograničenjima u povlačenju implicitnih jednakosti za* PIA *i* PNA*). Ovo izvođenje označavamo sa*

$$A \cup I \Rightarrow_{entail^+(impl-eqs,pra)} A \cup I \cup E$$

ili, alternativno, sa

$$A \cup I \Leftrightarrow_{entail^+(impl-eqs,pra)} A \cup I \cup E$$

Primer 5.4 *Eliminacija promenljivih je posebna vrsta povlačenja. Pretpostavimo da je raspoloživa procedura P za eliminaciju promenljivih za $\mathcal{T}$ (koja može da služi kao osnova za proceduru odlučivanja za teoriju $\mathcal{T}$), tj. procedura koja transformiše formulu f (teorije $\mathcal{T}$) u ekvivalentnu formulu f' sa manje promenljivih.*

Ako postoji podskup L datih literala A ($A = B \cup L$) takav da važi:

- *svi literali iz L pripadaju teoriji $\mathcal{T}$*
- *postoji promenljiva x takva da se ona ne pojavljuje u B*

onda možemo da primenimo proceduru P za eliminisanje promenljivih na skup literala L i na promenljivu x, dajući formulu E (naglasimo da E može da sadrži disjunkcije).

Procedure za eliminisanje promenljivih su najčešće usmerene ka eliminisanju egzistencijalno kvantifikovanih promenljivih i na nju se lako svodi eliminacija univerzalno kvantifikovanih promenljivih. Međutim, tipično, efikasnije verzije procedura ovog tipa mogu se dobiti definisanjem optimizovanih eliminacija posebno za egzistencijalno i za univerzalno kvantifikovane promenljive. Naglasimo da su u opštem okvitu sve promenljive (implicitno) egzistencijalno kvantifikovane i stoga je dovoljno imati raspoloživu samo proceduru za eliminaciju egzistencijalno kvantifikovanih promenljivih.

Eliminaciju svih promenljivih koje ispunjavaju date uslove označavamo sa:

$$B \cup L \Rightarrow_{entail(P,\mathcal{T})} B \cup E$$

ili, za neke procedure i teorije, sa

$$B \cup L \Leftrightarrow_{entail(P,\mathcal{T})} B \cup E$$

Neki primeri povlačenja zasnovanog na eliminaciji promenljivih su:

$$B \cup L \Leftrightarrow_{entail(Cooper,pia)} B \cup E$$

$$B \cup L \Leftrightarrow_{entail(Hodes,pra)} B \cup E$$

$$B \cup L \Leftrightarrow_{entail(Furier,pra)} B \cup E$$

$$B \cup L \Rightarrow_{entail(Hodes,pia)} B \cup E$$

Fourierov metod za eliminaciju promenljivih je upotrebljen u sistemu TECTON *[68] i suštinski je sličan Hodesovoj proceduri upotrebljenoj u sistemu* NQTHM *(za formule bez kvantifikatora). Naglasimo sa Hodesova procedure [52] čuva nezadovoljivost za* PRA*, ali ne i za* PIA*. Cooperova procedura [29] čuva (ne)zadovoljivost za* PIA*, a može biti prilagođena i za* PNA*.*

Razmatramo i jednu oslabljenu formu povlačenja zasnovanog na eliminaciji promenljivih: u ovoj varijanti povlačenja, promenljiva x može da se pojavi (najviše jednom) u skupu B i to u jednakosti oblika $x = f(t)$, gde je f neinterpretirani funkcijski simbol (tj. funkcijski simbol bez bilo kakvih informacija osim onih o njegovoj sorti). Ovo pravilo označavamo sa $\Rightarrow_{entail^-(P,\mathcal{T})}$. Na primer, važi $v_0 = f(x) \wedge b < v_0 \wedge v_0 < a \Rightarrow_{entail^-(Hodes,pra)} b < a$. Ovo pravilo je primarno usmereno ka eliminaciji apstrahovanih promenljivih (promenljivih uvedenih pravilima apstrahovanja). Ako je $\Rightarrow_{entail(P,\mathcal{T})}$ pravilo koje čuva nezadovoljivost, onda i pravilo $\Rightarrow_{entail^-(P,\mathcal{T})}$ čuva nezadovoljivost. Na primer, pravilo $\Rightarrow_{entail^-(Furier,pra)}$ čuva nezadovoljivost, pa ga označavamo i sa $\Leftrightarrow_{entail^-(Furier,pra)}$.

Za veći broj odlučivih teorija postoje procedure odlučivanja koje se zasnivaju na ideji uzastopne eliminacije kvantifikatora iz formule koja se dokazuje [72].

Neke od procedura za eliminaciju promenljivih mogu se fleksibilno implementirati korišćenjem pravila prezapisivanja (videti [22, 23] i poglavlje D.5).

5.1.8 Konstantno kongruentno zatvorenje/Bazno upotpunjavanje

Pravilo za konstantno kongruentno zatvorenje *ccc* je zasnovano na relaciji $\simeq_C$ (koju definišemo u nastavku) i na algoritmu za njeno izračunavanje. Klase ekvivalencije indukovane konstantnim kongruentnim zatvorenjem se zatim međusvode (dajući bazni kanonski sistem za prezapisivanje) nakon čega se normalizuju svi literali u formuli koja se dokazuje.

Definicija 5.1 Konstantno kongruentno zatvorenje $\simeq_C$ *generisano skupom jednakosti* C *je minimalna relazija ekvivalencije koja zadovoljava sledeća svojstva:*

- *za svaku jednakost* $t_1 = t_2$ *of* C*, važi* $t_1 \simeq_C t_2$*;*
- *ako važi* $t_1 \simeq_C t_2$*, onda je* $f(\ldots t_1 \ldots) \simeq_C f(\ldots t_2 \ldots)$*.*

Kao što je rečeno u radu [124], ova relacija se razlikuje od jednakosnog kongruentnog zatvorenja jer se u konstantnom kongruentnom zatvorenju promenljive tretiraju kao konstante (tj. nije dozvoljeno instanciranje promenljivih).[4]

Konstantno kongruentno zatvorenje može biti izračunato korišćenjem algoritama za kongruentno zatvorenje [104, 87], korišćenjem varijante algoritma za kongruentno zatvorenje interpretiranog kao upotpunjavanje [67] ili korišćenjem Knuth–Bendixove procedure upotpunjavanja za bazne termove (svaki bazni sistem jednakosti dopušta kanonski sistem za prezapisivanje; za veze između kongruentnog zatvorenja i baznog upotpunjavanja i za izvođenje baznog kanonskog sistema iz grafova za kogruentno zatvorenje videti [11, 8]). Kao što je rečeno u [124], možemo koristiti i samo aksiome jednakosti, ali je taj pristup znatno neefikasniji od navedenih metoda.

U našem opštem okviru, izvršavamo konstantno kongruentno zatvorenje nad skupom svih jednakosti iz formule koja se dokazuje. Ono daje klase ekvivalencije; iz svake klase biramo najmanji element (u skladu za uređenjem $\prec$) i izjednačavamo ga sa svim preostalim termovima u toj klasi. Jednakosti se zatim orijentišu kao pravila prezapisivanja (u skladu za uređenjem $\prec$) i međusvode (pri čemu se trivijalne jednakosti eliminišu). Ove orijentisane jednakosti čine bazni kanonski sistem (naglasimo da promenljive tretiramo kao konstante). One zamenjuju polazni skup jednakosti i dalje se koriste za normalizovanje preostalih literala u formuli koja se dokazuje. Ovu transformaciju zvaćemo jednostavno *konstantno kongruentno zatvorenje* i označavaćemo ga sa $\Rightarrow_{ccc}$. Ono čuva nezadovoljivost, pa ga označavamo i sa $\Leftrightarrow_{ccc}$.

Primer 5.5 *Pretpostavimo da je formula koja se dokazuje*

$$\{f(f(f(x))) = x, f(f(f(f(f(x))))) = x\} \ .$$

Svi termovi koji se pojavljuju u datoj formuli se particionišu u klase ekvivalencije algoritmom za konstantno kongruentno zatvorenje i, ovom slučaju, dobijamo

[4] U opštem okviru, bavimo se samo teorijama bez kvantifikatora i koristimo dokaz pobijanjem, pa možemo da tretiramo sve promenljive u negiranom tvrđenju kao konstante.

samo jednu klasu ekvivalencije:

$$\{x, f(x), f(f(x)), f(f(f(x))), f(f(f(f(x)))), f(f(f(f(f(x)))))\}$$

Ova klasa daje sledeći skup jednakosti (orijentisanih sleva nadesno):

$$\{f(x) = x, f(f(x)) = x, f(f(f(x))) = x, f(f(f(f(x)))) = x, f(f(f(f(f(x))))) = x\} .$$

koje, nakon međusvođenje i nakon eliminisanja trvijalnih jednakosti, daju

$$\{f(x) = x\} .$$

Dakle, važi:

$$\{f(f(f(x))) = x, f(f(f(f(f(x))))) = x\} \Leftrightarrow_{ccc} \{f(x) = x\}$$

5.1.9 Superponiranje

Ako je $\mathcal{R}$ kanonski sistem za prezapisivanja takav da se za svaki konačan skup baznih jednakosti E izraženih korišćenjem simbola iz $\mathcal{R}$ i konstantama, upotpunjavanje zaustavlja dajući konačan kanonski sistem za prezapisivanje $\mathcal{R}'$ onda se $\mathcal{R}$ može koristiti za ispitivanje da li neko tvrđenje sledi iz jednakosti E i sâmog sistema $\mathcal{R}$. Kanonski sistem za prezapisanje $\mathcal{R}$ koji ima to svojstvo zovemo *dopustivim* (eng. admissible) sistemom za prezapisivanje.[5] Za dopustivi sistem $\mathcal{R}$, njegova teorija bez kvantifikatora je odlučiva korišćenjem upotpunjavanja. Ako neko pravilo $l \to r$ iz $\mathcal{R}$ može biti superponirano sa nekom jednakošću e iz formule f koja se dokazuje (pri čemu tretiramo promenljive iz f kao konstante), dobijena jednakost e' može da zameni jednakost e. Dodatno, svi literali se normalizuju primenom pravila iz sistema $\mathcal{R}$. Ovo pravilo označavamo sa $\Rightarrow_{superpose(\mathcal{R})}$ ili, alternativno, sa: $\Leftrightarrow_{superpose(\mathcal{R})}$

Primer 5.6 *Može se pokazati da je sistem za prezapisivanje $\{f(f(x)) \to x\}$ dopustiv. Neka je $f(a) = f(b) \wedge a \neq b$ formula koja se dokazuje i neka je $f(a) = f(b)$ orijentisano kao $f(a) \to f(b)$. Onda se $f(a)$ superponira sa $\{f(f(x)) \to x\}$ dajući novu jednakost $a = f(f(b))$ sa normalnom formom $a = b$. Nakon toga, trivijalno se pokazuje da je formula $a = b \wedge a \neq b$ nezadovoljiva.*

Navedeni primer ilustruje činjenicu da normalizovanje pravilima iz dopustivog sistema nije sâmo dovoljno, već je potrebno i superponiranje sa jednakostima iz formule koja se pobija.

Primer 5.7 *Teorija lista sa* `car`, `cdr`, `cons` *i* `atom` *je zadata jednakostima:*

1. $cons(car(x), cdr(x)) = x$
2. $car(cons(x, y)) = x$

[5] Definicija dopustivog sistema i primeri dati u ovom poglavlju preuzeti su iz rada [68]. Ovo makro pravilo prezapisivanja zasnovano je na tom radu.

3. $cdr(cons(x, y)) = y$

Sledeći sistem za prezapisivanje:

1. $cons(car(x), cdr(x)) \rightarrow x$
2. $car(cons(x, y)) \rightarrow x$
3. $cdr(cons(x, y)) \rightarrow y$

je dopustiv sistem za prezapisivanje.

5.1.10 Korišćenje lema

Pretpostavljamo da su raspoloživa neka dodatna pravila prezapisivanja, definicijske jednakosti, dodatne teoreme i leme[6] (naglašavamo da se ne bavimo problemom otkrivanja potrebnih lema — videti, na primer, [69, 65, 3]). Razmatraćemo uslovna pravila prezapisivanja (tj. pravila oblika $l \rightarrow r$ **if** $p_1, p_2, \ldots, p_k$ i $\rho \rightarrow \top$ **if** $p_1, p_2, \ldots, p_k$). Razmatraćemo samo definicijske jednakosti, teoreme i leme bez kvantifikatora i oblika $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow l = r$ i $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$. Zbog toga možemo da ih sve koristimo na isti način kao leme ovog oblika. Pretpostavljamo (kako bismo osigurali zaustavljanje) da je raspoloživo uređenje svođenja $\prec$ takvo da za svaku lemu $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow l = r$ važi $r \prec l$ i $p_i \prec l$ (za $i = 1, 2, \ldots, k$) i za svaku lemu $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$ važi $p_i \prec \rho$ (za $i = 1, 2, \ldots, k$).

Ako je dokazivačka strategija zasnovana na procedurama odlučivanja za teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$ pretpostavljamo da raspoložive leme (tj. raspoloživa dodatna pravila) čine konzistentno konzervativno proširenje za svaku od njih. Tada je smisleno tragati samo za lemama koje nisu formule ovih teorija; zaista, lema koja sadrži nekoj od ovih teorija na može sadržati informaciju koja ne bi mogla da bude izvedena od strane procedura odlučivanja za $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$. Dakle, mehanizam za pozivanje lema formulisaćemo u odnostu na neke teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$.

Neka je t maksimalan strani term (u odnosu na skup teorija $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$) u formuli f koja se pobija (i neka ne postoji strani literal l takav da je $t \prec l$). Neka je c nova apstrakcijska promenljiva za term t (ukoliko već ne postoji jednakost oblika $t = c$). Pretpostavimo da postoji raspoloživa lema (bez kvantifikatora) $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$ u skupu $\mathcal{L}$ takva da važi: maksimalan strani term u ρ (u odnosu na $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$) je term t' i postoji najopštija supstitucija ϕ takva da je $t'\phi$ jednako t. Takođe, pretpostavljamo da su sve promenljive koje su sadržane u lemi instancirane supstitucijom ϕ. Formula f se transformiše u $f \wedge (p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho)\phi$ i u ovoj formuli sva pojavljivanja terma t se

[6] Na primer, pretpostavimo da radimo sa teorijom $\mathcal{T}$ koja uključuje Peanovu aritmetiku. Peanova aritmetika je neodlučiva, pa za nju ne možemo imati proceduru odlučivanja. Međutim, postoje odlučivi fragmenti Peanove aritmetike (kao što je Prezburgerova aritmetika). Dakle, na primer, u dokazivanju teorema teorija $\mathcal{T}$ možemo koristiti proceduru odlučivanja za Prezburgerovu aritmetiku i neke od aksioma koje se odnose na množenje kao dodatna pravila prezapisivanja.

zamenjuju promenljivom c dajući rezultujuću formulu f' (i vode razdvajanju slučajeva ako je $k > 0$). Ako ne postoji takva lema[7], onda se sva pojavljivanja terma t zamenjuju promenljivom c dajući formulu f'. Ovo pravilo izvođenja označavamo sa:

$$f \Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_j)} f'$$

Primer 5.8 *Neka je formula f koja se opovrgava*

$$\{1 \leq v,\ b \leq 0,\ max(a,b) = a,\ min(a,b) = v\},$$

neka je raspoloživa lema

$$max(x,y) = x \Rightarrow min(x,y) = y$$

i pretpostavimo da važi $max \prec min$. Najteži strani term (u odnosu na PRA*) u formuli f je $min(a,b)$, najteži strani term u $min(x,y) = y$ je $min(x,y)$. Supstitucija $\phi = \{x \mapsto a, y \mapsto b\}$ ispunjava potrebne uslove i formula f se prezapisuje u*

$$\{1 \leq v,\ b \leq 0,\ max(a,b) = a,\ min(a,b) = v, \neg(max(a,b) = a) \vee min(a,b) = b\}.$$

Sva pojavljivanja terma $min(a,b)$ se zamenjuju promenljivom v, dajući:

$$\{1 \leq v,\ b \leq 0,\ max(a,b) = a,\ v = v, \neg(max(a,b) = a) \vee v = b\}.$$

Nakon toga, postoji grananje koje daje dve konjunkcije literala:

$$\{1 \leq v,\ b \leq 0,\ max(a,b) = a,\ v = v, \neg(max(a,b) = a)\}$$

i

$$\{1 \leq v,\ b \leq 0,\ max(a,b) = a,\ v = v, v = b\},$$

koje obe lako vode do kontradikcije.

Neka je $\neg\varrho$ maksimalni strani literal (ϱ je atomička formula, strana u odnosu na teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$) u formuli koja se opovrgava takva da ne postoji strani term t u f takav da važi $\varrho \prec t$. Pretpostavimo da postoji lema (bez kvantifikatora) $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho$ raspoloživa u skupu $\mathcal{L}$ takva da postoji najopštija supstitucija ϕ takva da je $\rho\phi$ jednako ϱ. Takođe, pretpostavimo da su sve promenljive koje se pojavljuju u lemi instancirane supstitucijom ϕ. Formula f se transformiše u $f \wedge (p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \rho)\phi$ i u ovoj formuli sva pojavljivanja ϱ se zamenjuju logičkom konstantom $\perp$ dajući rezultujuću formulu f' (i vodeći grananju dokaza ako je $k > 0$). Ako ne postoji takva lema, onda se sva

[7]Za neka tvrđenja sa stranim termovima nisu potrebne leme da bi bila dokazana — na primer, tvrđenje $\forall\alpha\ \forall k\ \ (max(\alpha) \leq k \vee max(\alpha) > k)$ može biti dokazano bez korišćenja lema.To čini ovu oslabljenu varijantu pravila razumnom. U interaktivnim dokaziavačima teorema, u ovim situacijama, korisnik treba da bude obavešten da se pokušaj pobijanja izvodi bez lema i da korisnik treba da razmotri dodavanje nekih lema u slučaju konačnog neuspeha strategije.

pojavljivanja ϱ zamenjuju sa $\bot$, dajući rezultujuću formulu f'. Ovo izvođenje označavamo sa:

$$f \Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_j)} f'$$

Pravilo je dualno ako je ϱ maksimalni strani literal koja se pobija (ϱ je atomička formula, strana u odnosu na teorije $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_j$) u formuli koja se opovrgava takav da ne postoji strani term t u f takav da važi $\varrho \prec t$: tražimo lemu oblika $p_1 \wedge p_2 \wedge \ldots \wedge p_k \Rightarrow \neg\rho$ i sva pojavljivanja ϱ zamenjujemo logičkom konstantom $\top$.

Primer 5.9 *Neka je formula f koja se pobija*

$$\{\neg p(a), g(a) < h(a)\},$$

neka je raspoloživa lema

$$g(x) < h(x) \Rightarrow p(x)$$

i pretpostavimo da važi $g \prec p$ and $h \prec p$. Najteži strani literal (u odnosu na PRA*) u formuli f je $\neg p(a)$. T Supstitucija $\phi = \{x \mapsto a\}$ zadovoljava potrebne uslove i formula f se prezapisuje u*

$$\{\neg p(a),\ g(a) < h(a),\ \neg(g(a) < h(a)) \vee p(a)\}.$$

Sva pojavljivanja $p(a)$ se zamenjuju konstantom $\bot$, dajući:

$$\{\neg\bot,\ g(a) < h(a),\ \neg(g(a) < h(a)) \vee \bot\}.$$

Nakon toga, postoji grananje koje daje dve konjunkcije literala:

$$\{\neg\bot,\ g(a) < h(a),\ \neg(g(a) < h(a))\}$$

i

$$\{\neg\bot,\ g(a) < h(a),\ \bot\}$$

koje obe jednostavno vode do kontradikcije.

Opisani mehanizam za korišćenje lema je oslabljena forma onog koji se koristi u pristupima [68, 69, 2] — korišćenjem opisanog postupka, proces opovrgavanja date formule se nastavlja čak i ako ne postoji raspoloživa lema koja odgovara tekućem maksimalnom stranom termu/literalu. Pravilo $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_j)}$ je, dakle, primenljivo čak i ako je skup $\mathcal{L}$ prazan. Primetimo da je $\Leftrightarrow_{entail^-(P,\mathcal{T})}$ ekvivalentno kombinaciji pravila $\Rightarrow_{lemma^+(\emptyset,\mathcal{T})}$ i $\Leftrightarrow_{entail(P,\mathcal{T})}$ (gde je P procedura za eliminaciju promenljivih za teoriju $\mathcal{T}$).

Ako postoji više od jedne leme koja zadovoljava potrebne uslove, u slučaju neuspešnog opovrgavanja, treba omogućiti bektreking i pokušati sa sledećom raspoloživom lemom koja ispunjava zadate uslove. Kao što je rečeno, čak i kada su sve leme bezupešno isprobane, može se pokušati bez primene ijedne leme u tom koraku.

U pristupima opisanim u [68, 69, 2], kada se koriste uslovna pravila prezapisivanja, za svaki uslov postoji po jedan poddokaz i to vodi dokazima u obliku

stabla. To je analogno sa ovde predloženim mehanizmom za korišćenje lema, jer za zaključak i za svaki uslov iz upotrebljenog uslovnog pravila (odnosno leme), postoji po jedna disjunkcija (nakon primene $\Leftrightarrow_{dnf}$ and $\Leftrightarrow_{disj}$) koja treba da bude pobijena.

Postoje i dodatne varijane pravila za korišćenje lema [124, 68, 69, 59] (npr. korišćenje lema može biti suženo u zavisnosti od dominantnog funkcijskog simbola), ali njih nećemo razmatrati.

Primetimo da je kombinacija pravila $\Leftrightarrow_{ccc}$ i $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T})}$ (zajedno sa $\Leftrightarrow_{dnf}$ i $\Leftrightarrow_{disj}$) ekvivalentna kontekstualnom prezapisivanju [124] do na dualnost (kontekstualno prezapisivanje se koristi za pojednostavljivanje klauza) i do na restrikcije u izboru lema.

Naglasimo da se polje potpunosti obično napušta kada se koriste leme [68, 69, 59]. Štaviše, čak i kada važi $\mathcal{T}, \mathcal{L}, f \vdash \bot$, nije nužno da se nezadovoljivost formule f može pokazati korišćenjem opisanog mehanizma za korišćenje lema (ili mehanizama opisanih u radovima in [17, 68, 2]). Ovo ilustruje sledeći primer (iz [124]):

Primer 5.10 *Neka je $\mathcal{L}$ skup koji se sastoji od sledeća dva pravila:*

$$q \Rightarrow \neg p(a, x)$$

$$\neg q \Rightarrow \neg p(x, b)$$

Tada je $\neg p(a, b)$ logička posledica $\mathcal{L}$, ali nezadovoljivost formule $p(a, b)$ ne može biti pokazana korišćenjem ni prvog ni drugog pravila.

U rešavanju ovog problema, može da se koristi jedna oslabljena verzija kontekstualnog prezapisivanja [123] koja ne zahteva da su uslovi pravila dokazani da bi pravilo bilo primenjeno. Za zadržavanje potpunosti pri korišćenju dodatnih pravila prezapisivanja, predložen je pojam *dobre–pokrivenosti* (eng. well-coveredness) u [123], kao i neke varijante u [14].

Mehanizam za pozivanje lema koji je ovde predložen može biti podešen tako da u nekim slučajevima razrešava ovaj problem. Naime, umesto korišćenja samo jedne leme koja zadovoljava date uslove, možemo da primenimo sve takve raspoložive leme. Tako, u navedenom primeru, tvrđenje $p(a, b)$ bi bilo transformisano u $p(a, b) \wedge (q \Rightarrow \neg p(a, b)) \wedge (\neg q \Rightarrow \neg p(a, b))$ i primenom pravila $\Leftrightarrow_{dnf}$ and $\Leftrightarrow_{disj}$ bilo bi dalje transformisano u četiri podcilja: $p(a, b) \wedge q \wedge \neg q$, $p(a, b) \wedge q \wedge \neg p(a, b)$, $p(a, b) \wedge \neg p(a, b) \wedge \neg q$ i $p(a, b) \wedge \neg p(a, b) \wedge \neg p(a, b)$ od kojih svaki jednostavno vodi do kontradikcije. Ova verzija pravila proširuje domen osnovnog mehanizma za korišćenje lema, kao i domen opšteg okvira koji se predlaže.

5.1.11 Svojstva makro pravila izvođenja

Nije teško dokazati saglasnost svih predstavljenih makro pravila izvođenja i dokazati da pravila $\Leftrightarrow_X$ čuvaju nezadovoljivost.

Svojstva pravila $\Leftrightarrow_{entail(solve,\mathcal{T})}$ su indukovana svojstvima funkcije *solve*. Dokazi svojstava pravila $\Leftrightarrow_{entail^+(impl-eqs,pra)}$, $\Leftrightarrow_{entail(Furier,pra)}$, $\Leftrightarrow_{entail(Cooper,pia)}$

i $\Leftrightarrow_{entail(Hodes,pra)}$ zasnovani su na dokazima iz [74], [68], [52] i [29] redom. Saglasnost pravila $\Rightarrow_{entail(Hodes,pia)}$ može biti dokazana na osnovu saglasnosti $\Leftrightarrow_{entail(Hodes,pra)}$ i na osnovu jednostavnog model–teoretkog tvrđenja 2.5. Dokaz svojstava pravila $\Leftrightarrow_{ccc}$ zasnovan je na svojstvima baznog upotpunjavanja (videti, na primer, [8]). Dokaz svojstava pravila $\Leftrightarrow_{superpose(\mathcal{R})}$ je dat (tj. skiciran) u radu [68]. Čuvanje zadovoljivosti i nezadovoljivosti od strane preostalih pravila može se jednostavno dokazati na osnovu svojstava teorije prvog reda sa jednakošću. Dakle, važi sledeće tvrđenje (f je formula bez kvantifikatora):

Teorema 5.1 *Ako važi $f \Rightarrow^* f'$ i ako je f' nezadovoljivo, onda je i f nezadovoljivo. Ako važi $f \Leftrightarrow^* f'$ i ako je f nezadovoljivo, onda je i f' nezadovoljivo.*

Neposredna posledica navedene teoreme je sledeće tvrđenje: ako važi $f \Rightarrow^* \bot$ (ili $f \Leftrightarrow^* \bot$), onda je f nezadovoljivo i $\neg f$ je teorema; ako važi $f \Leftrightarrow^* \top$, onda je f zadovoljivo i $\neg f$ nije teorema. Ako važi $f \Rightarrow^* \top$ (a ne važi $f \Leftrightarrow^* \top$), onda je *nepoznato* da li je $\neg f$ teorema ili nije.

Zaustavljanje svakog pravila izvođenja može biti dokazano jednostavno (eventualno sa izuzetkom pravila $\Leftrightarrow_{ccc}$ za koje se zaustavljanje bazira na zaustavljanju baznog upotpunjavanja). Međutim, zaustavaljanje shema izgrađenih od više makro pravila izvođenja mora da bude dokazano posebno.

Nećemo diskutovati složenost izračunavanja makro pravila izvođenja i shema koje su nad njima izgrađene. Složenost izračunavanja makro pravila izvođenja kreće se u opsegu od linearne do supereksponencijalne (za Cooperovu eliminaciju promenljivih za PIA [90]). Ukupna složenost izračunvanja kombinacionih/integracionih shema zavisi od složenosti pravila koje se koriste.

5.2 Analiza pojedinačnih shema

U ovom delu daćemo opis nekoliko metoda za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema na bazi uvedenih makro pravila izvođenja.

Naglašavamo da ovi opisi ne reprezentuju potpuno verno originalne metode[8], već oslikavaju njihove osnovne ideje. Dati opisi mogu na različite načine biti optimizovani (slično kao originalne procedure). Međutim, suština opisanog opšteg okvira je u odabiru skupa zajedničkih i ključnih ideja koje mogu da predstavljaju osnovu za novi, moćni i fleksibilni okvir za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema. Verujemo da ove prednosti nadmašuju potencijalne gubitke u efikasnosti.

U nastavku, u opisima metoda koristimo kraći termin "primeni [pravilo izvođenja]", iako bi preciznije bilo reći "pokušaj da primeniš [pravilo izvođenja]". Neka pravila izvođenja su uvek primenljiva, mada moguće bez efekta (kao, na primer, pravilo *ccc*).

U opisima shema koje slede, kada je korišćeno, pravilo $\Leftrightarrow_{simpl(\mathcal{T}_i)}$ može biti zamenjeno pravilom $\Leftrightarrow_{simpl}$, ali su verzije sa $\Leftrightarrow_{simpl(\mathcal{T}_i)}$ najčešće efikasnije.

[8] Ovo je posebno istaknuto u slučaju sheme u Shostakovom stilu, čija originalna verzija je veoma kompaktna i optimizovana.

5.2.1 Nelson/Oppenovo kombinovanje teorija

Neka je $\mathcal{T}$ teorija bez kvantifikatora koja je kombinacija teorija $\mathcal{T}_i$ $(1 \leq i \leq k)$ koje nemaju zajedničke nelogičke simbole osim jednakosti. Neka je F formula koju treba dokazati. Procedura ima sledeći oblik (njen ulaz je $\neg F$):

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$
4. Primeni $\Leftrightarrow_{dnf}$
5. Primeni $\Leftrightarrow_{disj}$
6. Primeni $\Leftrightarrow_{unsat(\mathcal{T}_i)}$, za neko i, $1 \leq i \leq k$
7. Primeni $\Leftrightarrow_{entail^+(NO,\mathcal{T}_i)}$, za neko $1 \leq i \leq k$; ako je pravilo neuspešno, vrati $\top$.
8. Primeni $\Leftrightarrow_{repl=}$
9. Idi na korak 4

Ako su teorije $\mathcal{T}_i$ konveksne, primetimo da u koraku 7, može da se ide na korak 6, umesto na korak 4. Zamena koju vrši pravilo $\Leftrightarrow_{repl=}$ ne koristi se u originalnoj proceduri. Ono je korisno jer smanjuje broj promenljivih koje se razmatraju u traganju za implicitnim jednakostima.

Primer 5.11 *Neka je*

$$x \leq y \wedge y \leq x + car(cons(0, x)) \wedge p(h(x) - h(y)) \rightarrow p(0)$$

formula koju treba dokazati [86]. Ona pripada kombinaciji teorija PRA*, teorije lista (čiji elementi su racionalni brojevi) i čiste teorije jednakosti (označene sa $\mathcal{E}$). Njena negacija je $x \leq y \wedge y \leq x + car(cons(0, x)) \wedge p(h(x) + (-1) * h(y)) \wedge \neg p(0)$ i potrebno je pokazati da je ona nezadovoljiva. Jednostavnosti radi, izostavljamo pravila koja su neuspešna ili nemaju efekat:*

$\{x \leq y, y \leq x + car(cons(0, x)), p(h(x) + (-1) \cdot h(y)), \neg p(0)\}$

$\Leftrightarrow_{absp^+(pra,lists,\mathcal{E})}$

$\{x \leq y, y \leq x + v_0, car(cons(v_1, x)) = v_0, 0 = v_1, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(y) = v_3, h(x) = v_4, \neg p(v_1)\}$

$\Leftrightarrow_{entail^+(NO,lists)}$

$\{x \leq y, y \leq x + v_0, car(cons(v_1, x)) = v_0, 0 = v_1, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(y) = v_3, h(x) = v_4, \neg p(v_1), v_1 = v_0\}$

$\Leftrightarrow_{repl=}$

$\{x \leq y, y \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(y) = v_3, h(x) = v_4, \neg p(v_0)\}$

$\Leftrightarrow_{entail^+(NO,pra)}$

$\{x \leq y, y \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(y) = v_3, h(x) = v_4, \neq p(v_0), x = y\}$

$\Leftrightarrow_{repl^=}$

$\{x \leq x, x \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(x) = v_3, h(x) = v_4, \neg p(v_0)\}$

$\Leftrightarrow_{entail^+(NO,\mathcal{E})}$

$\{x \leq x, x \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_3 = v_2, h(x) = v_3, h(x) = v_4, \neg p(v_0), v_3 = v_4\}$

$\Leftrightarrow_{repl^=}$

$\{x \leq x, x \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_4 = v_2, h(x) = v_4, h(x) = v_4, \neg p(v_0)\}$

$\Leftrightarrow_{entail^+(NO,pra)}$

$\{x \leq x, x \leq x + v_0, car(cons(v_0, x)) = v_0, 0 = v_0, p(v_2), v_4 + (-1) \cdot v_4 = v_2, h(x) = v_4, h(x) = v_4, \neg p(v_0), v_0 = v_2\}$

$\Leftrightarrow_{repl^=}$

$\{x \leq x, x \leq x + v_2, car(cons(v_2, x)) = v_2, 0 = v_2, p(v_2), v_4 + (-1) \cdot v_4 = v_2, h(x) = v_4, h(x) = v_4, \neg p(v_2)\}$

$\Leftrightarrow_{unsat(\mathcal{E})}$

$\bot$

Dokazaćemo korektnost navedene procedure za slučaj dve teorije–komponente, pri čemu je dokaz u opštem slučaju jednostavno uopštenje. Kako bismo dokazali da se navedena procedura zaustavlja i da je korektna, koristićemo narednu teoremu [120] (ova teorema se koristi u [120] za dokazivanje korektnosti nedeterministične verzije Nelson/Oppenove procedure):

Teorema 5.2 *Neka su $\mathcal{T}_1$ i $\mathcal{T}_2$ dve stabilno–beskonačne teorije bez kvantifikatora sa disjunktnim signaturama i neka su ϕ_1 and ϕ_2 konjunkcije literala koji pripadaju $\mathcal{T}_1$ i $\mathcal{T}_2$ redom. Neka je V skup promenljivih deljenih od strane ϕ_1 i ϕ_2 i neka je ψ konjunkcija svih nejednakosti $x_i \neq x_j$ takvih da je $x_i, x_j \in V$ i $i \neq j$. Ako je $\phi_1 \wedge \psi$ zadovoljivo u $\mathcal{T}_1$ i $\phi_2 \wedge \psi$ zadovoljivo u $\mathcal{T}_2$, onda je $\phi_1 \wedge \phi_2$ zadovoljivo u $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Teorema 5.3 *Ako su $\mathcal{T}_i$ $(1 \leq i \leq k)$ dve stabilno–beskonačne teorije bez kvantifikatora sa disjunktnim signaturama, onda se navedena procedura, procedura u Nelson/Oppenovom stilu, zaustavlja i korektna je (tj. saglasna je i potpuna), tj. ona vraća $\bot$ ako je $\neg F$ nezadovoljivo (tj. ako je F teorema) i vraća $\top$ ako je $\neg F$ zadovoljivo (tj. ako F nije teorema).*

Dokaz. Pravilo $\Leftrightarrow_{absp^+}$ primenjuje se samo jednom i nakon toga nema uvođenja novih promenljivih. S druge strane, nakon svake uspešne primene pravila $\Leftrightarrow_{entail^+}$, primenjuje se pravilo $\Leftrightarrow_{repl^=}$ koje eliminiše bar jednu promenljivu. U slučaju nekonveksnih teorija, postoji konačan broj grana i u svakoj od njih bar jedna promenljiva je eliminisana primenom pravila $\Leftrightarrow_{repl^=}$. Dakle, u svakoj grani broj promenljivih strogo opada (u iteracijama) i pravilo $\Leftrightarrow_{entail^+}$ ne može biti primenjeno beskonačan broj puta. Stoga se procedura zaustavlja.

Procedura može da vrati samo $\bot$ ili $\top$. Sva makro pravila izvođenja su saglasna, pa ako procedura vrati $\bot$, formula $\neg F$ je nezadovoljiva. Ako procedura vrati $\top$, to znači da u nekom koraku pravilo $\Leftrightarrow_{entail^+}$ nije moglo da detektuje novu implicitnu jednakost (ili novu disjunkciju jednakosti). U tom koraku, restrikcije ϕ_1 i ϕ_2 tekuće formule ϕ samo na $\mathcal{T}_1$ i $\mathcal{T}_2$ literale su zadovoljive (inače bi pravilo $\Leftrightarrow_{unsat}$ detektovalo nezadovoljivost). Kako pravilo $\Leftrightarrow_{entail^+}$ nije uspelo u detektovanju nove jednakosti, to znači da su formule ϕ_1 i ϕ_2 zadovoljive sa konjunkcijama ψ' negacija svih jednakosti između svih promenljivih. Odatle sledi da su formule ϕ_1 i ϕ_2 zadovoljive sa konjunkcijama ψ negacija jednakosti između deljenih promenljivih (mogli smo i da razmatramo samo deljene promenljive). Na osnovu teoreme 5.2, to znači da je i formula ϕ zadovoljiva. Sva preostala korišćena makro pravila izvođenja čuvaju zadovoljivost, pa na osnovu svojstva pravila $\Leftrightarrow_{disj}$, sledi da je polazna formula zadovoljiva. Dakle, procedura vraća $\bot$ ako i samo ako je formula $\neg F$ nezadovoljiva.

5.2.2 Shostakova kombinacija teorija

Neka je teorija $\mathcal{T}$ bez kvantifikatora kombinacija algebarski rešivih σ-teorija $\mathcal{T}_i$ $(1 \leq i \leq k)$. Neka je sa $\mathcal{E}$ označena čista teorija jednakosti. Neka je F formula bez kvantifikatora koju treba dokazati. Procedura u Shostakovom stilu ima sledeći oblik ($\neg F$ je njen ulaz):

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{ccc}$
4. Primeni $\Leftrightarrow_{simpl}$
5. Primeni $\Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k,\mathcal{E})}$
6. Primeni $\Leftrightarrow_{entail(solve,\mathcal{T}_i)}$, za neko i, $1 \leq i \leq k$; ako je pravilo neuspešno, vrati $\top$.
7. Primeni $\Leftrightarrow_{repl}$
8. Idi na korak 1

Primetimo da koristimo funkcije *solve* za pojedinačne teorije pomognute apstrakcijom umesto funkcije *solve* za teoriju $\mathcal{T}$. U pravilu $\Leftrightarrow_{entail(solve,\mathcal{T}_i)}$, apstrakcijske promenljive se rešavaju prve i u pravilu $\Leftrightarrow_{repl}$, apstrakcijske promenljive se zamenjuju prve; na taj način, nakon svake iteracije nema apstrakcijskih promenljvih u formuli koja se dokazuje. Ako je neka promenljiva rešena u jednoj jednakosti, onda ne pokušavamo da je rešimo u drugim jednakostima.

U originalnom Shostakovom algoritmu (kao i u njegovim varijantama) funkcije *solve* za pojedinačne teorije su čvrsto kombinovane specifičnim strukturama podataka i funkcijama kao što su $find$ i σ [32]. U slučaju čiste teorije jednakosti,

$find$ daje kanonskog predstavnika terma u odnosu na do tada obrađene jednakosti. U slučaju kombinovanja algebarski rešivih teorija (bez neinterpretiranih funkcija), funkcija σ vraća kanonskog prestavnika terma u terminima teorije koja se koristi. Kao što je rečeno u radu [32], Shostakova procedura je metod za efikasno kombinovanje ove dve kanonske forme. U navedenoj, proceduri u Shostakom stilu, koristimo funkcije *solve* za pojedinačne teorije i kombinujemo njihive rezultate kongruentnim zatvorenjem, ali to kombinovanje nije tako čvrsto (ni tako efikasno) kao u originalnoj Shostakovoj proceduri.

U originalnom Shostakovom algoritmu, negacije jednakosti se ne obrađuju kao jednakosti, već imaju sledeću ulogu: nakon što su obrađene sve jednakosti obrađene, obe strane svake negacije jednakosti se "normalizuju" u odnosu na obrađene jednakosti (tj. u odnosu na tekuće stanje odgovarajućih struktura podataka) i proverava se da li su sintaksno indentične; ako u nekoj negaciji jednakosti dve strane imaju istu "normalnu" formu, onda je data formula nezadovoljiva, U našoj verziji Shostakovog algoritma, negacije jednakosti dovoljno je kanonizovati i prezapisati baznim kanonskim sistemom indukovanim jednakostima. Međutim, zbog jednostavnosti, tretiramo ih (tj. rešavamo) na isti način kao jednakosti. Kako rešena jednakost može biti zamenjena konjunkcijom jednakosti, rešena negacija jednakosti može biti zamenjena disjunkcijom negacija jednakosti. Zbog toga je potrebno da procedura iterira od koraka 1.

Primer 5.12 *Neka je*

$$z = f(x + (-1) \cdot y) \wedge x = y + z \rightarrow y + f(f(z)) = x$$

formula koju treba dokazati [104]. Ona pripada jednakosnoj realnoj linearnoj aritmetici sa neinterpretiranim funkcijama. Njena negacija je $z = f(x + (-1) \cdot y) \wedge x = y + z \wedge \neg(y + f(f(z)) = x)$ *i potrebno je pokazati da je ona nezadovoljiva. Zbog jednostavnosti izostavljamo pravila koja su neuspešna ili nemaju efekat:*

$\{z = f(x + (-1) \cdot y), x = y + z, \neg(y + f(f(z)) = x)\}$
$\Leftrightarrow_{ccc}$
$\{f(x + (-1) \cdot y) = z, y + z = x, \neg(y + f(f(z))) = x)\}$
$\Leftrightarrow_{absp^+(pra,\mathcal{E})}$
$\{f(v_1) = z, x + (-1) \cdot y = v_1, z + y = x, \neg(y + v_0 = x), f(f(z)) = v_0\}$
$\Leftrightarrow_{entail(solve,pra)}$
$\{f(v_1) = z, x + (-1) \cdot y = v_1, z + y = x, \neg(v_0 = x + (-1) \cdot y), f(f(z)) = v_0\}$
$\Leftrightarrow_{repl}$
$\{f((z + y) + (-1) \cdot y) = z, \neg(f(f(z)) = (z + y) + (-1) \cdot y)\}$
$\Leftrightarrow_{absp^+(pra,\mathcal{E})}$
$\{f(v_1) = z, (z + y) + (-1) \cdot y = v_1, \neg(f(f(z)) = v_0), v_0 = (z + y) + (-1) \cdot y\}$
$\Leftrightarrow_{entail(solve,pra)}$
$\{f(v_1) = z, (z + y) + (-1) \cdot y = v_1, \neg(f(f(z)) = v_0), v_0 = z\}$
$\Leftrightarrow_{repl}$
$\{f((z + y) + (-1) \cdot y) = z, \neg(f(f(z)) = z)\}$
$\Leftrightarrow_{absp^+(pra,\mathcal{E})}$
$\{f(v_0) = z, (z + y) + (-1) \cdot y = v_0, \neg(f(f(z)) = z)\}$

$\Leftrightarrow_{entail(solve,pra)}$
$\{f(v_0) = z, z = v_0, \neg(f(f(z)) = z)\}$
$\Leftrightarrow_{repl}$
$\{f(z) = z, \neg(f(f(z)) = z)\}$
$\Leftrightarrow_{ccc}$
$\{f(z) = z, \neg(z = z)\}$
$\Leftrightarrow_{simpl}$
$\bot$

U originalnom Shostakovom radu [104] i u nekim radovima o Shostakovoj proceduri [32] postoje dokazi da se ona zaustavlja i da vraća $\bot$ ako i samo ako je ulazna formula nezadovoljiva. Međutim, Rueß i Shankar su nedavno pokazali [100] da su svi ti dokazi pogrešni, da su Shostakova procedura i sve njene objavljene varijante [32, 9, 11] nepotpune i, štaviše, da se ne zaustavljaju.[9] Ovde navedena procedura je saglasna (jer su sva makro pravila izvođenja saglasna), njen domen je sličan domenu originalnog Shostakovog algoritma i ona je nepotpuna,[10] ali se uvek zaustavlja. U ovom tekstu nećemo pokušati da uvedenim makro pravilima izvođenja oponašamo varijante Shostakove procedure nedavno predložene u radovima [100, 10].

Teorema 5.4 *Ako su $\mathcal{T}_i$ $(1 \leq i \leq k)$ algebarski rešive σ-teorije bez kvantifikatora, navedena procedura u Shostakovom stilu se zaustavlja i saglasna je, tj. ona vraća $\bot$ samo ako je $\neg F$ nezadovoljivo (tj. ako je F teorema).*

5.2.3 Rezonovanje o brojevima u sistemu TECTON

Opis rezonovanja o brojevima u sistemu TECTON [68, 69] dat je sa tri procedure zasnovane na Prezburgerovoj aritmetici. Teorije PNA i PIA se u tom radu takođe razmatraju, ali mi ćemo govoriti samo o varijantama za PRA. Sa $\mathcal{E}$ označavamo čistu teoriju jednakosti. Kao i u originalnom radu, razmatramo samo formule bez kvantifikatora. Pojednostavljivanje PRA termova koje se koristi slično je pojednostavljivanju baziranom na Shostakovim kanonizatorima (odgovarajuće pojednostavljivanje nije precizno opisano u originalnom radu). Dajemo opis tri procedure za PRA predložene u radu [68], ali se usredsređujemo na treću proceduru koja koristi leme kao dodatna pravila.

Teorija PRA sa neinterpretiranim simbolima

Neka je F PRA formula bez kvantifikatora sa neinterpretiranim simbolima koja se dokazuje. Procedura ima sledeći oblik (njen ulaz je $\neg F$):

[9] Rueß i Shankar u [100] daju jednostavan primer koji potvrđuje nepotpunost Shostakovog algoritma: konjunkcija $f(v-1)-1 = v+1, f(u)+1 = u-1, u+1 = v$ je nezadovoljiva, ali to ne može biti utvrđeno Shostakovim algoritmom. Štaviše, on se ne zaustavlja za sledeću ulaznu konjunkciju: $f(v) = v, f(u) = u-1, u = v$.

[10] Procedura u Shostakovom stilu predstavljena ovde takođe ne može da pokaže nezadovoljivost konjunkcije $f(v-1)-1 = v+1, f(u)+1 = u-1, u+1 = v$ (sa sličnim objašnjenjem kao za originalnu proceduru).

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{absp^+(pra,\mathcal{E})}$
4. Primeni $\Leftrightarrow_{simpl(pra)}$
5. Primeni $\Leftrightarrow_{unsat(pra)}$
6. Primeni $\Leftrightarrow_{ccc}$
7. Primeni $\Leftrightarrow_{entail^-(Furier,pra)}$; ako je pravilo uspešno, idi na korak 1
8. Primeni $\Leftrightarrow_{entail^+(impl-eqs,pra)}$
9. Idi na korak 1

Navedena procedura je procedura odlučivanja za PRA sa neinterpretiranim simbolima (tj. sa čistom teorijom jednakosti) [68]. Primetimo da u koraku 7 mora da se ide na korak 1 (umesto na korak 3). To je tako, jer eliminisanje promenljivih može da uvede disjunkcije. U slučaju teorije PRA disjunkcije mogu da budu uvedene eliminacijom promenljivih samo ako su pre te eliminacije postojale negacije jednakosti. U originalnoj proceduri [68] sve negacije jednakosti se eliminišu na početku (pre transformisanja u disjunktivnu normalnu formu). Ova mala modifikacija u odnosu na originalnu proceduru napravljena je zbog namere da se formulišu procedure iz [68] na takav način da one lako mogu biti formulisane i za neke druge teorije i za neke druge bazične procedure odlučivanja (a u nekim od tih slučajeva uvođenje disjunkcija eliminisanjem promenljivih ne može biti izbegnuto). Slično, pravilo $\Leftrightarrow_{entail^+(impl-eqs,\mathcal{T})}$, kada je primenjeno na nekonveksnu teoriju $\mathcal{T}$ može da uvede disjunkcije jednakosti, pa iza koraka 8, u opštem slučaju, treba ići na korak 1 (umesto na korak 3). Teorija PRA je konveksna, pravilo $\Leftrightarrow_{entail^+(impl-eqs,\mathcal{T})}$ ne uvodi disjunkcije i u ovom specijalnom slučaju može se ići na korak 3 iza koraka 8. Isto obrazloženje važi i za naredne dve procedure.

Teorije PRA sa dopustivim sistemom za prezapisivanje

Neka je $\mathcal{R}$ dopustivi sistem za prezapisivanje. Neka je F PRA formula bez kvantifikatora sa neinterpretiranim simbola i sa teorijom bez kvantifikatora sistema $\mathcal{R}$ koju treba dokazati. Procedura ima sledeći oblik (njen ulaz je $\neg F$):

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{absp^+(pra,\mathcal{E})}$
4. Primeni $\Leftrightarrow_{simpl(pra)}$
5. Primeni $\Leftrightarrow_{unsat(pra)}$

6. Primeni $\Leftrightarrow_{ccc}$
7. Primeni $\Leftrightarrow_{superpose(\mathcal{R})}$; ako je pravilo uspešno, idi na korak 3
8. Primeni $\Leftrightarrow_{entail^{-}(Furier,pra)}$; ako je pravilo uspešno, idi na korak 1
9. Primeni $\Leftrightarrow_{entail^{+}(impl-eqs,pra)}$
10. Idi na korak 1

Navedena procedura je procedura odlučivanja za PRA sa čistom teorijom jednakosti i sa interpretiranim funkcijskim simbolima aksiomatizovanim dopusitivim sistemom za prezapisivanje [68].

Teorija PRA sa uslovnim pravilima prezapisivanja

Neka je F PRA formula bez kvantifikatora sa neinterpretiranim i sa interpretiranim simbola[11] koju treba dokazati. Procedura ima sledeći oblik (njen ulaz je $\neg F$):

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{absp^{+}(pra,\mathcal{E})}$
4. Primeni $\Leftrightarrow_{simpl(pra)}$
5. Primeni $\Leftrightarrow_{ccc}$
6. Primeni $\Leftrightarrow_{unsat(pra)}$
7. Primeni $\Leftrightarrow_{entail^{-}(Furier,pra)}$; ako je pravilo uspešno, idi na korak 1
8. Primeni $\Leftrightarrow_{entail^{+}(impl-eqs,pra)}$; ako je pravilo uspešno, idi na korak 1
9. Primeni $\Rightarrow_{lemma^{+}(\mathcal{L},pra)}$
10. Idi na korak 1

U pravilu apstrakcije svi funkcijski i predikatski simboli strani teoriji PRA tretiraju se kao neinterpretirani simboli. U pravilu $\Leftrightarrow_{entail^{-}(Fourier,pra)}$ svi funkcijski simboli koji se pojavljuju u raspoloživim lemama tretiraju se kao interpretirani simboli. Zahvaljujući lema pravilu koje se koristi, pravilo $\Leftrightarrow_{entail^{-}(Fourier,pra)}$ u opisanoj proceduri može biti zamenjeno pravilom $\Leftrightarrow_{entail(Fourier,pra)}$.

Ako navedena procedura vrati $\bot$, onda je formula $\neg F$ nezadovoljiva i F je teorema [68]. Međutim, navedena procedura nije potpuna i može da vrati $\top$ čak i ako je formula $\neg F$ nezadovoljiva. (to može biti slučaj kada je $\neg F \Rightarrow^{*} \top$ a nije $\neg F \Leftrightarrow^{*} \top$; primetimo da je u ovim situacijama *nepoznato* da li je $\neg F$ teorema

[11] U pravilima za apstrahovanje, sve funkcijske i predikatske simbole koji su strani u odnosu na PRA tretiramo kao neinterpretirane simbole.

ili nije). To je karakteristično za sheme koje se odnose na integrisanje procedura odlučivanja u dokazivače teorema i ove sheme obično sâme nisu procedure odlučivanja. Međutim, to se ne smatra bitnim nedostatkom jer ove sheme i jesu usmere ka neodlučivim teorijama — one pokušavaju da prošire doseg procedura odlučivanja. Navedena procedura se donekle razlikuje od originalne u mehanizmu za pozivanje lema: čak i kada nema raspoloživih lema koje odgovaraju tekućem najtežem stranom termu, navedena procedura nastavlja rad i ne objavljuje neuspeh (videti §5.1.10).

Naglasimo da uređenje koje se koristi u pravilima $\Leftrightarrow_{ccc}$ i u $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_j)}$ ne mora nužno da bude isto.

Pretpostavljajući da u uređenju $\prec$ koje se koristi u $\Leftrightarrow_{ccc}$ ne važi $x \prec t$, gde je x neka promenljiva i t neki složeni term (na primer, totalno uređenje nad baznim termovima može biti zasnovanao na veličini termova i njihovom alfabetskom poretku), pravilo $\Leftrightarrow_{ccc}$ (primenjeno nakon $\Leftrightarrow_{absp^+(pra,\mathcal{E})}$) ne uvodi nove strane termove.

Primer 5.13 *Neka su raspoložive leme* $p(x) \Rightarrow f(x) \leq g(x)$ *i* $max(x,y) = x \Rightarrow min(x,y) = y$ *i neka je*

$$p(a) \wedge l \leq f(max(a,b)) \wedge 0 < min(a,b) \wedge a \leq max(a,b) \wedge max(a,b) \leq a \rightarrow l < g(a) + b$$

formula koju treba dokazati [68]. Potrebno je pokazati da je njena negacija nezadovoljiva. Zbog jednostavnosti, izostavljamo pravila koja su neuspešna ili nemaju efekat:

$p(a) \wedge l \leq f(max(a,b)) \wedge 0 < min(a,b) \wedge a \leq max(a,b) \wedge max(a,b) \leq a \wedge \neg(l < g(a) + b)$

$\Leftrightarrow_{absp^+(pra,\mathcal{E})}$

$\{p(a), l \leq v_0, f(max(a,b)) = v_0, 0 < v_1, min(a,b) = v_1, a \leq v_2, max(a,b) = v_2, v_2 \leq a, \neg(l < v_4 + b), g(a) = v_4\}$

$\Leftrightarrow_{simpl(pra)}$

$\{p(a), l \leq v_0, f(max(a,b)) = v_0, 1 \leq v_1, min(a,b) = v_1, a \leq v_2, max(a,b) = v_2, v_2 \leq a, \neg(((-1) \cdot v_4 + 1) + l \leq b), g(a) = v_4\}$

$\Leftrightarrow_{ccc}$

$\{p(a), l \leq v_0, f(v_2) = v_0, 1 \leq v_1, min(a,b) = v_1, a \leq v_2, max(a,b) = v_2, v_2 \leq a, \neg(((-1) \cdot v_4 + 1) + l \leq b), g(a) = v_4\}$

$\Leftrightarrow_{entail^-(Furier,pra)}$ *(eliminated: l)*

$\{p(a), f(v_2) = v_0, 0 \leq v_1 + (-1), min(a,b) = v_1, 0 \leq a + (-1) \cdot v_2, max(a,b) = v_2, 0 \leq v_2 + (-1) \cdot a, g(a) = v_4, v_4 + b \leq v_0\}$

$\Leftrightarrow_{simpl(pra)}$

$\{p(a), f(v_2) = v_0, 1 \leq v_1, min(a,b) = v_1, v_2 \leq a, max(a,b) = v_2, a \leq v_2, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4\}$

$\Leftrightarrow_{entail^+(impl-eqs,pra)}$

$\{p(a), f(v_2) = v_0, 1 \leq v_1, min(a,b) = v_1, v_2 \leq a, max(a,b) = v_2, a \leq v_2, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4, v_2 = a\}$

$\Leftrightarrow_{ccc}$

$\{p(a), f(a) = v_0, 1 \leq v_1, min(a,b) = v_1, a \leq a, max(a,b) = a, a \leq a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4, v_2 = a\}$

$\Leftrightarrow_{entail^-(Furier,pra)}$ *(eliminated: v_2)*
$\{p(a), f(a) = v_0, 0 \leq v1 + (-1), min(a,b) = v_1, max(a,b) = a, g(a) = v_4, 0 \leq v_0 + (-1) \cdot v_4) + (-1) \cdot b, a = a\}$

$\Leftrightarrow_{simpl(pra)}$
$\{p(a), f(a) = v_0, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4)\}$

$\Rightarrow_{lemma^+(\mathcal{L},pra)}$ *(lemma: $p(x) \Rightarrow f(x) \leq g(x)$)*
$\{p(a), v_0 = v_0, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), p(a) \Rightarrow v_0 \leq g(a)\}$

- $\{p(a), v_0 = v_0, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), \neg p(a)\}$

  $\Leftrightarrow_{simpl(pra)}$

  $\bot$

- $\{p(a), v_0 = v_0, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), v_0 \leq g(a)\}$

  $\Leftrightarrow_{absp^+(pra,\mathcal{E})}$

  $\{p(a), v_0 = v_0, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), v_0 \leq v_5, g(a) = v_5\}$

  $\Leftrightarrow_{simpl(pra)}$

  $\{p(a), 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), v_0 \leq v_5, g(a) = v_5\}$

  $\Leftrightarrow_{ccc}$

  $\{p(a), 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq v_0 + (-1) \cdot v_4), v_0 \leq v_5, v_5 = v_4\}$

  $\Leftrightarrow_{entail^-(Furier,pra)}$ *(eliminated: v_0)*

  $\{p(a), 0 \leq v_1 + (-1), min(a,b) = v_1, max(a,b) = a, g(a) = v_4, b \leq 0, 0 = v_4 + (-1) \cdot v_5\}$

  $\Leftrightarrow_{simpl(pra)}$

  $\{p(a), 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, b \leq 0\}$

  $\Rightarrow_{lemma^+(\mathcal{L},pra)}$ *(eliminated: $p(a)$)*

  $\{\top, 1 \leq v_1, min(a,b) = v_1, max(a,b) = a, b \leq 0\}$

  $\Leftrightarrow_{simpl(pra)}$

  $\{1 \leq v_1, min(a,b) = v_1, max(a,b) = a, b \leq 0\}$

  $\Rightarrow_{lemma^+(\mathcal{L},pra)}$ *(lemma: $max(x,y) = x \Rightarrow min(x,y) = y$)*

  $\{1 \leq v_1, v_1 = v_1, max(a,b) = a, b \leq 0, max(a,b) = a \Rightarrow v_1 = b\}$

  - $\{1 \leq v_1, v_1 = v_1, max(a,b) = a, b \leq 0, \neg max(a,b) = a\}$

    $\Leftrightarrow_{simpl(pra)}$

    $\bot$

– $\{1 \leq v_1, v_1 = v_1, max(a,b) = a, b \leq 0, v_1 = b\}$
$\Leftrightarrow_{simpl(pra)}$
$\{1 \leq v_1, max(a,b) = a, b \leq 0, v_1 = b\}$
$\Leftrightarrow_{ccc}$
$\{1 \leq b, max(a,b) = a, b \leq 0, v_1 = b, \}$
$\Leftrightarrow_{unsat(pra)}$
$\bot$

$\Leftrightarrow_{disj}$
$\bot$

$\Leftrightarrow_{disj}$
$\bot$

Teorema 5.5 *Navedena procedura, u stilu* TECTON*, se zaustavlja i saglasna je.*

Dokaz. Pretpostavljamo da u uređenju $\prec$ koje koristi u $\Leftrightarrow_{ccc}$ ne važi $x \prec t$, gde je x neka promenljiva i t je neki složeni term. U tom slučaju, pravilo $\Leftrightarrow_{ccc}$ ne uvodi nove strane termove. Štaviše nijedno od pravila u koracima od 1 do 8 ne ne uvodi nove strane termove i ne uvodi nove promenljive (izuzev pravilo $\Leftrightarrow_{absp^+(pra,\mathcal{E})}$ kada je prvi put primenjeno u okviru bloka koji čine koraci od 1 do 8). Svako od ovih pravila se uvek zaustavlja. Dodatno, pravilo $\Leftrightarrow_{entail^-(Furier,pra)}$ kada je primenljivo, eliminiše bar jednu promenljivu, pa ne može biti primenjeno beskonačan broj puta. Slično važi za pravilo $\Leftrightarrow_{entail^+(impl-eqs,pra)}$. Dakle, blok koraka od 1 do 8 se uvek zaustavlja (ili vraćanjem rezultata procedure ili prelaskom na korak 9). Ako je pravilo za korišćenje lema uspešno primenjeno, ono eliminiše maksimalni strani term (u odnosu na PRA) u skladu sa korišćenim uređenjem svođenja $\prec$ (i ne uvodi teži ili jednako težak strani term). Kako je uređenje $\prec$ dobro zasnovana relacija, ne postoji beskonačan niz takvih formula, pa se, stoga, procedura zaustavlja.

Sva makro pravila izvođenja koja se koriste u navedenoj proceduri u stilu TECTON su saglasna, pa je saglasna i procedure. Dakle, ako ona vrati $\bot$, onda je $\neg F$ nezadovoljivo i polazna formula F je teorema. Dodatno, ako ona vrati $\top$ nizom izvođenja $\Leftrightarrow^*$, polazna formula nije teorema.

Dodatno, navedena procedura je potpuna za PRA sa čistom teorijom jednakosti.

Primetimo da Prezburgerova aritmetika u opisanoj shemi može biti zamenjena nekom drugom odlučivom teorijom (za koju su date primitivne procedure za otkrivanje nezadovoljivosti itd), na primer, strogo multiplikativnom aritmetikom.

Opisana shema odgovara ograničenom kontekstulnom prezapisivanju instanciranom za linearnu aritmetiku (tj. PRA) i čistu teoriju baznih jednakosti (označenoj $CCR(DP_{lacc})$ u radu [2]).

5.2.4 EPM shema

U delu 4 dat je opis sheme EPM za fleksibilno integrisanje procedura odlučivanja u dokazivače teorema. Taj okvir može biti upotrebljen za različite teorije i za različite procedure odlučivanja. On je zasnovan na procedurama za eliminaciju kvantifikatora i nova takva procedure može biti jednostavno uključena u sistem. Neka je $\mathcal{T}^e$ neodlučiva teorija bez kvantifikatora, neka je $\mathcal{T}$ njen potpun i odlučiv fragment koji dopušta eliminaciju kvantifikatora i neka je A odgovarajuća procedura za eliminaciju kvantifikatora.[12] Sa $\mathcal{E}$ označavamo čistu teoriju jednakosti[13] Neka je F formula teorije $\mathcal{T}^e$ koju treba dokazati. Shema u stilu EPM ima sledeći oblik (njen ulaz je $\neg F$):

1. Primeni $\Leftrightarrow_{dnf}$
2. Primeni $\Leftrightarrow_{disj}$
3. Primeni $\Leftrightarrow_{simpl(pra)}$
4. Primeni $\Leftrightarrow_{unsat(\mathcal{T})}$; ako pravilo nije uspešno (tj. ako pravilo $\Leftrightarrow_{unsat(\mathcal{T})}$ ne može da detektuje nezadovoljivost) onda ako tekuća formula pripada teoriji $\mathcal{T}$, vrati $\top$, a inače idi na sledeći korak;
5. Primeni $\Leftrightarrow_{absp^+(\mathcal{T})}$, zatim $\Leftrightarrow_{entail(A,\mathcal{T})}$ i zatim $\Leftrightarrow_{repl}$; ako su pravila uspešno primenjena, idi na korak 1
6. Primeni $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T})}$
7. Idi na korak 1.

U pravilu $\Leftrightarrow_{absp^+(\mathcal{T})}$ svi funkcijski i predikatski simboli strani teoriji $\mathcal{T}$ se tretiraju kao neinterpretirani simboli. U pravilu $\Leftrightarrow_{repl}$, apstrakcijske promenljive su zamenjene prve. Primetimo da pravilo $\Leftrightarrow_{entail(A,\mathcal{T})}$ može da uvede disjunkcije (na primer, za $A = Cooper$ i $\mathcal{T}$ =PIA), pa (u opštem slučaju) treba ići na korak 1 nakon koraka 5.

Ako opisana procedura vrati $\bot$, polazna formula je F teorema; ako vrati $\top$ i korak 6 nije primenjen, onda polazna formula nije F teorema (videti 4 i [59]). Međutim, procedura nije potpuna i ona može da vrati $\top$ i ako je polazna formula F teorema, ne uspevši da je dokaže. Ove situacije javljaju se kada je $\neg F \Rightarrow^* \top$, ali ne i $\neg F \Leftrightarrow^* \top$, tj. u situacijama kada navedena procedura vrati $\top$ i pravilo za korišćenje lema jeste bilo primenjeno. U ovim situacijama, procedura može da vrati *nepoznato* kao rezultat.

Opisana procedura (ili, preciznije, shema) se razlikuje od originalne verzije u nekoliko aspekata, uključujući restrikcije u pozivanju lema. Dodatno, u originalnoj verziji procedure, jednakosno rezonovanje se izvodi na osnovu korišćenja

[12] Pretpostavljamo da je $\mathcal{T}^e$ konzervativno proširenje teorije $\mathcal{T}$. Ako je teorija $\mathcal{T}$ potpuna, taj uslov je trivijalno ispunjen.

[13] U pravilu apstrakcije sve funkcijske i predikatske simbole strane teoriji $\mathcal{T}$ tretiramo kao neinterpretirane simbole.

aksioma supstitucije kao dodatnih pravila, odnostno lema. U navedenoj proceduri, slaba tačka je upravo jednakosno rezonovanje (što u predloženom opštem okviru lako može da se promeni dodavanjem pravila $\Leftrightarrow_{ccc}$). Međutim, slabo jednakosno rezonovanje čini ovu proceduru efikasnijom nego što je, na primer, procedura opisana u §5.2.3, na problemima u kojima jednakosno rezonovanje nije potrebno ili nije korisno. Navedena procedura razlikuje se od originalne i po tome što se u originalnoj verziji najteži term eliminiše u okviru pravila za pozivanje lema procedurom za eliminisanje promenljivih. Originalna procedura definisana je za lame i tvrđenja (bez kvantifikatora) proizvoljne strukture i ne transformiše ih u disjunktivnu normalnu formu.

Primer 5.14 *Neka je raspoloživa lema* $minl(x) \leq maxl(x))$ *i neka je*

$$l \leq minl(\alpha)) \wedge 0 < k \rightarrow \neg(maxl(\alpha) + k \leq l)$$

formula koju treba dokazati [17, 59]. Ona pripada teoriji PRA *proširenoj funkcijama* $minl$ *i* $maxl$*. Treba da pokažemo da je njena negacija nezadovoljiva. U* EPM *shemi instanciranoj za* PRA *koristimo Hodesovu proceduru kao algoritam A. Izostavljamo pravila koja su neuspešna ili nemaju efekat:*

$\{l \leq minl(\alpha)), 0 < k, maxl(\alpha) + k \leq l\}$
$\Leftrightarrow_{simpl(pra)}$
$\{l \leq minl(\alpha), 1 \leq k, maxl(\alpha) + k \leq l\}$
$\Leftrightarrow_{absp^+(pra)}$, $\Leftrightarrow_{entail(Hodes,pra)}$ *(eliminated:* k*),* $\Leftrightarrow_{repl}$
$\{0 \leq minl(\alpha) + (-1) \cdot l, 1 \leq l + (-1) \cdot maxl(\alpha)\}$
$\Leftrightarrow_{absp^+(pra)}$, $\Leftrightarrow_{entail(Hodes,pra)}$ *(eliminated:* l*),* $\Leftrightarrow_{repl}$
$\{maxl(\alpha) + 1 \leq minl(\alpha)\}$
$\Rightarrow_{lemma^+(\mathcal{L},pra)}$ *(lemma:* $minl(x) \leq maxl(x)$*)*
$\{v_1 + 1 \leq minl(\alpha), minl(\alpha) \leq v_1, v_1 = v_1\}$
$\Leftrightarrow_{absp^+(pra)}$, $\Leftrightarrow_{entail(Hodes,pra)}$ *(eliminated:* v_1*),* $\Leftrightarrow_{repl}$
$\{minl(\alpha) \leq minl(\alpha) + (-1)\}$
$\Rightarrow_{lemma^+(\mathcal{L},pra)}$ *(eliminated:* $minl(\alpha)$*)*
$\{v_0 \leq v_0 + (-1)\}$
$\Leftrightarrow_{unsat(pra)}$
$\bot$

Teorema 5.6 *Opisana procedura, u* EPM *stilu, je zaustavljajuća i saglasna.*

Dokaz. U opisanoj proceduri, ako je korak 5 uspešno primenjen, on ne ostavlja apstrakcijske promenljive uvedene od strane pravila $\Leftrightarrow_{absp^+(\mathcal{T})}$ (jer se apstrakcijske promenljive zamenjuju prve) i bar jedna dodatna promenljiva je eliminisana; dakle, formula prosleđena koraku 1 ima manje promenljivih neko formula u prethodnoj iteraciji. Ako je korak 6 uspešno primenjen, on eliminiše najteži strani term (u odnosu na $\mathcal{T}$) u skladu sa korišćenim uređenjem svođenja $\prec$. Dakle, u svakoj iteraciji, tekuća formula se transformiše u formulu (ili, pravilima $\Leftrightarrow_{dnf}$ i $\Leftrightarrow_{disj}$, u skup formula) ili sa najtežim stranim termom ne težim nego u prethodnoj iteraciji ili u formulu sa manje promenljivih. Kako je $\prec$ dobro

zasnovana relacija, ne postoji beskonačan niz takvih formula, pa se procedura zaustavlja.

Sva makro pravila izvođenja koja se koriste su saglasna, pa je i procedura saglasna. Dodatno, ako procedura vrati $\top$ i pravilo $\Rightarrow_{lemma^+(\mathcal{L})}$ nije korišćeno, onda to znači da važi $\neg F \Leftrightarrow^* \top$ (je sva ostala pravila čuvaju nezadovoljivost), tj. formula F nije teorema.

Dodatno, opisana procedura je potpuna za formule koje pripadaju teoriji $\mathcal{T}$. Ako se aksiome jednakosti koriste kao leme, onda je opisana procedura procedura odlučivanja za teoriju $\mathcal{T}$ proširenu čistom teorijom jednakosti.

5.3 Svojstva opšteg okvira za kombinovanje i ugradnju procedura odlučivanja

Opisani opšti okvir zasnovan je na izvođenjim koja se primenjuju u poznatim sistemima (i na njihovim varijacijama). Opisi pojedinih procedura i shema u opštem okviru ne reprezentuju potpuno verno originalne metode, već daju njihove ključne ideje i omogućavaju poređenje različitih metoda i njihovo kombinovanje.

Procedure opisane na ovaj način manje su efikasne od čvrsto integrisanih originalnih procedura, ali verujemo da su od toga značajniji dobici na osnovu fleksibilnosti. U nekim shemama, bazno upotpunjavanje se primenjuje u svakoj iteraciji i u međuvremenu se ne čuvaju pravila prezapisivanja izvedena na osnovu tekućih jednakosti; na jednoj strani, ovo čini sistem fleksibilnijim, ali na drugoj strani smanjuje njegovu efikasnost. Međutim, bazno upotpunjavanje može biti izvršeno u vremenu $O(n^2)$ ili $O(n \log n)$ [67, 11] i ovim vremenom dominira vreme izračunavanje potrebno za, na primer, eliminaciju promenljvih ili slični koraci (koji mogu biti eksponekcijalne ili superekponencijalne složenosti). Situacija je slična sa drugim pravilima koja se primenjuju manje restriktivno nego u originalnim shemama i verujemo da ovi gubici u efikasnosti u shemama implementiranim u opštem okviru nisu značajni. Verujemo da predloženi okvir predstavlja dobar balans između fleksibilnosti i efikasnosti.

Osnovne pogodnosti koje pruža opisani opšti okvir za kombinovanje i ugradnju procedura odlučivanja su sledeće:

- dokazi reprezentovani u terminima primena uvedenih makro pravila izvođenja su relativno jednostavni i laki za razumevanje;
- precizni opisi malog broja makro pravila izvođenja vode do preciznog opisa više različitih kombinacionih/integracionih shema; ovo omogućava formalno rezonovanje o tih shemama;
- u opisanom opštem okviru, različite kombinacione/integracione sheme dele veliki deo kôda što ih čini jednostavnijim za implementiranje); u isto vreme, ceo okvir je izgrađen potpuno fleksibilno, od različitih modula.

- opšti okvir omogućava raspoloživost više različitih shema i za kombinovanje i za integrisanje procedura odlučivanja na uniforman način; ako je više shema primenljivo na tekuće tvrđenje, one mogu biti uređene na osnovu nekih heurističkih skorova koji treba da mogu da se dinamički prilagođavaju teoremama koje su već uspešno dokazane.

- fleksibilna struktura shema implementiranih u predloženom opštem okviru čini mogućom i jednostavnom komunikaciju sa ostalim komponentama i strategijama dokazivača (npr. sa indukcijom); sheme implementirane u opštem okviru ne koriste nikakve dodatne strukture podataka, baze podataka ili ograničenja i sl; jedini spoljni mehanizam koji se koristi je uređenje svođenja nad termovima; nakon svakog koraka u ovim shemama, tekući (pod)cilj može biti sâm prosleđen nekoj drugoj dokazivačkoj strategiji.

- ideje iz različitih shema za korišćenje procedura odlučivanja mogu lako da se kombinuju (na primer, Nelson/Oppenov algoritam može lako biti proširen mehanizmom za pozivanje lema, metod EPM može lako biti proširen pravilom za kongruentno zatvorenje itd).

- svaka shema implementirana u opštem okviru može biti zasnovana na različitim bazičnim procedurama za osnovnu teoriju (npr. može da bira između Hodesove i Cooperove procedure za Prezburgerovu aritmetiku).

- bilo koji modul koji odgovara nekom makro pravilu može biti unapređen ili potpuno promenjen pri čemo je potrebno jedino da zadovoljava svoju specifikaciju (npr. kongruentno zatvorenje može biti zasnovano na Shostakovom umesto na Nelson/Oppenovom algoritmu); to može biti učinjeno bez bilo kakvih posledica po druge komponente opšteg okvira ili nekih njegovih instanci; ovo čini sistem potpuno fleksibilnim i, u isto vreme, pruža mogućnosti za popravljanje njegove efikasnosti.

5.4 General Setting for Combining and Integrating Decision Procedures: Summary

We propose a general setting for describing and implementing different schemes for both combining and integrating decision procedures into theorem provers. The setting is based on macro inference rules motivated by the approaches known from the literature. We denote macro inference rules by $f_1 \Rightarrow_X f_2$, where X is the name of a specific rule. If a rule X is parametrised by parameters P then we denote it by $X(P)$. The soundness of each macro inference rule has to be ensured; i.e., for all rules $f_1 \Rightarrow_X f_2$, it has to be ensured that if f_2 is unsatisfiable, then f_1 is unsatisfiable, too. If the inference rule X is unsatisfiability preserving, we denote it by $\Leftrightarrow_X$. We abbreviate the sequence $f_1 \Rightarrow_{X_{i_1}} f_2$, ..., $f_{n-1} \Rightarrow_{X_{i_{n-1}}} f_n$ by $f_1 \Rightarrow^* f_n$. Similarly, we abbreviate the sequence $f_1 \Leftrightarrow_{X_{i_1}} f_2$, ..., $f_{n-1} \Leftrightarrow_{X_{i_{n-1}}} f_n$ by $f_1 \Leftrightarrow^* f_n$. If $\neg F \Rightarrow^* \bot$ then $\neg F$

is unsatisfiable, so the original formula F is valid. If $\neg F \Leftrightarrow^* \top$ then there is an implicitly constructed counterexample for F, so it is invalid. The setting is built from the following macro inference rules:

- $\Leftrightarrow_{dnf}$ for transforming a formula into disjunctive normal form;
- $\Leftrightarrow_{disj}$ for dealing with disjunctions;
- $\Leftrightarrow_{simpl}$ for general simplification and $\Leftrightarrow_{simpl(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$ for theories specific simplification;
- $\Leftrightarrow_{abs(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$, $\Leftrightarrow_{abs^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$ and $\Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$ for abstracting alien terms;
- $\Leftrightarrow_{repl}$ and $\Rightarrow_{repl^=}$ for replacing isolated variables;
- $\Leftrightarrow_{unsat(\mathcal{T})}$ for detecting unsatisfiability in a specific decidable theory;
- $\Leftrightarrow_{entail^+(P,\mathcal{T})}$, $\Rightarrow_{entail(P,\mathcal{T})}$, $\Leftrightarrow_{entail(P,\mathcal{T})}$, $\Rightarrow_{entail^-(P,\mathcal{T})}$ and $\Leftrightarrow_{entail^-(P,\mathcal{T}}$) for theory–specific entailing; some examples of it are $\Leftrightarrow_{entail^+(NO,\mathcal{T}_i)}$, $\Leftrightarrow_{entail(solve,\mathcal{T})}$, $\Leftrightarrow_{entail(Cooper,pia)}$, $\Leftrightarrow_{entail(Hodes,pra)}$, $\Leftrightarrow_{entail^-(Furier,pra)}$ and $\Rightarrow_{entail(Hodes,pia)}$;
- $\Leftrightarrow_{ccc}$ for congruence closure/ground completion;
- $\Leftrightarrow_{superpose(\mathcal{R})}$ for superposing literals from a formula being proved with rules from available admissible rewriting system;
- $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_j)}$ for lemma invoking.

Within this setting, a number of different procedures for both integrating and combining decision procedures into theorem provers can be formally described, including Shostak's, Nelson–Oppen's, the approach used in the TECTON system, EPM scheme etc. The general setting gives the key ideas of one combination/integration scheme and makes different schemes comparable. All macro inference rules are sound and, therefore, all schemes implemented within our general setting are sound. All macro inference rules are terminating, but termination of different combination/integration schemes has to be proved separately. For some of these schemes it is proved that they are terminating and complete (e.g. for Nelson/Oppen's scheme). Concerning efficiency, procedures obtained in this way (in this setting) would be less efficient than some tightly integrated procedures, but we believe that gains from the flexibility overcome the potential losses in efficiency. Some of the main features of this general setting are the following: proofs are relatively simple and easy to understand; precise descriptions of only several macro inference rules lead to precise descriptions of number of combination/integration schemes (which enables formal reasoning about different combination/integration schemes); within the described general setting, different combination/integration schemes share a lot of code (which makes them easier to implement); it provides the possibility of having available different schemes both for combination and integration of decision procedures

in a uniform way; ideas from different schemes for combining and integrating decision procedures can be easily combined; any module that corresponds to an inference rule can be improved or completely changed as long as it meets its specification (this makes the system fully flexible and, at the same time, provides an opportunity for improving its efficiency).

We have implemented the proposed general setting and some schemes within it. The description of this implementation and some experimental results are given in chapter 6.

Glava 6

Implementacija i rezultati

Shema EPM (i neke njene instance), kao i opšti okvir za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema implementirani su u SWI–PROLOG-u (verzija 2.9; autor Jan Wielemaker, Department of SWI, University of Amsterdam) u okviru sistema *Clam*, baziranom na paradigmi planiranja dokaza. Paradigma planiranja dokaza je pogodna za opis i implementaciju predloženi opštih okvira, ali se oni mogu implementirati i koristiti i u dokazivačima zasnovanim na drugim paradigmama.

6.1 Planiranje dokaza i *Oyster/Clam* sistem

Planiranje dokaza (eng. proof–planning) je tehnika za vođenje pretrage u dokazivanju teorema [21, 20]. Plan dokaza (eng. proof plan) je skica dokaza, ona pokriva osnovne tipove zaključivanja u familijama sličnih dokaza i koristi se za pružanje globalne kontrolne strategije za nalaženje novih dokaza iz iste familije. Kada se dokazuje neko tvrđenje, najpre se konstruiše plan i zatim se taj plan koristi za vođenje kontrukcije sâmog dokaza. Za razliku od "lokalnih" heuristika koje donose posebne odluke u svakoj tački izbora, paradigma planiranja dokaza odnosi se prema dokazu kao celini. Takav pristup čini planiranje dokaza bliže intuiciji matematičara koji obično polaze od globalnog plana dokaza, a zatim razrešavaju detalje.

Paradigma planiranje dokaza zasniva se na sledeća tri tipa entiteta:

taktike: taktike su programi koji konstruišu deo dokaza primenjivanje pravilam izvođenja u dokazivačkom sistemu [49]. Jednostavna taktika može da primenjuje samo jedno pravilo izvođenja; složena taktika može da bude definisana u terminima jednostavnih taktika i može da konstruiše čitav dokaz. Taktike imaju hijerarhijsku strukturu; one mogu da budu izgrađene od pravila i podtaktika sekvencijalno, iterativno i uslovno.

metode: metode su specifikacija taktika — njihova deklarativna, ali i izvršna reprezentacija. Metod opisuje preduslove za upotrebu taktike kao i efekte

njenog korišćenja. Ovi preduslovi i efekti su sintaktička svojstva logičkih izraza koje obrađuje taktika i izrečeni su na jeziku meta–logike. U sistemu *Clam*, opis metoda ima šest "slotova" koji su opisani tabelom 6.1.

kritike: kritike pokrivaju najčešće uzroke neuspeha metoda i sugerišu moguće dopune za parcijalne dokaze. Kritika je pridružena metodu i ima sličnu strukturu, s tim što njeni predulsovi opisuju situaciju u kojoj metod propada. Umesto efekta, kritika sadrži uputstvo za dopunjavanje neuspelog plana dokaza.

- **Ime** — ime metoda (obično se za ime metoda uzima ime odgovarajuće taktike, prošireno dodatnim argumentima ukoliko je to potrebno);
- **Ulaz** — shematska reprezentacija formule koja se dokazuje pre primene taktike;
- **Izlaz** — shematska reprezentacija formule koja se dokazuje nakon primene taktike;
- **Preduslovi** — sintaktička reprezentacija dodatnih uslova potrebnih da bi taktika bila primenjena;
- **Efekti** — sintaktička reprezentacija dodatnih efekata metoda, uključujući svojstva izlaza i veze između ulaza i izlaza; ona važe ako je taktika uspešno primenjena;
- **Taktika** — program za primenjivanje pravila izvođenja na objektnom nivou

Tabela 6.1: Slotovi metoda u sistemu *Clam*

Oyster je interaktivni editor dokaza na objektnom nivou kojim se može upravljati taktikama [18]. *Oyster* je, suštinski, reimplementacija sistema NuPRL — editora dokaza za Martin Löfovu intuicionističku teoriju tipova [79, 89] razvijenog na Cornell University [26]. *Clam* je program za formiranje planova dokaza i implementiran je kao proširenje sistema *Oyster* koje podržava planiranje dokaza. Plan dokaza čini skup metoda povezanih sekvencijalno, iterativno i/ili uslovno. *Clam* sadrži mehanizam za reprezentaciju metoda i kritika. On takođe sadrži više planera koji omogućavaju automatsko generisanje planova dokaza na osnovu kombinovanja raspoloživih metoda (npr. najpre u dubinu, najpre u širinu). Sistem *Oyster/Clam* razvija se na Grupi za matematičko rezonovanje Univerziteta u Edinburgu od 1988. godine i do sada je dao mnoge značajne rezultate. Prostor pretrage u sistemu *Oyster/Clam* obično je za nekoliko redova veličina manji nego u prostoru pretrage na objektnom nivou. Osnovna primena paradigme planiranja dokaza i sistema *Oyster/Clam* je u induktivnim dokazima.

Generalno, moguće je da primena planom izabrane taktike ne uspe, te se stoga za planove dokaza ne može garantovati uspešnost (pogotovu u neodlučivim teorijama).

6.2 EPM shema

6.2.1 Implementacija

Shema EPM implementirana je u okviru sistema *Clam* na nivou metoda. Implementirani su metodi koji odgovaraju opštim procedurama pojednostavljivanja opisanim u poglavlju 4.2. Od specijalizovanih procedura implementirane su procedure za PIA zasnovane na Hodesovom algoritmu (videti poglavlje D.1) i procedure za PIA i PNA zasnovane na Cooperovom algoritmu (videti poglavlje D.2). Ove procedure su implementirane na fleksibilan način i zasnivaju se na ideji primene serija normalizacija (videti [22] i poglavlje D.5). Kao ilustraciju, navodimo metod koji odgovara proceduri $\texttt{ElimOneVar}_{\mathcal{T},A}$:

```
method(elim_one_var_T_A(Theory,Algorithm,ListGenTerms,V),
       H==>F,
       [
        generalise_non_T_terms(Theory,F,ListGenTerms,F1),
        matrix(Vars,M,F1),
        member(V:Sort,Vars),
        not ((member(GenTerm-_,ListGenTerms),exp_at(GenTerm,_,V))),
        not member(_Gt-V,ListGenTerms)
       ],
       [
        delete(Vars,V:Sort,Vars1),
        append(Vars1,[V:Sort],Vars2),
        matrix(Vars2,M,F2),
        dp_once(Theory,Algorithm,F2,F3),
        de_generalise(F3,ListGenTerms,Fnew)
       ],
       [H==>Fnew],
       elim_one_var_T_A(Theory,Algorithm,ListGenTerms,V)).
```

Odgovarajuće taktike nisu implementirane. Naime, u ovoj primeni paradigme planiranja dokaza sve taktike (koje su u planu dokaza) su uvek uspešne, pa je čitava pretraga za dokazom koncentrisana u metodama. Dodatno, za generisani plan dokaza, konstruisanje dokaza na objektnom nivou je jednostavno. Zbog toga su implementirani samo potrebni metodi, ali ne i taktike. Drugi razlog za to je činjenica da su opšti okviri koji se predlažu ovom tezom zasnovani na klasičnoj logici, te ne bi uvek bilo moguće interpretirati ih u intuicionističkoj logici sistema *Oyster*.

6.2.2 Rezultati

Implementacija sheme EPM (odnosno njenih instanci) opisana u prethodnom poglavlju uspešno je primenjena za dokazivanje većeg broja standardnih primera iz literature. Neki rezultati (većina primera je preuzeta iz [17]) dati su u tabeli 6.2. [1] Shema EPM primenjena je sa tri različita bazična algoritma za Prezburgerovu aritmetiku (pa je, na primer, promenljiva l sorte $\mathbf{Z}$ u 1a i 1b i sorte $\mathbf{N}$ u 1c itd.). Leme su formule koje su tačne u strukturi prirodnih brojeva (ali mogu da budu prilagođene i za procedure koje rade nad celim brojevima).

#	$\mathcal{T}$	A	leme	CPU vreme (s)
1	$\forall l \forall \alpha \forall k\ (l \leq min(\alpha) \wedge 0 < k \rightarrow l < max(\alpha) + k)$			
1a	PIA	Hodes	$\forall \xi\ (min(\xi) \leq max(\xi))$	0.53
1b	PIA	Cooper	$\forall \xi\ (min(\xi) \leq max(\xi))$	0.80
1c	PNA	Coopers	$\forall \xi\ (min(\xi) \leq max(\xi))$	0.62
2	$\forall lp \forall lt \forall i \forall pat \forall c\ lp + lt \leq maxint \wedge i \leq lt \rightarrow i + delta(pat, lp, c) \leq maxint$			
2a	PIA	Hodes	$\forall x \forall y \forall z\ delta(x, y, z) \leq y$	0.26
2b	PIA	Cooper	$\forall x \forall y \forall z\ delta(x, y, z) \leq y$	0.38
2c	PNA	Cooper	$\forall x \forall y \forall z\ delta(x, y, z) \leq y$	1.87
3	$\forall a \forall b \forall c ms(c) + ms(a)^2 + ms(b)^2 < ms(c) + ms(b)^2 + 2 \cdot ms(a)^2 \cdot ms(b) + ms(a)^4$			
3a	PIA	Hodes	$\forall x\ 0 < ms(x)\ , \forall i \forall j\ 0 < i \rightarrow j \leq i \cdot j$	4.86
3b	PIA	Cooper	$\forall x\ 0 < ms(x)\ , \forall i \forall j\ 0 < i \rightarrow j \leq i \cdot j$	16.24
3c	PNA	Cooper	$\forall x\ 0 < ms(x)\ ,\ \forall i \forall j\ 0 < i \rightarrow j \leq i \cdot j$	?
4	$\forall a \forall b \forall c ms(c) + ms(a)^2 + ms(b)^2 < ms(c) + ms(b)^2 + 2 \cdot ms(a)^2 \cdot ms(b) + ms(a)^4$			
4a	PIA	Hodes	$\forall i \forall j\ j \leq ms(i) \cdot j$	1.47
4b	PIA	Cooper	$\forall i \forall j\ j \leq ms(i) \cdot j$	5.32
4c	PNA	Cooper	$\forall i \forall j\ j \leq ms(i) \cdot j$	0.52
5	$\forall k \forall l\ 0 < k \wedge 0 < l \wedge 2 \cdot k + 1 \leq 2 \cdot l \rightarrow 2 \cdot k + 2 \leq 2 \cdot l$			
5a	PIA	Hodes		/
5b	PIA	Cooper		1.69
5c	PNA	Cooper		0.17
6	$\forall x\ x \leq double(double(x))$			
6a	PIA	Hodes'	$\forall y\ y \leq double(y)$	0.08
6b	PIA	Cooper's	$\forall y\ y \leq double(y)$	0.17
6c	PNA	Cooper's	$\forall y\ y \leq double(y)$	0.26
7	$\forall x\ x \leq triple(x)$			
7a	PIA	Hodes'	$\forall y\ y \leq double(y)\ ,\ \forall y\ double(y) \leq triple(y)$	0.09
7b	PIA	Cooper's	$\forall y\ y \leq double(y)\ ,\ \forall y\ double(y) \leq triple(y)$	0.16
7c	PNA	Cooper's	$\forall y\ y \leq double(y)\ ,\ \forall y\ double(y) \leq triple(y)$	0.27

Tabela 6.2: Eksperimentalni rezultati instanci EPM sheme

Iz tabele se može videti da je za sve tri varijante treće tvrđenje bilo najteže

[1] Testovi su izvršeni na računaru PC Pentium 466Mhz, pod operativnim sistemom Linux. CPU vreme je dato u sekundama.

(varijanta sa Cooperovim algoritmom za PNA je čak premašila osnovni stek). U ovom primeru, trebalo je da leme budu iskorišćene (bar) šest puta što je dovodilo do veoma složenih međurezultata. Međutim, ako zamenimo dve leme iz trećeg primera jednom lemom koja je dovoljna za dokaz (četvrti primer), situacija je promenjena i sistem je znatno efikasniji. Objašnjenje je sledeće: tvrđenja $0 < ms(a)$ i $0 < ms(b)$ nisu mogla biti iskorišćena sve dok termovi $ms(a)$ i $ms(b)$ nisu postali najteži ne-$\mathcal{T}$-termovi, pa su međurezulati bili veoma složene formule (one su imale veliki broj pojavljivanja atomičkih formula $0 < ms(a)$ i $0 < ms(b)$). Ovo nije bio problem u četvrtom primeru. Ovaj problem može biti izbegnut u nekim slučajevima: ako je lema koja se koristi oblika $\forall\vec{x}(H_1 \wedge \cdots \wedge H_k \rightarrow C)$ onda, nakon instanciranja, možemo da pokušamo da dokažemo atomičke formule H_i $(i = 1, \ldots, k)$ posebno i onda, u glavnom dokazu, možemo da koristimo samo atomičku formulu C (sistemi NQTHM i RRL koriste leme na ovaj način). Ovaj pristup može značajno da poveća mogućnosti sistema u nekim slučajevima (na primer, za formulu $\forall a \forall b \forall c\, ms(c) + ms(a)^2 + ms(b)^2 < ms(c) + ms(b)^2 + 2 \cdot ms(a)^2 \cdot ms(b) + ms(a)^4$, ubrzanje bi bilo slično onom koje je dobijeno u četvrtom primeru).

Četvri primer takođe ilustruje činjenicu da Cooperova procedura za PNA može biti znatno efikasnija nego ona za PIA ili Hodesova za PIA. Zbog toga, dobro je imati na raspolaganju sve ove procedure (pri čemu njihov poredak može biti baziran na određenim heurističkim skorovima koji se mogu dinamički podešavati [78]).

Peto tvrđenje nije tačno u strukturi racionalnih brojeva, pa ne može biti dokazano Hodesovim algoritmom (i sledstveno tome, ne može biti dokazano Boyer/Mooreovom procedurom za linearnu aritmetiku), ali može biti dokazano Cooperovim algoritmima za PIA i PNA. Dakle, i ovaj primer ilustruje koristnost većeg broja raspoloživih procedura za Prezburgerovu aritmetiku.

6.3 Opšti okvir za kombinovanje i integrisanje procedura odlučivanja

6.3.1 Implementacija

Opšti okvir za kombinovanje i integrisanje procedura odlučivanja, kao i neke sheme, implementirani su u ukviru sistema *Clam*. Opšti okvir oslonjen je mehanizam za prezapisivanje, uređenje nad termovima sistema *Clam* itd. Makro pravila izvođenja, pa i kombinacione/integracione sheme implementirane su kao eksterne procedure koje proširuju jezik metoda. To je urađeno po analogiji na primenu samostalnih procedura odlučivanja za iskaznu logiku i Prezburgerovu aritmetiku, ali u daljem radu se može modifikovati tako da sva pravila izvođenja i pojedinačne sheme budu implementirani kao metodi. Odgovarajuće taktike nisu implementirane i to iz istih razloga kao i u EPM shemi. Kao ilustraciju navodimo deo implementacije pravila $\Leftrightarrow_{dnf}$, $\Leftrightarrow_{disj}$ i implementaciju Shostakove sheme (pri čemu je mehanizam biblioteka teorija koji se poziva implementiran samo u najosnovnijem obimu):

```
dpi_scheme(Scheme,F,A):- dnf(F,F1),disj(Scheme,F1,A).

disj(Scheme,F1\F2,Fr):-!,disj(Scheme,F1,F1r),disj(Scheme,F2,F2r),
                       ( (refuted(F1r),refuted(F2r))
                        -> Fr=void
                        ; ( (proved(F1r); proved(F2r))
                            -> !, Fr={true}
                            ; Fr=(F1r\F2r))).
disj(Scheme,F1,F3):-
    conjuction_to_list(F1,F2),
    prove_conjuction(Scheme,F2,F3).

prove_conjuction(shostak,[void],[void]):-!.
prove_conjuction(shostak,[{true}],[{true}]):-!.
prove_conjuction(shostak,F,A):-
            solvable(Theories),
            append(Theories,[e],Theories1),
            ccc(F,F1),
            simpl(F1,F2),
            (F2=[void]
            -> A=[void]
            ; (abspplus(F2,Theories1,F3),
               entail(solve,Theories,F3,F4),
               (equal_sets(F3,F4)
               -> A=[{true}]
               ; repl(F4,F5),
                 prove_conjuction(shostak,F5,A)))).
```

Implementirana su sva opšta makro pravila izvođenja, kao i pojedina pravila specifična za teorije:

- $\Leftrightarrow_{dnf}$;
- $\Leftrightarrow_{disj}$;
- $\Leftrightarrow_{simpl}$ i and $\Leftrightarrow_{simpl(\mathcal{T}_1)}$ za PRA, PIA i PNA;
- $\Leftrightarrow_{abs(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$, $\Leftrightarrow_{abs^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$ i $\Leftrightarrow_{absp^+(\mathcal{T}_1,\mathcal{T}_2,\ldots,\mathcal{T}_k)}$; $\Leftrightarrow_{repl}$ and $\Rightarrow_{repl^=}$;
- $\Leftrightarrow_{ccc}$ — na osnovu Nelson/Oppenovog algoritma za kongruentno zatvorenje [87][2] i na način opisan u poglavlju 5.1.8;
- $\Leftrightarrow_{unsat(\mathcal{T})}$ za čistu teoriju jednakosti, teoriju lista i teorije PRA, PIA i PNA; nezadovoljivost za čistu teoriju jednakosti se proverava korišćenjem

[2]Preliminarni testovi govore da vreme potrošeno od strane algoritma za kongruentno zatvorenje ne utiče značajno na ukupno utrošeno CPU vreme, pa je u prvoj implementaciji opšteg okvira iskorišćen Nelson/Oppenov algoritam koji je znatno jednostavniji (mada i manje efikasan) od Shostakovog algoritma

algoritma za kongruentno zatvorenje; nezadovoljivost za teoriju lista se proverava korišćenjem sistema prezapisivanje koji je opisan u primeru 5.7; ispitivanje nezadovoljivosti za PRA, PIA i PNA je zasnovano na Hodesovoj i Cooperovoj proceduri;

- $\Leftrightarrow_{entail^+(no,\mathcal{T})}$; zasnovano je na vezi $f \Leftrightarrow_{entail^+(no,\mathcal{T})} f \cup x = y$ ako i samo ako $f \cup \neg(x = y) \Leftrightarrow_{unsat(\mathcal{T})} \bot$;
- $\Leftrightarrow_{entail(solve,\mathcal{T})}$ za jednakosnu realnu linearnu artimetiku;
- $\Leftrightarrow_{entail(Hodes,pra)}$, $\Rightarrow_{entail(Hodes,pia)}$, $\Rightarrow_{entail(Hodes,pna)}$, $\Leftrightarrow_{entail(Cooper,pia)}$, $\Leftrightarrow_{entail(Cooper,pna)}$ za eliminaciju promenljivih; (Hodesovu procedura za PRA koristimo za PIA i PNA kao saglasnu, ali nepotpunu proceduru);
- $\Rightarrow_{lemma^+(\mathcal{L},\mathcal{T}_1,\mathcal{T}_2,...,\mathcal{T}_j)}$.

Umesto pravila $\Rightarrow_{entail(impl-eqs,pra)}$ koristimo $\Rightarrow_{entail^+(no,\mathcal{T}_i)}$ i umesto pravila $\Leftrightarrow_{entail(Furier,pra)}$ koristimo $\Leftrightarrow_{entail(Hodes,pra)}$.[3]

U procedurama se koriste mehanizami za prezapisivanje sistema *Clam* uključujući uređenje rekurzivne staze sa statuson. Sa implementiranim makro pravilima izvođenja, bilo je jednostavno implementirati sheme za kombinovanje i integrisanje procedura odlučivanja koje su opisane u prethodnim delovima. Implementirane su sledeće sheme: Nelson/Oppenova, Shostakova, TECTON i EPM.

6.3.2 Rezultati

Neki rezultati opisane implementacije opšteg okvira i nekih kombinacionih/integracionih shema (na primerima iz literature) dati su u tabelama 6.3 i 6.4 (pod shemom TECTON podrazumevamo shemu opisanu u poglavlju 5.2.3).[4] Tabela 6.3 prikazuje eksperimentalne rezultate za primere urađene u poglavlju 5.2.[5]

Tabela 6.4 prikazuje neke eksperimentalne rezultate poređenja različitih implementiranih shema. Iako shema u stilu Shostakove procedure nije implementirana korišćenjem specijalnih strukutura podataka (kao *find* i *use*) i optimizovana kao originalna procedura, dobijeni eksperimentalni rezultati se slažu sa ranijim rezultatima o poređenju Nelson/Oppenove i Shostakove sheme; naime, primećeno je da je Shostakova shema generalno znatno efikasnija od Nelson/-Oppenove (videti i [32]). Takođe, ona je generalno efikasnija od sheme u stilu TECTON (kada su obe primenljive). Primer 6 ilustruje nepotpunost originalne Shostakove procedure i njegove varijante u predloženom opštem okviru. Primer 7 pokazuje da se shema u stilu Shostakove procedure zaustavlja za tvrđenje za

[3] Ovim se takođe pokazuje da predloženi opšti okvir odnosne odgovarajuće sheme mogu da imaju domen (mada ne uvek i njihovu efikasnost) više različitih shema samo na osnovu malog broja implementiranih procedura.

[4] Testovi su vršeni na računaru PC Pentium 466Mhz pod operativnim sistemom Linux. CPU vreme je dato u sekundama.

[5] Razlike u CPU vremenima potrebnim za dokazivanje istih teorema u instancama EPM sheme opisanim u prethodnom poglavlju i u EPM shemi potiče od nešto različitih domena, u određenoj meri različitih mehanizama za korišćenje lema i u različitim uređenjima nad termovima.

1	Shema:	Nelson/Oppen
	Tvrđenje:	$x \leq y \wedge y \leq x + car(cons(0, x)) \wedge p(h(x) - h(y)) \wedge \neg p(0)$
	Teorije:	PRA, liste, čista teorija jednakosti
	CPU vreme	0.58s
2	Shema:	Shostak
	Tvrđenje:	$z = f(x - y) \wedge x = y + z \wedge \neg(y + f(f(z)) = x)$
	Teorije:	jednakosna realna aritmetika
	CPU vreme:	0.17s
3	Shema:	TECTON
	Tvrđenje:	$p(a) \wedge l \leq f(max(a, b)) \wedge 0 < min(a, b) \wedge a \leq max(a, b) \wedge$ $max(a, b) \leq a \wedge \neg(l < g(a) + b)$
	Lema:	$p(x) \Rightarrow f(x) \leq g(x)$
	Lema:	$max(x, y) = x \Rightarrow min(x, y) = y$
	CPU vreme:	2.39s
4	Shema:	EPM
	Tvrđenje:	$l \leq minl(\alpha) \wedge 0 < k \wedge maxl(\alpha) + k \leq l$
	Teorija $\mathcal{T}$:	PRA
	Procedura A:	Hodesova procedura
	Lema:	$minl(\xi) \leq maxl(\xi)$
	CPU vreme:	0.14s

Tabela 6.3: Rezultati kombinacionih/integracionih shema implementiranih u predloženom opštem okviru

koje se originalna procedura ne zaustavlja. Slaba tačka sheme u EPM stilu je jednakosno rezonovanje (slično kao u Boyer/Mooreovoj proceduri za linearnu aritmetiku) i to je potvrđeno dobijenim rezultatima. Bez aksioma supstitucije kao dodatnih pravila prezapisivanja, ova shema može da dokaže samo veoma jednostavna tvrđenja sa neinterpretiranim funkcijskim simbolima koji zahtevaju jednakosno rezonovanje (primer 7). Dodatno, ova shema je bila neuspešna za primer 8 koji zahteva upotrebu lema, ali i jednakosno rezonovanje. S druge strane, shema u EPM stilu je bila efikasnija od TECTON sheme na primerima u kojima jednakosno rezonovanje nije potrebno ili nije korisno (primeri 9, 10, 11). U primeru 11, za razliku od instanci originalne EPM sheme, nisu dobijane ogromne formule i to ilustruje činjenicu da svođenje tvrđenja na konjunkcije literala čini sheme robusnijim. CPU vremena originalne implementacije EPM sheme i implementacije u generalnom okviru razlikuju se na istim tvrđenjima zbog razlika između ove dve implementacije (npr. u uređenju nad termovima, strukturi dokaza, mehanizmu za korišćenje lema itd; na primer, u primeru 11 originalna EPM shema efikasnija je od sheme implementirane u opštem okviru zbog jednostavnijeg uređenja koje se intenzivno koristi).

Tabela 6.4 ilustruje i činjenicu da za svaku shemu postoje tvrđenja za koje je ta shema najuspešnija. Drugim rečima, ona ilustruje činjenicu da je dobro imati na raspolaganju više kombinacionih/integracionih shema u jednom dokazivaču i birati između njih na osnovu određenih heurističkih skorova (koji bi mogli biti dinamički podešavani u skladu sa teoremama koje su već uspešno dokazane [78]).

#	Tvrđenje\Shema	N/O	Shostak	TECTON	EPM
1	$x \leq y, y \leq x + car(cons(0,x)), p(h(x) - h(y)), \neg p(0)$ [86]	0.58	n/i	?	?
2	$z = f(x - y) \wedge x = y + z \wedge \neg(y + f(f(z)) = x)$ [104]	0.18	0.17	0.40	?
3	$x = y, f(f(f(x))) = f(x), \neg(f(f(f(f(f(y))))) = f(x))$ [32]	0.01	0.01	0.03	?
4	$y = f(z), x - 2 \cdot y = 0, \neg(f(x - y) = f(f(z)))$ [32]	0.16	0.10	0.14	?
5	$f(x) = 4, f(2 \cdot y - x) = 3, x = f(2 \cdot x - y),$ $4 \cdot x = 2 \cdot x + 2 \cdot y$ [32]	0.62	0.17	0.48	?
6	$f(a-1) - 1 = a + 1, f(b) + 1 = b - 1, b + 1 = a$ [100]	0.85	?	0.76	?
7	$f(a) = a, f(b) = b - 1, a = b$ [100]	0.03	0.02	0.06	0.03
8	$p(a), l \leq f(max(a,b)), 0 < min(a,b), a \leq max(a,b)$ $max(a,b) \leq a, \neg(l < g(a) + b)$ [68] lemma: $p(x) \Rightarrow f(x) \leq g(x)$ lemma: $max(x,y) = x \Rightarrow min(x,y) = y$	?	?	2.39	?
9	$l \leq minl(\alpha) \wedge 0 < k \wedge maxl(\alpha) + k \leq l$ [17] lemma: $minl(\xi) \leq maxl(\xi)$	?	?	0.26	0.14
10	$lp + lt \leq maxint, i \leq lt, \neg(i + \delta(pat, lp, c) \leq maxint)$ [17] lemma: $\delta(x,y,z) \leq y$	?	?	0.34	0.23
11	$(ms(c) + (ms(a) \cdot ms(a)) + ms(b) \cdot ms(b) <$ $((ms(c) + (ms(b) \cdot ms(b))) + (2 \cdot (ms(a) \cdot (ms(a) \cdot ms(b))))$ $+(ms(a) \cdot (ms(a) \cdot (ms(a) \cdot ms(a)))))$ [17] lemma: $0 < i \Rightarrow j < i \cdot j$ lemma: $0 < ms(x)$	?	?	6.71	5.73

Tabela 6.4: Poređenje različitih kombinacionih/integracionih shema implementiranih u predloženom opštem okviru ('?' označava neuspeh sheme da dokaže, tj. opovrgne tvrdjenje; *n/i* znači "nije implementirano")

Svi dobijeni rezultati potvrđuju da je moguće napraviti okvir za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema koji je strogo opisan, ali fleksibilan i efikasan.

6.4 Implementation and Results: Summary

We have made an implementation of the EPM scheme (and some of its instances) in SWI–PROLOG, within the proof–planning system *Clam* [18] (proof planning is a technique for guiding the search for a proof in automated theorem proving; to prove a conjecture, proof planning first construct the proof plan for a proof and then uses it to guide the construction of the proof itself.) We have implemented corresponding methods for all general EPM simplification procedures and corresponding methods for theory specific procedures for PIA (based on Hodes' algorithm), PIA and PNA (based on Cooper's algorithm). These procedures are implemented in a flexible way, based on the applications of a series of normalisations. We have not implemented the corresponding tactics. The reasons for this are the following: (*i*) in EPM scheme, neither of needed tactics would ever fail; this moves the whole search process to the methods level and generating

object-level proofs is straightforward (*ii*) more importantly, EPM simplification procedures are based on classical logic, so it is not always possible to interpret them in intuitionistic logic of *Oyster*. We have used this implementation and proved successfully a number of conjectures from the literature.

We have also implemented the general setting for combining and integrating decision procedures and some specific schemes. We have implemented them in SWI–PROLOG, within the proof–planning system *Clam* (we used its rewriting mechanism, including recursive path ordering with status). We have implemented the macro inference rules and the combination/integration schemes as new predicates which extend the standard *Clam*'s method language, while in the future work they might be implemented as methods. We have implemented the following schemes: Nelson/Oppen's, Shostak's, TECTON and EPM style schemes. We have used these schemes and proved a number of theorems from the literature and the obtained results are satisfactory (all conjectures were proved in only seconds or parts of seconds).

Both results for EPM scheme and for the general setting suggests that it is sensible to have more combination/integration schemes available in one theorem prover (while their ordering could be based on some heuristic scores and adjusted dynamically).

All obtained results are encouraging, in the sense that they show it is possible to make a framework for combining and integrating decision procedures into theorem provers which is formal, flexible and efficient.

The proposed frameworks are well-suited to the proof-planning paradigm, but can be also used in theorem provers based on some other paradigms.

Glava 7

Pravci daljeg rada

Nekoliko problema koji su rešavani ili samo pomenuti u ovoj tezi zahteva dalje razmatranje. Ukratko ćemo opisati probleme kojima planiramo da se bavimo u daljem radu.

- Planiramo da u potpunosti integrišemo implementaciju predloženog opšteg okvira u sistem *Clam*. Makro pravila izvođenja bila bi implementirana kao metodi zasnovani na novim predikatima koji proširuju standardni jezik metoda sistema *Clam*. Makro pravila izvođenja implementirana kao metodi služiće za građenje nadmetoda za kombinovanje i integrisanje procedura odlučivanja. Sistem za planiranje dokaza *Clam* treba da bude proširen mehanizmom koji podržava biblioteku signatura teorija. Taj mehanizam ne be trebalo samo da doda potrebne predikate (npr. za proveravanje da li neka formula pripada nekoj teoriji ili uniji teorija, različite forme apstrakcije itd) u standardni jezik metoda, veće i da pruža informacije o teorijama i raspoloživim odgovarajućim procedurama. Teorije bi bile reprezentovane svojim signaturama, svojim svojstvima (npr. konveksna, kanonizabilna, dopušta eliminaciju kvantifikatora itd) i raspoloživim primitivnim procedurama (npr. proveravanje nezadovoljivosti, funkcija *solve*, eliminacija promenljivih itd); primenljivost nekog nadmetoda — neke kombinacione/integracione sheme — bila bi ispitivana u terminima svojstava raspoloživih teorija.

- U predloženom opštem okviru za kombinovanje i integrisanje teorema jednostavno je dokazati saglasnost svake opisane sheme. Pokušaćemo da proširimo opšti okvir tako da bude moguće dokazati zaustavljanje više (ili svih) shema na uniforman način. Takvo proširenje bilo bi zasnovano na novim zahtevima postavljenim za sva makro pravila izvođenja i ono bi služilo kao osnova za neko dobro zasnovano uređenje nad formulama. Razmatraćemo i još znatno teži problem dokazivanja potpunosti za više shema istovremeno. Dodatno, razmatraćemo i sledeći problem: kada se uporede međusobno, sheme implementirane u opštem okviru, imaju veoma slične strukture. Štaviše, čini se da mogu da budu napravljene određene

male izmene kako bi ove strukture bili jasno uporedive u smislu da su slična pravila na istim ili sličnim pozicijama. Dalje, to znači da bi bilo moguće nametnuti neki fiksni poredak nad pravilima i da sheme (i one koje su ovde opisane i nove) mogu da budu kreirane jednostavnim biranjem i (de)aktiviranjem određenih pravila na njihovim fiksnim pozicijama.

- Istražićemo mogućnosti za potpuno formalno opisivanje predloženih makro pravila izvođenja koja čine opšti okvir, kao i za ispitivanje preciznih veza između njih i određenih računa (na primer, [122, 6, 7]) ili nekih specifičnih pravila izvođenja (na primer, [123, 14, 124]).

- Efikasna implementacija kombinacione/integracione sheme zasniva se na efikasnosti okvira u kojem je implementirana, ali i na efikasnosti specifičnih procedura odlučivanja koje se koriste. Optimizacije svih specifičnih procedura su svakako korisne za ukupnu efikasnost sistema. Dodatno, kada je raspoloživo više procedura odlučivanja za jednu istu teoriju, potrebno je napraviti izbor i, štaviše, potrebno je praviti poseban izbor za svako pojedinačno zadato tvrđenje. Analiza najgoreg slučaja često može da zavara, nije mnogo informativna i ne može mnogo da pomogne u rešavanju ovog problema. Umesto toga, potrebna je određena veza između sintaksnih svojstava tvrđenja i očekivane složenosti izračunavanja za raspoložive procedure odlučivanja. Postoji obilje rezultata u ispitivanju odnosta između sintaksnih svojstava i prosečne složenosti izračunavanja za probleme zadovoljivosti u iskaznoj logici (SAT) [24, 48, 83, 46] i bilo bi korisno ako bi slični rezultati postojali i za neke druge teorije. Na primer, bilo bi interesantno ispitati ponašanje Cooperovog i Hodesovog algoritma za PIA tvrđenja kada se neki sintaksni parametri menjaju. Rezultati ovog tipa bili bi upotrebljivi u izboru između različitih raspoloživih procedura odlučivanja kada ih je više primenljivo.

- Ako je, u nekom dokazivaču i za neko tvrđenje, primenljivo više kombinacionih/integracionih shema, jedna od njih treba da bude izabrana i primenjena. Ovaj problem je analogan problemu za procedure specifične za pojedinačne teorije, ali jezik u ovom problemu može da bude mnogo bogatiji. Dakle, čini se da je veoma teško utvrditi neku jednostavnu relaciju između sintaksnih svojstava i prosečne složenosti izračunavanja za tvrđenja iz tako širove klase. Umesto toga, određeni heurističko skorovi treba da budu upotrebljeni za poredak primenljivih shema (videti, na primer, [78]). Trebalo bi da ti skorovi mogu da se dinamički podešavaju u skladu sa teoremama koje su već dokazane.

7.1 Further work

There are several questions being answered or just tackled in this thesis which need further investigations. We briefly discuss problems we are planning to address in future work.

- We are planning to further and fully integrate the implementation of the proposed general setting within the proof-planning system *Clam*. The macro inference rules would be implemented as methods based on a number of new predicates which extend the standard *Clam*'s method language. The macro inference rules implemented as methods would serve as a basis for building different supermethods for combining and integrating decision procedures. Proof–planning system *Clam* should be extended by a mechanism for theories' signatures library. It should not only add needed predicates (e.g. checking whether a formula belongs to some theory, or to union of some theories, different forms of abstraction etc) into the standard method language, but should also provide information about theories and associated procedures available. Theories would be represented by their signatures, by their properties (e.g. convex, canonizable, admits quantifier elimination etc) and by available primitive procedures (e.g. checking unsatisfiability, solve, variable elimination etc); applicability of some supermethod — some combination/integration scheme would be interpreted in terms of available theories.

- Within the proposed general framework for combining and integrating decision procedures it is easy to prove soundness of all schemes described within it. We will try to extend the general setting in such a way that it would possible to prove the termination for more (or even all) schemes uniformly. Such an refinement would be based on some further properties imposed on all macro inference rules which would serve as a basis for some well–founded ordering of formulae. Proving completeness for several schemes at the same time seems also an interesting and challenging problem. In addition, we will investigate the following issue: merged with each other, the schemes implemented within the proposed general setting have rather similar structures. Moreover, it seems that some slight changes can be made in order to have these structures clearly comparable in the sense that similar rules are in the same or similar positions. Further, it might be possible to impose some fixed ordering on the rules and that schemes (both presented here and new ones) can be simply created by choosing and (de)activating certain rules at their fixed positions.

- We will look at possibilities for fully formal describing the proposed macro inference rules and for precise relationships between them and some calculi (such as calculi discussed in [122, 6, 7]) or some other specific inference rules (such as discussed in [123, 14, 124]).

- An efficient implementation of a combination/integration scheme relies on the efficiency of the framework it is implemented within, but also on the efficiency of the theory specific decision procedures used. Optimisations of all theory specific procedures is fruitful to the efficiency of the overall system. In addition, when there are more than one decision procedure for some theory available, a choice has to be made and, moreover, a separate choice should be made for each conjecture. Worst case analysis is

often misleading, not very informative and cannot be of much help in handling this problem. Instead, we need some relation between syntactical properties of the conjecture and the expected computational cost for some specific decision procedure. There are plenty of results in investigation of relations between syntactical properties and average computational cost for satisfiability problems in propositional logic (SAT) [24, 48, 83, 46] and it would be helpful to have similar results for some other theories as well. For instance, it would be interesting to investigate the behaviour of Cooper's and Hodes' algorithms on PIA conjectures as some syntactical parameters vary. Results of this kind could help in choosing between different available decision procedure when more than one is applicable.

- If, in some theorem prover and for some conjecture, there are more combination/integration schemes applicable, a choice has to be made. This problem is related to the analogous problem for theory specific procedures, but a language can be much more expressive than in the latter case. Therefore, it seems unlikely that one can detect some simple relation between syntactical properties and average computational cost for conjecture from such large class. Instead, some heuristic scores can be used for the ordering of applicable schemes. (see, for instance, [78]). Such scores should be dynamically adjusted according to theorems already successfully proved.

Glava 8

Zaključci

Uloga procedura odlučivanja je često veoma značajna u dokazivanju teorema. Međutim, u nekim primenama, samo mali broj tvrđenja pripada domenu jedne procedure odlučivanja, ali je, u određenom smislu, "blizu" tog domena. U takvim situacijama potrebno je koristiti leme, procedure odlučivanja moraju da komuniciraju međusobno ili sa drugim komponentama dokazivača itd. Postoji nekoliko dominantnih pristupa problemu efikasnog kombinovanja i ugradnje procedura odlučivanja u dokazivače teorema. Istraživanja koja se odnose na ove probleme uglavnom su išla u dva pravca: jedan koji se odnosi na kombinovanje procedura odlučivanja [86, 87, 104, 32, 9, 11, 100, 10] i jedan koji se odnosi na integrisanje procedura odlučivanja u heurističke dokazivače teorema [17, 68, 69, 2, 59, 4].

Za jednu shemu za kombinovanje/integraciju procedura odlučivanja poželjno je:

- da bude opisana na apstraktan, formalan način; s jedne strane, to treba da dâ jasan opis sheme i olakša njeno implementiranje, a s druge strane, da omogući formalno rezonovanje o jednoj shemi (razmatranje zaustavljanja, saglasnosti, potpunosti);
- da bude fleksibilna i da omogućava jednostavnu zamenu teorija i procedura odlučivanja na koje se odnosti (što treba da obezbedi široku primenljivost te sheme).

Na fleksibilno kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema odnosi se nekoliko nedavno objavljenih pristupa [2, 10].

U ovoj tezi predlažemo jedan fleksibilan okvir za integrisanje jedne klase procedura odlučivanja u dokazivače teorema (EPM shemu) i jedan opšti okvir za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema. EPM shema je opšti metod za fleksibilno integrisanje procedura odlučivanja za teorije koje dopuštaju eliminaciju kvantora. Ona je delimično zasnovana na Boyer/Mooreovoj proceduri za linearnu aritmetiku, ali je znatno strožije opisana, opštija i fleksibilnija. Specifično znanje je sadržano u manjim podmodulima i jedna procedura odlučivanja može biti jednostavno upotrebljena ako

pruža ograničen skup funkcionalnosti kao što su proveravanje zadovoljivosti i eliminacija promenljivih. U ovoj tezi takođe predlažemo opšti okvir za opisivanje i implementiranje različitih shema za kombinovanje i integrisanje procedura odlučivanja u dokazivače teorema. Ovaj okvir zasnovan je na makro pravilima izvođenja motivisanim poznatim pristupima. Neka od njih su apstrakcija, kongruentno zatvorenje, korišćenje lema itd. Korišćenjem ovog opšteg okvira, opisali smo nekoliko različitih procedura za kombinovanje i integrisanje procedura odlučivanja. Opšti okvir daje ključne ideje jedne kombinacione/integracione sheme, čini različite sheme uporedivim i omogućava formalno rezonovanje o njima. Sva makro pravila izvođenja su saglasna pa su, prema tome, sve sheme implementirane u opštem okviru saglasne. Sva makro pravila izvođenja su zaustavljajuća, ali zaustavljanje različitih kombinacionih/integracionih shema mora da bude dokazivano posebno. Za neke od implementiranih shema može se pokazati da su potpune (npr. za Nelson/Oppenovu shemu). Sheme implementirane primenom predloženog opšteg okvira su najšće manje efikasne od izvornih procedura (u nekima od kojih su primenjene specifične optimizacije), ali verujemo da dobici na osnovu fleksibilnosti i formalnog opisa prevazilaze te gubitke u efikasnosti.

U okviru teze dat je opis implementacije EPM sheme (i neke od njenih instanci), kao i opis implementacije opšteg okvira (i nekih shema). Tako implementiranim shemama dokazali smo veći broj tvrđenja i dobijeni rezultati su veoma dobri. U daljem radu planiramo dalje unapređenje ovih implementacija.

Doprinos ove teze ima sledeća tri aspekta:

metodološki: predloženi opšti okviri daju novu metodologiju za kombinovanje i ugradnju procedura odlučivanja; oni su zasnovani na modularnosti, fleksibilnosti i apstraknom opisu shema za kombinovanje i ugradnju procedura odlučivanja;

teorijski: strogi opis shema za kombinovanje i ugradnju procedura odlučivanja opisanim u predloženim opštim okvirima omogućava formalno rezonovanje o tim shemama; za sve sheme implementirane u predloženim opštim okvirima dokazana je saglasnost, dok je za neke sheme dokazano i zaustavljanje i potpunost.

praktični: predloženi opšti okviri i scheme omogućavaju implementiranje različitih metoda na osnovu malog broja bazičnih procedura; najveći deo kôda za različite sheme je zajednički, pa je moguće imati više raspoloživih shema na bazi relativno malo kôda. Dodatno, u tezi su dati i praktični rezultati koji se odnose na efikasniju, odnosno fleksibilniju implementaciju procedura odlučivanja za Prezburgerovu aritmetiku.

8.1 Conclusions

The role of decision procedures is very important in theorem proving. However, in some applications only a small number of conjectures fall within the scope of

one decision procedure. In these situations some additional knowledge is needed and lemmas have to be invoked, decision procedures have to communicate with each other or with the heuristic component of a theorem prover. There are several dominating approaches used in handling the problem of efficient combining and integrating decision procedures into theorem provers. The research concerning these issues have gone mostly along different two lines: one concerning combining decision procedures [86, 87, 104, 32, 9, 11, 100, 10] and one concerning integrating decision procedures into heuristic theorem provers [17, 68, 69, 2, 59, 4].

For one combination/integration scheme it is plausible:

- to be described in an abstract, formal form; on one hand, it should provide a clear description and enable implementing the scheme, and, on the other hand, it should enable formal reasoning about the scheme (and considering its termination, soundness, completeness);

- to be flexible and providing the possibility of easily changing underlying theories and decision procedures (which should provide wider applicability of the scheme).

Flexible combining and integrating decision procedures into theorem provers is addressed by several recent approaches [2, 10].

In this thesis we propose one flexible framework for the integration of one class of decision procedures into theorem provers (EPM scheme) and one general setting for both combining and integrating decision procedures into theorem provers. EPM scheme is a general method for the flexible integration of decision procedures for theories which admit quantifier elimination. It is partly based on Boyer/Moore linear arithmetic procedure, but is more general and more flexible. Specific knowledge is encapsulated in smaller submodules and decision procedures can be simply 'plugged-in' to the system only if it provides a limited number of functionalities such as checking satisfiability and variable elimination. In this thesis we also propose a general setting for describing and implementing different schemes for both combining and integrating decision procedures into theorem provers. This setting is based on macro inference rules motivated by the approaches known from the literature. Some of them are abstraction, entailment, congruence closure, lemma invoking etc. The general setting gives the key ideas of one combination/integration scheme, makes different schemes comparable and enables the formal reasoning about them. All macro inference rules are sound and, therefore, all schemes implemented within our general setting are sound. All macro inference rules are terminating, but termination of different combination/integration schemes has to be proved separately. For some of these schemes it can be proved they are complete (e.g. for Nelson/Oppen's scheme). Concerning efficiency, procedures obtained in this way (in this setting) would be less efficient than some tightly integrated procedures, but we believe that gains from the flexibility would overcome the potential losses in efficiency.

We have implemented EPM scheme (and some of its instances) and the proposed general setting (and some schemes within it). By these implementations

we proved a number of conjectures with good results. In future work we will work on further refinements of these implementations.

The contribution of the research reported in this thesis has three aspects:

methodological: the proposed frameworks give new methodology for combining and integrating decision procedures; they are based on modular and flexible architecture and abstract description of schemes for combining and integrating decision procedures;

theoretical: formal description of combining and integrating decision procedures within the proposed frameworks enables formal reasoning about schemes implemented within them; for all implemented schemes we proved soundness and for some of them we proved termination and completeness;

practical: the proposed frameworks enable implementing of different schemes on the basis of only several basic procedures; most of the code is shared and a number of different combination/integration schemes is available on the basis of only small amount of code; in addition, the thesis gives some practical results about more efficient or more flexible implementation of decision procedures for Presburger arithmetic.

Dodatak A

Odlučivost i procedure odlučivanja

A.1 Odlučivost

Postoji više formalizama kojima se uvodi pojam izračunljivosti (URM mašine, Turingove mašine, Postove mašine, rekurzivne funkcije, Markovljevi algoritmi) i za njih se može dokazati da su ekvivalentni. Churchova teza tvrdi da je klasa intuitivno, neformalno izračunljivih funkcija identična sa tim, strogo zasnovanim klasama izračunljivih funkcija. U daljem tekstu, pojam odlučivih teorija biće strogo uveden na bazi rekurzivnih funkcija (a intuitivno na bazi pojma efektivno izračunljivih funkcija), pa će taj formalizam biti opisan.

A.1.1 Izračunljivost

Za funkciju $f : \mathbf{N}^{n+1} \to \mathbf{N}$ kažemo da je dobijena *primitivnom rekurzijom* od funkcija g i h (čije su arnosti redom n i $n+2$) ako važi:

$$f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n)$$

$$f(x_1, x_2, \ldots, x_n, y+1) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y))$$

i pišemo $f = Rec(g; h)$.

Neka je $x = (x_1, x_2, \ldots, x_n)$, neka su g_1, g_2, ..., g_k funkcije iz $\mathbf{N}^n$ u $\mathbf{N}$ i neka je h funkcija iz $\mathbf{N}^k$ u $\mathbf{N}$. Za funkciju f iz $\mathbf{N}^n$ u $\mathbf{N}$ kažemo da je dobijena *supstitucijom* (slaganjem) od funkcija h i g_i $(i = 1, \ldots, n)$ ako je definisana na sledeći način:

$$f(x) = h(g_1(x), g_2(x), \ldots, g_k(x))$$

i pišemo $f = Sub(h; g_1, g_2, \ldots, g_k)$.

Minimizaciju definišemo na sledeći način:

$$\mu y[g(x_1, x_2, \ldots, x_n, y) = 0] =$$

$$\stackrel{def}{=} \begin{cases} y & \text{gde je } y \text{ najmanja vrednost takva da je } g(x_1, x_2, \ldots, x_n, y) = 0, \\ & \text{ako takva postoji i ako je vrednost } g(x_1, x_2, \ldots, x_n, z), \\ & \text{definisana za sve vrednosti } z \leq y \\ \text{nedefinisano} & \text{inače} \end{cases}$$

Skup *primitivno rekurzivnih funkcija* je skup koji zadovoljava sledeća svojstva:

1. Nula funkcija $z : \mathbf{N} \to \mathbf{N}$, sledbenik funkcija $z : \mathbf{N} \to \mathbf{N}$ i funkcija projekcije $P_i^n : \mathbf{N}^n \to \mathbf{N}$ $(1 \leq i \leq n)$ koje zadovoljavaju uslove

$$z(x) = 0$$

$$s(x) = x + 1$$

$$P_i^n(x_1, x_2, \ldots, x_n) = x_i$$

   su primitivno rekurzivne funkcije.

2. Ako su funkcije $g_1, g_2, \ldots, g_k : \mathbf{N}^n \to \mathbf{N}$ i $h : \mathbf{N}^k \to \mathbf{N}$ primitivno rekurzivne, onda je i $Sub(h; g_1, g_2, \ldots, g_k)$ primitivno rekurzivna funkcija.

3. Ako su funkcije g i h primitivno rekurzivne funkcije, onda je i $Rec(h; g)$ primitivno rekurzivna funkcija.

4. Primitivno rekurzivne funkcije su samo one koje se mogu dobiti konačnom primenom prethodnih pravila.

Skup *rekurzivnih funkcija* je skup koji zadovoljava sledeća svojstva:

1. Svaka primitivno rekurzivna funkcija je rekurzivna funkcija.

2. Ako su funkcije $g_1, g_2, \ldots, g_k : \mathbf{N}^n \to \mathbf{N}$ i $h : \mathbf{N}^k \to \mathbf{N}$ rekurzivne, onda je i $Sub(h; g_1, g_2, \ldots, g_k)$ rekurzivna funkcija.

3. Ako su funkcije g i h rekurzivne funkcije, onda je i $Rec(h; g)$ rekurzivna funkcija.

4. Ako je $g : \mathbf{N}^{n+1} \to \mathbf{N}$ rekurzivna funkcija onda je rekurzivna i funkcija $\mu y[g(x_1, x_2, \ldots, x_n, y) = 0]$.

5. Rekurzivne funkcije su samo one koje se mogu dobiti konačnom primenom prethodnih pravila.

Skup *totalnih rekurzivnih funkcija* je skup rekurzivnih funkcija koje su definisane za svaku torku svojih argumenata.[1]

Postoje funkcije koje su totalne rekurzivne, a nisu primitivno rekurzivne (npr. Ackermanova funkcija [111]). Postoje funkcije koje su rekurzivne, a nisu totalne rekurzivne funkcije (npr. funkcija f koja nije totalna, tj. za koju postoji vrednost a za koju $f(a)$ nije definisano).

[1]Neki autori zahtevaju u definiciji totalnih rekurzivnih funkcija da sve funkcije koje učestvuju u njihovoj konstrukciji takođe budu totalne.

Na osnovu Churchove teze, rekurzivne funkcije odgovaraju intuitivno izračunljivim funkcijama, dok totalne rekurzivne funkcije odgovaraju efektivno izračunljivim funkcijama.

Za podskup A skupa prirodnih brojeva kažemo da je *rekurzivan* (ili *odlučiv*) ako je njegova karakteristična funkcija $f : \mathbf{N} \to \mathbf{N}$:

$$f(x) = \begin{cases} 1, & \text{ako je } x \in A \\ 0, & \text{ako je } x \notin A \end{cases}$$

rekurzivna.

Za podskup A skupa prirodnih brojeva kažemo da je *polurekurzivan* (ili *rekurzivno nabrojiv* ako je funkcija

$$f(x) = \begin{cases} 1, & \text{ako je } x \in A \\ \text{nedefinisano}, & \text{ako je } x \notin A \end{cases}$$

rekurzivna.

Svaki rekurzivan skup je i rekurzivno nabrojiv, ali obratno ne važi.

A.1.2 Gedelizacija

Svaki prebrojiv skup moguće je preslikati u skup prirodnih brojeva. *Gödelova funkcija* predstavlja osnovu mehanizma za kodiranje formula prirodnim brojevima i njihovo dekodiranje. Smisao gedelizacije je u tome da se odlučivost teorije razmatra u terminima rekurzivnosti skupa svih teorema.

Neka je $\mathcal{L} = \langle T, V, \Sigma, \Pi, d \rangle$ jezik prvog reda. Funkcija g_s koja skup svih simbola jezika $\mathcal{L}$ preslikava u skup prirodnih brojeva je Gödelova funkcija ako je ona bijektivna i ako su skupovi $g_s(V)$, $g_s(\Sigma)$, $g_s(\Pi)$ rekurzivni. Funkcija g_e koja preslikava niz simbola jezika $\mathcal{L}$ u skup prirodnih brojeva je Gödelova funkcija (u skladu sa funkcijom g_s) ako niz simbola s_1, s_2, ..., s_n (u tom poretku) preslikava u broj

$$\prod_{i=1}^{n} pn(i)^{1+g_s(s_i)} ,$$

gde je $pn(i)$ i-ti prost broj. Može se pokazati da je ovako definisana funkcija g_e i injektivna, tj. različitim nizovima simbola odgovaraju različiti kôdovi. Skup kôdova svih rečenica jezika $\mathcal{L}$ je rekurzivan podskup skupa $\mathbf{N}$.

Jezik $\mathcal{L}$ kojem je na opisani način pridružena funkcija g_e nazivamo *efektivizovanim jezikom* i označavamo sa $\hat{\mathcal{L}} = \langle T, V, \Sigma, \Pi, d, g_e \rangle$. Kôdove $g_s(s)$ i $g_e(\phi)$ označavamo najčešće sa $\lceil s \rceil$ i $\lceil \phi \rceil$.

A.1.3 Odlučive teorije

Neka je $\mathcal{T}$ teorija prvog reda efektivizovanog jezika $\hat{\mathcal{L}}$ i g_e njegova Gödelova funkcija. Za teoriju $\mathcal{T}$ kažemo da je *odlučiva*[2] (ili *razrešiva*) ako je skup $\{\lceil \phi \rceil \mid \phi \in$

[2]Termin *odlučiv* ima sasvim drugačiji smisao kada je u pitanju pojedinačna formula: kažemo da je formula ϕ neprotivrečne teorije $\mathcal{T}$ *odlučiva* ako je ili ϕ ili $\neg\phi$ teorema teorije $\mathcal{T}$. Ako je teorija potpuna, sve rečenice na njenom jeziku su odlučive. Teorija može da bude nepotpuna i odlučiva.

$\mathcal{T}\}$ rekurzivan, tj. ako je karakteristična funkcija teorije $\mathcal{T}$ $f_{\mathcal{T}} : \mathbf{N} \to \mathbf{N}$:

$$f_{\mathcal{T}}(\lceil\phi\rceil) = \begin{cases} 1, & \text{ako je } \phi \in \mathcal{T} \\ 0, & \text{ako je } \phi \notin \mathcal{T} \end{cases}$$

totalna rekurzivna. Funkciju $f_{\mathcal{T}}$ tada nazivamo i *procedurom odlučivanja* za teoriju $\mathcal{T}$[3]. Za teoriju $\mathcal{T}$ kažemo da je *esencijalno neodlučiva* ako je neodlučiva teorija $\mathcal{T}$, kao i bilo koje njeno neprotivrečno proširenje. Za teoriju $\mathcal{T}$ kažemo da je *neodlučiva* ako nije odlučiva.

Za teoriju $\mathcal{T}$ kažemo da je *poluodlučiva* ako je skup $\{\lceil\phi\rceil \mid \phi \in \mathcal{T}\}$ rekurzivno nabrojiv, tj. ako je funkcija $h_{\mathcal{T}} : \mathbf{N} \to \mathbf{N}$:

$$h_{\mathcal{T}}(\lceil\phi\rceil) = \begin{cases} 1, & \text{ako je } \phi \in \mathcal{T} \\ \text{nedefinisano}, & \text{ako je } \phi \notin \mathcal{T} \end{cases}$$

rekurzivna. Funkciju $h_{\mathcal{T}}$ tada nazivamo i *procedurom poluodlučivanja* za teoriju $\mathcal{T}$.

Prethodnim definicijama strogo je uveden pojam odlučive teorije. Taj pojam moguće je uvesti i nešto neformalnije, intuitivno, na sledeći način: ukoliko postoji algoritam A (postupak, procedura) takav da za svaku rečenicu α daje odgovor *da* ako i samo ako je α teorema teorije $\mathcal{T}$ (i *ne* inače) onda kažemo da je teorija $\mathcal{T}$ odlučiva. Veza između formalno i neformalno uvedenog pojma odlučive teorije bazira se (između ostalog) i na Churchovoj tezi koja tvrdi da su klase rekurzivnih i efektivno izračunljivih funkcija identične.

Za jezik $\mathcal{L} = \langle T, V, \Sigma, \Pi, d\rangle$ kažemo da je rekurzivan, ako su skupovi T, V, Σ, Π rekurzivni. Za teoriju $\mathcal{T}$ nad rekurzivnim jezikom $\mathcal{L}$ kažemo da je *aksiomatibilna* ako postoji rekurzivan neprotivrečan skup A rečenica jezika $\mathcal{L}$ takav da je rečenica α teorema teorije $\mathcal{T}$ ako i samo ako važi $A \vdash \alpha$.

Naredna teorema daje potrebne i dovoljne uslove da potpuna teorija bude odlučiva [116]:

Teorema A.1 *Za potpunu teoriju $\mathcal{T}$ sledeći uslovi su ekvivalentni:*

- *$\mathcal{T}$ je neodlučiva,*
- *$\mathcal{T}$ je esencijalno neodlučiva,*
- *$\mathcal{T}$ nije aksiomatibilna.*

O odnosu odlučivosti za neku teoriju i njeno proširenje govori naredno tvrđenje:

Teorema A.2 *Neka su $\mathcal{T}_1$ i $\mathcal{T}_2$ dve teorije sa istim konstantama takve da je $\mathcal{T}_2$ konačno proširenje teorije $\mathcal{T}_1$. Ako je teorija $\mathcal{T}_2$ neodlučiva, onda je neodlučiva i teorija $\mathcal{T}_1$.*

[3]Ponekad se procedurom odlučivanja naziva funkcija koja je definisana samo za Gödelove kodove nizova simobla koji su formule jezika $\hat{\mathcal{L}}$. U skladu sa ovde navedenom definicijom, funkcija $f_{\mathcal{T}}$ najpre "proverava" da li je za datu vrednost argumenta $\lceil\phi\rceil$ niz simbola ϕ zaista formula teorije $\mathcal{T}$, a ako jeste, onda se proverava da li je ta formula teorema teorije $\mathcal{T}$.

U ispitivanju odlučivosti koristi se često i sledeća teorema [81]:

Teorema A.3 *Ako teorija $\mathcal{T}$ ima samo prebrojivo mnogo kompletnih proširenja i ako se ona mogu efektivno nabrojati, onda je ona odlučiva.*

Postoji značajna metodološka razlika u proučavanju odlučivosti i neodlučivosti, uprkos činjenici da su ta dva pojma u neposrednoj vezi. Da bi se pokazalo da je neka teorija neodlučiva, obično se poseže za strogim formalizmima uzračunljivih funkcija. S druge strane, za dokaz odlučivosti neke teorije dovoljno je postojanje efektivnog postupka koji za svaku rečenicu utvrđuje da li jeste ili nije teorema date teorije. Zbog toga, neki rezultati u vezi sa odlučivošću prethodili su uvođenju pojma rekurzivnih funkcija sredinom tridesetih godina, dok su se prvi rezultati u vezi sa neodlučivošću pojavili tek nakon toga. Više o odlučivim teorijama videti u [98]; više o neolučivim teorijama videti u [116].

Na kraju ovog poglavlja navedimo nekoliko značajnih odlučivih i neodlučivih teorija (za više detalja videti [81, 44]).

Neke od odlučivih teorija su (u zagradi je ime matematičara koji je dokazao odlučivost te teorije kao i godina):

1. Čist predikatski račun sa jednakošću (L. Löwenheim, 1915).
2. Teorija unarnih relacija (L. Löwenheim, 1915).
3. Teorija ekvivalencije (A. Janiczak, 1953).
4. Teorija unarne operacije (A. Ehrenfeucht, 1959).
5. Teorija linearnog uređenja (A. Ehrenfeucht, 1959).
6. Teorija gusto uređenih skupova (R. Vaught, 1954).
7. Teorija dobro uređenih skupova (A. Tarski, A. Mostowski, 1949).
8. Teorija diskretnog uređenja (E. Zakon, A.Robinson, 1960).
9. Teorija Bulovih algebri (A. Tarski, 1949).
10. Teorija slobodnih grupoida (A. I. Malcev, 1961).
11. Prezburgerova aritmetika (M. Presburger, 1929).
12. Teorija množenja prirodnih brojeva (A. I. .Malcev).
13. Teorija sabiranja ordinalnih brojeva (A. Erenfucht, 1957).
14. Teorija sabiranja kardinalnih brojeva skupova (A. Tarski, 1956).
15. Bulova algebra podskupova (T. Skolem, 1917).
16. Teorija slobodnih komutativnih semigrupa (A. Mostowski, 1949).
17. Teorija Abelovih grupa (W. Szmielew, 1949).
18. Teorija uređenih Abelovih grupa (Y. Gurevich, 1964).
19. Teorija cikličnih grupa (Ершов, Ю.Л., 1963).
20. Teorija p-grupa (Ершов, Ю.Л., 1963).
21. Teorija algebarski zatvorenih polja (A. Tarski, 1949).
22. Teorija realno zatvorenih polja (A. Tarski, 1949).
23. Teorija p-adskih polja (J. Ax, S. Kochen, 1965).
24. Elementarna euklidska geometrija (A. Tarski, 1949).
25. Elementarna hiperbolička geometrija (W. Schwabhauser, 1959).
26. Teorija klase konačnih polja (J. Ax, 1968).

Neke od neodlučivih teorija su:

1. Peanova aritmetika.

2. ZF — Zermelo–Fraenekelova teorija skupova.
3. Teorija celih brojeva.
4. Teorija racionalnih brojeva.
5. Teorija grupa.
6. Teorija prostih grupa.
7. Teorija prstena.
8. Teorija komutativnih prstena.
9. Teorija polja.
10. Teorija Teorija uređenih polja.
11. Teorija modularnih mreža.
12. Teorija distributivnih mreža.
13. Teorija komutativnih semigrupa.
14. Teorija parcijalnog uređenja.
15. Teorija dva linearna uređenja.
16. Teorija simetrične relacije.
17. Teorija dve ekvivalencije.
18. Projektivna geometrija.

A.1.4 Metode za dokazivanje odlučivosti

Tri osnovna metoda za dokazivanje odlučivosti neke teorija prvog reda su (opširniji pregled videti u [98, 81, 110]):

- metod eliminacije kvantora;
- model teoretski–metod;
- metod interpretacija.

Metod eliminacije kvantora je najstariji i u kontekstu automatskog dokazivanja teorema najznačajniji. On se zasniva na transformisanju date rečenice α u rečenicu β takvu da je $\mathcal{T} \vdash \alpha$ ako i samo ako važi $\mathcal{T} \vdash \beta$ i β pripada klasi rečenica za koje se može trivijalno proveriti da li su teoreme teorije $\mathcal{T}$. Ta transformacija najčešće predstavlja eliminaciju promenljivih iz formule α. Ovaj metod biće nam od posebnog značaja za formulisanje posebnih procedura za dokazivanje teorema, kako odlučivih tako i neodlučivih teorija. Zbog toga, ovom metodu poklanjamo posebnu pažnju u sledećem poglavlju.

U svojoj osnovnoj formi, model–teoretski metod zasniva se na teoremi A.3 i razmatranju rekurzivnog skupa aksioma A teorije $\mathcal{T}$. Model–teoretska svojstva se koriste ili u dokazivanju da je $\mathcal{T}$ potpuna teorija (odakle na osnovu teoreme A.1 sledi da je $\mathcal{T}$ odlučiva) ili za sistematično nabrajanje svih potpunih proširenja skupa A (u tom slučaju možemo da za svaku rečenicu α da znamo da li je $\mathcal{T} \cup \{\neg\alpha\}$ neprotivrečna i odatle da li važi $\mathcal{T} \vdash \alpha$).

Neka je $\mathcal{T}$ teorija jezika $\mathcal{L}$ i $\mathcal{T}'$ teorija jezika $\mathcal{L}'$. Metod interpretacija zasniva se na efektivnom preslikavanju t koje svakoj rečenici α jezika $\mathcal{L}$ pridružuje rečenicu $t(\alpha)$ jezika $\mathcal{L}'$ takvu da važi

$$\mathcal{T} \vdash (\alpha) \text{ ako i samo ako } \mathcal{T}' \vdash t(\alpha)\ .$$

Ako takvo preslikavanje postoji i ako je teorija $\mathcal{T}'$ odlučiva, onda je odlučiva i teorija $\mathcal{T}$. Naime, da bismo utvrdili da li važi $\mathcal{T} \vdash \alpha$ dovoljno je odrediti $t(\alpha)$ i ispitati da li važi $\mathcal{T}' \vdash t(\alpha)$. Obično interpretacija t uključuje model–teoretska razmatranja. Pokazuje se da modeli teorije $\mathcal{T}$ mogu biti izomorfno reprodukovani iz modela teorije $\mathcal{T}'$ relacijama definabilnim u $\mathcal{L}'$. Metod interpretacija se često koristi i za dokazivanje neodlučivosti teorija. Ako je teorija $\mathcal{T}'$ neodlučiva, i ako postoji opisano preslikavanje t, onda je neodlučiva i teorija $\mathcal{T}$.

A.2 Procedure odlučivanja

U ovom poglavlju biće reči o procedurama odlučivanja (u intuitivnom smislu) i nekim njihovim svojstvima. Od metoda za dokazivanje odlučivosti teorije za automatsko dokazivanje teorema od primarnog interesa je metod eliminacije kvantifikatora. Značaj tog metoda je, između ostalog, u tome što indukuje proceduru odlučivanja za teoriju o kojoj je reč. Naredno potpoglavlje posvećeno je ovom metodu. U narednom potpoglavlju diskutuje se složenost procedura odlučivanja.

A.2.1 Eliminacija kvantora

Naziv metoda — *eliminacija kvantora* — potiče od Tarskog[4] (1935) koji je dao nekoliko njegovih najznačjnih primena, dokazavši odlučivost nekoliko teorija: algebarski zatvorena polja (1949), realno zatvorena polja (1949), teorija dobrog uređenja (sa Mostowskim, 1949), Bulove algebre (1951) (više o ovom metodu videti u [71], [110]). Suštinski isti metod bio je primenjivan i pre toga za predikatski račun sa jednakošću (Lövenheim, 1915), gusto linearno uređenje bez krajeva (Langford, 1927), Prezburgerovu aritmetiku (Presburger, 1929), a kasnije, između ostalih, i za dokaz odlučivosti teorije Abelovih grupa (Szmielew, 1955)

Metod eliminacije kvantora za dokazivanje odlučivosti neke teorije ima i nekoliko "sporednih efekata":

- u nekim slučajevima, kada je teorija data aksiomatski, metod može da indukuje i dokaz potpunosti teorije.
- metod indukuje proceduru odlučivanja koja je efektivno opisiva i moguće ju je automatizovati.

Neformalno, smisao metoda eliminacije kvantora je u tome da se odredi skup "jednostavnih" formula jezika $\mathcal{L}$ takav da je svaka formula jezika $\mathcal{L}$ ekvivalentna nekoj jednostavnoj formuli. Termin "jednostavna formula" međutim često nije najpogodniji za različite primene ove metode, jer su formule dobijene njegovom primenom zapravo izuzetno složene. U primenama, termin "jednostavna formula" odnosiće se najčešće na formule bez kvantifikatora.

[4] Hilbert [51] ovaj metod naziva *metodom redukcije*.

Neka je $\mathcal{K}$ efektivno izabran skup formula jezika $\mathcal{L}$. Kažemo da teorija $\mathcal{T}$ *dopušta eliminaciju kvantora* do na klasu $\mathcal{K}$ ako i samo ako za svaku formulu ϕ jezika $\mathcal{L}$ postoji formula ψ iz skupa $\mathcal{K}$ takva da važi $\mathcal{T} \vdash \phi$ ako i samo ako $\mathcal{T} \vdash \psi$.

Ukoliko je za sve rečenice iz klase $\mathcal{K}$ moguće efektivno odrediti da li pripadaju teoriji $\mathcal{T}$, onda je teorija koja dopušta eliminaciju kvantora odlučiva. Pored toga, ako za svaku rečenicu ψ iz klase $\mathcal{K}$ važi $\mathcal{T} \vdash \psi$ ili $\mathcal{T} \vdash \neg\psi$, onda je teorija $\mathcal{T}$ potpuna. U većini primena, klasa $\mathcal{K}$ je skup formula bez kvantora.

Grubo, metod se (za dokazivanje odlučivosti teorije $\mathcal{T}$ jezika $\mathcal{L}$), dakle, generalno sastoji od sledećih koraka:

- bira se odgovarajući skup osnovnih formula jezika $\mathcal{L}$ i njihova bulovska kombinacija se proglasi klasom $\mathcal{K}$.
- pokaže se da za svaku formulu ϕ jezika $\mathcal{L}$ postoji formula ψ iz klase $\mathcal{K}$ takva da je $\mathcal{T} \vdash \phi$ ako i samo ako $\mathcal{T} \vdash \psi$.
- pokaže se da je za svaku formulu ψ iz $\mathcal{K}$ moguće efektivno ispitati da li važi $\mathcal{T} \vdash \psi$.

Primenu metoda ilustrovaćemo u potpoglavljima o Prezburgerovoj aritmetici (D.1, D.2).

A.2.2 Složenost izračunavanja

Razvrstavanje problema odlučivanja (problema koji mogu da imaju samo odgovore *da* i *ne*) na odlučive i neodlučive je polazna osnova, ali je za razne primene i previše gruba. Na primer, u kontekstu pitanja odlučivosti teorija može se, za neodlučive teorije, govoriti o stepenu nerazrešivosti. S druge strane, za odlučive teorije važno je, pogotovu iz perspektive automatskog dokazivanja teorema, pitanje složenosti pojedinih procedura odlučivanja za konkretnu teoriju. Pojam složenosti se obično vezuje za Tjuringovu mašinu (ili neki drugi formalizam) i opisuje resurse potrebne da bi neki problem bio rešen. Mere resursa izračunavanja koje oslikavaju složenost algoritma su *vreme* (odnosno broj koraka koje izvršava algoritam) i *prostor* (količina memorije koju algoritam koristi) (više o složenosti izračunavanja videti u [113]; tekst [85] govori o složenosti iz perspektive računarstva i veštačke inteligencije).

Za dati, specifični odlučiv problem, gornja granica potrebnog vremena i prostora obično se određuje razmatranjem konkretnog algoritma koji rešava dati problem u datim granicama resursa. Dodatno, da bi se dokazala optimalnost algoritma, potrebno je imati i odgovarajuće donje granice složenosti problema.

Razmatranje složenosti je od izuzetne važnosti za automatsko rezonovanje. Naime, činjenica da je neka teorija odlučiva nije od velike koristi ukoliko je donja granica potrebnog vremena i prostora previsoka, tj. ako je teorija *praktično* neodlučiva. Pitanje složenosti izračunavanja posebno je dobilo na značaju početkom sedamdesetih godina rezultatima Cooka [27]. Ti rezultati odnosili su se pre svega na klase P i NP, ali su delom i opštijeg karaktera.

Problem odlučivanja smatra se *efikasno rešivim* ako postoji algoritam koji rešava problem za sve njegove instance u broju koraka koji je polinomijalno ograničen veličinom ulazne instance, pri čemu se podrazumeva "tradicionalni" model izračunavanja tj. sekvencijalni, deterministički model (kao što je deterministička Tjuringova mašina (DTM) ili UR mašina). Ovu klasu problema označavamo sa P. Smatramo da problemi koji nisu u ovoj klasi nisu efikasno rešivi.

Nedeterministička Tjuringova mašina može da nedeterministički bira između dva različita puta (koristeći time tzv. *nd–izbore*). Kažemo da ona *prihvata* ulaznu vrednost ako i samo ako postoji niz *nd*–izbora takav da vodi prihvatanju te vrednosti u polinomijalnom vremenu. Klasu problema koje prihvata nedeterministička Tjuringova (NTM) mašina nazivamo klasom NP. Očigledno, važi P $\subseteq$ NP, ali se još uvek ne zna da li važi P=NP (pri čemu se generalno veruje da ne važi).

Uprkos tome što se ne zna da li važi P = NP, moguće je odrediti "najteže" probleme u klasi NP. Formalni alat za to je pojam *svodljivosti*. Kažemo da se skup A (odnosno problem za koji je odgovor *da* za elemente skupa A i samo za njih) može *više–jedan svesti* na skup B (odnosno odgovarajući problem) i pišemo $A \leq_m B$ ako postoji funkcija f takva da može da bude izračunata u polinomijalnom vremenu i ima svojstvo $w \in A$ ako i samo ako je $f(w) \in B$. Intuitivno, algoritam za problem B može biti iskorišćen za rešvanje problema A sa određenim dodatnim vremenom i prostorom potrebnim za svođenje problema. Problem koji ima svojstvo da svi problemi iz klase NP mogu biti u polinomijalnom vremenu svedeni na njega zovemo NP–*težak* problem. Problem koji pripada klasi NP i jeste NP–težak nazivamo NP–*kompletan* problem. Da bismo pokazali da je P = NP dovoljno je pokazati za jedan NP–kompletan problem da pripada klasi P. Cook je prvi pokazao da postoje NP–kompletni problemi[5] (jedan od njih je problem SAT — problem zadovoljivosti, [27]) i na bazi tog rezultata kasnije je dokazano za mnoge probleme da su NP–kompletni. Ni za jedan od njih, međutim, još uvek nije pokazano da ima polinomijalno rešenje. Opštije, ako su $\mathcal{C}_{lower}$ i $\mathcal{C}_{upper}$ dve klase kompleksnosti i ako važi $\mathcal{C}_{lower} \leq_{eff} A$ (tj. svaki problem iz $\mathcal{C}_{lower}$ je svodljiv na A), $A \in \mathcal{C}_{upper}$ i $\mathcal{C}_{lower} = \mathcal{C}_{upper} = \mathcal{C}$, onda kažemo da je problem A *$\mathcal{C}$-kompletan*.

Pojmove vremenske i prostorne složenosti izračunavanja uvešćemo za Tjuringovu mašinu. Ako je prilikom izračunavanja promenjeno t konfiguracija mašine, onda je t *vreme* tog izračunavanja. *Prostor* izračunavanja je broj polja mašine kojima se pristupa tokom izračunavanja. Tjuringova mašina *prihvata problem u vremenu (prostoru)* $F(n)$, ako za svaku ulaznu vrednost x koju prihvata (čija veličina može biti opisana prirodnim brojem) postoji izračunavanje koje je prihvata u vremenu (prostoru) koje ne prelazi $F(|x|)$. Neka NTIME($F(n)$) (NSPACE($F(n)$) označava klasu problema koje nedeterministička Tjuringova mašina prihvata u vremenu (prostoru) $F(n)$. Neka DTIME($F(n)$) (DSPACE($F(n)$) označava klasu problema koje deterministička Tjuringova mašina prihvata u

[5]Cook je prvi pokazao i da postoje P-kompletni problemi (jedan od njih je CVP — problem vrednosti kola).

vremenu (prostoru) $F(n)$. U opisivanju potrebnih resursa često koristimo O-notaciju. Ako je $G(n)$ funkcija iz $\mathbf{N}$ u $\mathbf{R}$, $O(G(n))$ je skup funkcija G' koje zadovoljavaju uslov $G'(n) \leq c \cdot G(n)$ za neku pozitivnu realnu konstantu c. O-notacija se u opisu potrebnih resursa koristi na prirodan način. Na primer, $\text{DTIME}(2^{O(n)})$ označava uniju skupova $\text{DTIME}(2^{cn})$ za sve konstante c. Klasu $O(G(n)^{O(1)})$ (klasu funkcija koje su ograničene odozgo nekom polinomijalnom funkcijom od G) označavamo sa $poly(G(n))$. Od posebne su važnosti sledeće klase problema:

$$\text{P}=\text{DTIME}(poly(n))$$
$$\text{NP}=\text{NTIME}(poly(n))$$
$$\text{PSPACE}=\text{DSPACE}(poly(n))$$
$$\text{DLOG}=\text{DSPACE}(\log(n))$$
$$\text{NLOG}=\text{NSPACE}(\log(n))$$

Između ostalog, može se dokazati da važi:

$$\text{P} \subseteq \text{NP}$$
$$\text{NP} \subseteq \text{PSPACE}$$
$$\text{DTIME}(T(n)) \subseteq \text{NTIME}(T(n))$$
$$\text{NTIME}(T(n)) \subseteq \text{DTIME}(2^{O(T(n))})$$
$$\text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$$
$$\text{NSPACE}(S(n)) \subseteq \text{DSPACE}(S(n)^2)$$
$$\text{NTIME}(T(n)) \subseteq \text{DSPACE}(T(n))$$
$$\text{DTIME}(T(n)) \subseteq \text{DSPACE}(T(n)/\log T(n))$$
$$\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(s(n))})$$

Između ostalih, još uvek su otvorena sledeća pitanja:

$$\text{P} = \text{NP} \ ?$$
$$\text{P} = \text{PSPACE} \ ?$$
$$\text{DLOG} = \text{NLOG} \ ?$$

Tabela A.1 prikazuje rezultate o kompleksnosti nekoliko teorija (više detalja videti u [113]). $\mathcal{C}_{lower}$ označava klasu problema koje je moguće svesti na ispitivanje dokazivosti rečenice date teorije (sva svođenja su ili polinomijalno–vremenskog ili logaritamsko–prostornog tipa). $\mathcal{C}_{upper}$ označava klasu za koju je dokazano da joj data teorija pripada.

Teorija	$\mathcal{C}_{lower}$	$\mathcal{C}_{upper}$
jednakost	$\text{NSPACE}(\sqrt{n})$	$\text{DSPACE}(n \log n)$
N sa sledbenikom	$\text{NSPACE}(n)$	$\text{DSPACE}(n^2)$
N sa sabiranjem	$\text{NTIME}(2^{2^n})$	$\text{DSPACE}(2^{2^{O(n)}})$
N sa množenjem	$\text{NTIME}(2^{2^{2^n}})$	$\text{DSPACE}(2^{2^{2^{O(n)}}})$
R sa sabiranjem	$\text{NTIME}(2^n)$	$\text{DSPACE}(2^{O(n)})$
R sa sabiranjem i množenjem	$\text{NTIME}(2^n)$	$\text{DSPACE}(2^{O(n^2)})$
konačne Abelove grupe	$\text{NTIME}(2^{2^n})$	$\text{DSPACE}(2^{2^{O(n)}})$

Slika A.1: Kompleksnost nekih teorija prvog reda

Iz tabele se vidi da su navedene teorije eksponencijalne ili supereksponencijalne složenosti. Isto važi za veliku većinu netrivijalnih teorija. Važno je, pogotovu sa aspekta automatskog dokazivanja teorema, i razmatranje veze složenosti teorije i odgovarajućih procedura odlučivanja. Kažemo da teorija $\mathcal{T}$ ima *inherentnu kompleksnost* $f(n)$ ako za svaku proceduru odlučivanja P za tu teoriju postoji beskonačno mnogo rečenica ϕ takvih da P zahteva više od $f(|\phi|)$ koraka za ispitivanje da li važi $\phi \in \mathcal{T}$. U potpoglavlju koje se odnosi na Prezburgerovu aritmetiku biće reči o složenosti teorije, ali i o osobinama pojedinih procedura odlučivanja.

A.3 Decidability and Decision Procedures: Summary

A theory $\mathcal{T}$ is decidable if there is an algorithm (which we call a *decision procedure*) such that for an input $\mathcal{T}$-formula F, it returns `true` if and only if $\mathcal{T} \vdash F$ (and returns `false` otherwise). The notion of decidable theories is formally introduced via recursive functions (or some other formalism) and is related to the informal definition via Church's thesis. Research on decidable and on undecidable theories have gone mainly along two different lines [98, 116]. Three basic methods for proving decidability of some theory are the following: the method of quantifier elimination; the model-theoretical method and the method of interpretations. We are mainly interested in the method of quantifier elimination as it induces effective decision procedures. Decision procedures are characterised by their computational complexity (among other attributes) and it typically varies from exponential (e.g. for propositional calculus) to double superexponential (e.g. for Presburger arithmetic).

Dodatak B

Sistemi za prezapisivanje, uređenja i zaustavljanje

Sistemi za prezapisivanje (eng. rewriting systems), imaju, u izvesnom smislu, svoj koren u sistemima za prezapisivanje niski Axela Thuea [117]. U računarstvu sistemi za prezapisivanje termova dobijaju na značaju od pojavljivanja znamenitog rada Knutha i Bendixa [70]. U tom radu Knuth i Bendix su razvili metodologiju za rešavanje jedne klase problema reči, ali je vremenom ta metodologija nalazila svoju primenu i u mnogim drugim domenima [38]. Danas se sistemi za prezapisivanje koriste u raznim tipovima dokazivača ili u njihovim pojedinim segmentima. Najčešća oblast primene sistema za prezapisivanje su jednakosne teorije.

Pravila prezapisivanja (eng. rewrite rules) su uređeni parovi izraza koje obično zapisujemo u obliku[1] $l \longrightarrow r$. Pravila su obično zasnovana na nekoj relaciji "sličnosti" između l i r. Ta relacija je najčešće jednakost ili ekvivalencija, ali može biti i relacija $\leq$, $\geq$ ili implikacija. U daljem tekstu u najvećoj meri razmatraćemo sisteme prezapisivanja zasnovane na relaciji jednakosti. Jednakost $l = r$ može biti orijentisana dajući pravilo $l \longrightarrow r$ koje onda govori da instance terma l mogu biti zamenjena odgovarajućim instancama terma r (ali ne i obratno). Na ovaj način se termovi obično zamenjuju jednostavnijim termovima (pri čemu se pojam "jednostavnosti" mora pažljivo razmotriti). Na primer, jednakost $x+s(y) = s(x+y)$, se može orijentisati u pravilo $x + s(y) \longrightarrow s(x + y)$. Primenom ovog pravila term $((x + 0) + s(x + y))$ može biti prezapisan u term $s((x + 0) + (x + y))$. Na ovaj način, ponekad je moguće dokazati da je neka jednakost posledica datog skupa jednakosti. Na primer, primenom navedenog pravila i pravila $x + 0 \longrightarrow x$ može biti dokazano da iz skupa $\{x+0 = x, x+s(y) = s(x+y)\}$ sledi jednakost $(x+0)+s(x+y) = x+(x+s(y))$. Zaista, prezapisivanjem obe strane date jednakosti može se dobiti identitet $s(x + (x + y)) = s(x + (x + y))$. U dokazima ovog tipa mogu se javiti dva

[1] U zapisu pravila prezapisivanja koristećemo simbol $\longrightarrow$ (dugačka strelica s leva nadesno) koji ne treba mešati sa simbolom $\rightarrow$ koji označava implikaciju.

problema:

- neka pravila prezapisivanja mogu se primenjivati beskonačno mnogo puta (npr. pravilo $a \longrightarrow f(a)$);
- ako se proces prezapisivanja nužno zaustavlja za bilo koji polazni term, dobijeni term može zavisiti od poretka primene pravila prezapisivanja.

Navedena dva svojstva sistema za prezapisivanje nazivaju se *zaustavljanje* (eng. termination) i *konfluentnost* (eng. confluence) sistema. Dokazano je da je u opštem slučaju neodlučiv i problem zaustavljanja i problem konfluentnosti sistema za prezapisivanje. Pod pretpostavkom da je sistem zaustavljajući (što može biti dokazano za neke sisteme), Knuth/Bendixova procedura upotpunjavanja predstavlja proceduru odlučivanja za problem konfluentnosti tog sistema; štaviše, ta procedura može, u nekim slučajevima, transformisati nekonfluentan sistem u njemu ekvivalentan konfluentan sistem.

Ukoliko je sistem za prezapisivanje termova istovremeno i zaustavljajući i konfluentan, onda on može biti korišćen i kao procedura odlučivanja za odgovarajuću jednakosnu teoriju: za svaku jednakost se, naime, može dokazati da li jeste ili nije logička posledica date jednakosne teorije.

Detaljniji pregled sistema za prezapisivanje, uređenja i zaustavljanja videti u radovima [96, 40, 61].

B.1 Termovi i pravila prezapisivanja

B.1.1 Termovi

Da bismo definisali pojam terma, najpre uvodimo pojam signature.[2] Svakoj formalnoj teoriji odgovara neki jezik, odnosno signatura. U narednim poglavljima uglavnom ćemo se zadržavati samo na sintaksnim pitanjima, dok ćemo pitanja semantike razmatrati samo za jednakosne teorije.

Definicija B.1 (Signatura) Signaturu *(eng. signature)* $\Sigma = (S, F, \tau)$ *čini:*

1. S*, neprazan skup* sorti *(ili* vrsta*)*[3]*;*
2. F*, neprazan skup* funkcijskih simbola *takav da je* $S \cap F = \emptyset$*;*
3. τ*, preslikavanje* $\tau : F \mapsto S^* \times S$ *koje pridružuje tip* $\tau(f) = (w, s)$ *svakom simbolu* $f \in F$*.*

Ako je $\tau(f) = (w, s)$*, onda* w *zovemo* tip domena*,* a s tip kodomena *ili* tip ranga*. Dužinu* $|w|$ *nazivamo* arnošću *simbola* f*. Ako je* $|w| = 0$*, onda tip* w *označavamo simbolom* λ*. Ako je* $\tau(f) = (\lambda, s)$ *onda za* f *kažemo da je* simbol konstante*. Skup simbola konstanti označavamo sa* $F_0 = \{f \in F \mid (\exists s \in S)\ \tau(f) = (\lambda, s)\}$*.*

[2] Pojam signature uveden u ovom delu razlikuje se u određenoj meri od pojma uvedenog u delu 2.1 i to zbog pogodnosti u izlaganju materijala u vezi sa prezapisivanjem.

[3] Za skup S definišemo operacije konkatenacije ($S \cdot S = \{s_1 s_2 | s_1, s_2 \in S\}$); stepena ($S^0 = \{\lambda\}$, $S^{i+1} = S^i \cdot S$) i iteracije ($S^* = \bigcup_{i \geq 0} S^i$).

Ako je $|S| = 1$, onda kažemo da je signatura *jednosortna* (eng. one sorted), a inače kažemo da je *višesortna* (eng. many sorted).

Definicija B.2 (Termovi nad signaturom Σ) *Neka je $\Sigma = (S, F, \tau)$ signatura i neka je X prebrojiv skup* promenljivih *(ili* varijabli *takav da važi $X \cap (S \cup F) = \emptyset$. Neka je preslikavanje τ definisano i za skup X ($\tau : F \cup X \mapsto S^* \times S$), pridružujući svakoj promenljivoj njenu jedinstvenu sortu s. Neka $X_s = \{x \in X \mid \tau(x) = (\lambda, s)\}$ označava skup promenljivih sorte s. Ako je signaturi Σ pridružen skup promenljivih X, onda tu signaturu označavamo sa (Σ, X). $Ter_s(\Sigma, X)$ označava skup svih termova sorte s i $Ter(\Sigma, X) = \bigcup_{s \in S} Ter_s(\Sigma, X)$ označava skup svih termova. Ovi skupovi definišu se na sledeći način:*

1. *Svaki simbol konstante $f \in F_0$ sorte s je term sorte s (tj. pripada skupu $Ter_s(\Sigma, X)$);*
2. *Svaka promenljiva $x \in X_s$ je term sorte s (tj. pripada skupu $Ter_s(\Sigma, X)$);*
3. *Ako je f funkcijski simbol tipa $\tau(f) = (s_1 s_2 \dots s_k, s)$ i t_i je term sorte s_i (za $1 \leq i \leq k$), onda je $f(t_1, t_2, \dots, t_k)$ term sorte s (tj. pripada skupu $Ter_s(\Sigma, X)$)*
4. *Termovi se mogu dobiti samo konačnom primenom pravila 1, 2. i 3.*

Term koji ne sadrži nijednu promenljivu nazivamo bazni term *ili* zatvoren term *(eng. ground term, closed term). Sa $Ter(\Sigma)$ označavamo skup baznih termova. Term nazivamo* linearnim *ako se nijedna promenljiva u njemu ne pojavljuje više nego jednom. Skup promenljivih koje se pojavljuju u termu t označavamo sa $var(t)$. Ukoliko su termovi t_1 i t_2 identični, to zapisujemo to $t_1 \equiv t_2$.*

Posmatran kao skup niski, skup $Ter_s(\Sigma, X)$ je kontekst-slobodan jezik $L_s \subseteq (\Sigma \cup X)^*$ za svaku sortu s ako su i S i X konačnu skupovi. U definisanju pojma terma koristi se prefiksna notacija, ali zbog čitljivosti često ćemo koristi i infiksnu notaciju (npr. umesto $\circ(x, y)$ pisaćemo često $x \circ y$).

Definicija B.3 (Podterm) *Kažemo da je term u* podterm *(eng. subterm) terma t ako je u jednako t ili ako je t oblika $f(t_1, t_2, \dots, t_n)$ i u je podterm terma t_i za neko i.*

Za svaki podterm t' nekog terma t moguće je definisati njegovu poziciju u okviru terma t. Time se svaki podterm može jednoznačno opisati.

Definicija B.4 (Broj pojavljivanja) *Neka je $\Sigma = (S, F, \tau)$ signatura i X pridružen skup promenljivih. Za svaki term $t \in Ter(\Sigma, X)$ definišemo* broj pojavljivanja *(u oznaci $|t|_g$) funkcijskog simbola ili promenljive $g \in F \cup X$ u termu t na sledeći način:*

$$|t|_g = \begin{cases} 0, & \text{ako je } t \neq g \text{ i } t \in (X \cup F_0) \\ 1, & \text{ako je } t = g \text{ i } t \in (X \cup F_0) \\ \sum_{i=1}^{n} |t_i|_g, & \text{ako je } t = f(t_1, \dots, t_n),\ n \geq 1 \text{ i } f \neq g \\ 1 + \sum_{i=1}^{n} |t_i|_g, & \text{ako je } t = f(t_1, \dots, t_n),\ n \geq 1 \text{ i } f = g \end{cases}$$

B.1.2 Supstitucija i unifikacija

Definicija B.5 (Supstitucija) *Neka je* (Σ, X) *signatura.* Supstitucija *(eng. substitution)* σ *je preslikavanje* $\sigma : Ter(\Sigma, X) \mapsto Ter(\Sigma, X)$ *za koje važi:*

1. $\sigma(f(t_1, \ldots, t_k)) = f(\sigma(t_1), \ldots, \sigma(t_k))$ *za svaki funkcijski simbol* $f \in F$*;*
2. $\sigma(x) \in Ter_s(\Sigma, X)$ *za svaku promenljivu* $x \in X_s$*.*

Ako je $\sigma(x) \in Ter(\Sigma)$ *za svaku promenljivu* $x \in X$*, onda* σ *nazivamo* baznom supstitucijom *(eng. ground substitution). Za term* t_2 *kažemo da* odgovara *(eng. match) termu* t_1 *ili da je* instanca terma t_1 *ako je* $t_2 = \sigma(t_1)$ *za neku supstituciju* σ*. Term dobijen od terma* t *primenom supstitucije* σ *označavamo i sa* $t\sigma$*.*

Kompoziciju σ dve supstitucije φ i ψ označavamo (kao i obično kompoziciju preslikavanja) sa $\sigma = \psi \circ \varphi$, a efekat primene te supstitucije na term t označavamo sa $\psi(\varphi(t))$ ili $t\varphi\psi$. Supstituciju σ često označavamo i u formi $\sigma = \{x \mapsto \sigma(x_1), \ldots, x_k \mapsto \sigma(x_k)\}$ (ili u formi $\sigma = \{\sigma(x_1)/x, \ldots, \sigma(x_k)/x_k\}$) opisujući σ samo za one promenljive $x_1, \ldots, x_k$ za koje je $\sigma(x_i) \neq x_i$. Obično se pretpostavlja da je $\sigma(x) \neq x$ samo za konačno mnogo promenljivih.

Definicija B.6 (Sintaksna unifikacija) *Neka je* $t_1, t_2 \in Ter(\Sigma, X)$*.*

1. *Supstituciju* σ *koja zadovoljava* $t_1\sigma = t_2\sigma$ *nazivamo* unifikatorom *(eng. unifier) termova* t_1 *i* t_2*.*
2. $U(t_1, t_2) = \{\sigma \mid t_1\sigma = t_2\sigma\}$ *je skup svih unifikatora termova* t_1 *i* t_2*.*
3. *Za unifikator* σ *termova* t_1 *i* t_2 *kažemo da je* najopštiji unifikator *(eng. most general unifier) (i zapisujemo* $mgu(t_1, t_2)$*) ako se svaki unifikator* φ *može dobiti iz* σ *primenom neke dodatne supstitucije* ψ*, tj. ako za svaki unifikator* φ *postoji supstitucija* ψ *takva da važi* $\varphi = \psi \circ \sigma$*.*

Ako je sintaksna unifikacija za termove t_1 i t_2 moguća, tj. ako postoji bar jedan njihov unifikator, onda postoji najviše jedan najopštiji unifikator (do na imena promenljivih) [99]. Slično, ako je term t_1 instanca terma t_2, tj. ako postoji supstitucija ψ takva da je $t_2\psi = t_1$, onda postoji najopštija takva supstitucija σ, tj. svaka takva supstitucija φ može se dobiti iz σ primenom neke dodatne supstitucije.

Primetimo, ako je term t_1 instanca terma t_2, onda je supstitucija σ za koju važi $t_1 = \sigma t_2$ restrikcije unifikacije (ona se primenjuje samo na jedan term) i tu unifikaciju nazivamo *jednosmerno uparivanje* (eng. one way matching).

Primer B.1

- *Ne postoji unifikator za termove* $t_1 = f(g(a, x), y)$ *i* $t_2 = f(x, g(a, b))$ *(jer u svakoj supstituciji* σ*, term* $x\sigma$ *je pravi podterm terma* $g(a, x)\sigma$*.*
- *Najopštiji unifikator za termove* $t_1 = f(y, g(a, x))$ *i* $t_2 = f(x, g(a, b))$ *je* $\sigma = \{x \mapsto b, y \mapsto b\}$*.*

- *Najopštiji unifikator za termove* $t_1 = f(g(x_0, x_0), g(x_1, x_1), \ldots, g(x_{n-1}, x_{n-1})$ *i* $t_2 = f(x_1, x_2, \ldots, x_n)$ *je supstitucija* σ *eksponencijalne veličine. Ona se može opisati rekurzivnim procesom zamenjivanja: najpre se* x_n *preslikava u* $g(x_{n-1}, x_{n-1})$*, zatim se* x_{n-1} *preslikava u* $g(x_{n-2}, x_{n-2})$*, ..., i konačno* x_1 *se preslikava u* $g(x_0, x_0)$*. Tek onda se određuju termovi u koje se preslikavaju promenljive* $x_2, x_3, \ldots, x_n$*.*

Problem određivanja najopštijeg unifikatora prvi je rešio Robinson [99]. Osnovni algoritam unifikacije je eksponencijalne složenosti, ali postoji i linearno rešenje problema uz korišćenje posebnih struktura podataka (ali ono nije pogodno za praktičnu primenu).

B.1.3 Pravila prezapisivanja

Definicija B.7 (Pravilo prezapisivanja) *Ako je* (Σ, X) *signatura, element* (l, r) *skupa* $Ter(\Sigma, X) \times Ter(\Sigma, X)$ *nazivamo* pravilom prezapisivanja *(*pravilom redukcije*,* pravilom svođenja*) (eng. rewrite rule, reduction rule) ako važi:*

1. $l, r \in Ter_s(\Sigma, X)$ *za neko* $s \in S$*;*
2. $l \notin X$*;*
3. $var(r) \subseteq var(l)$*.*

Pravilo prezapisivanja (l, r) *zapisujemo i u obliku* $l \longrightarrow r$*. Pravilo prezapisivanja* $l \longrightarrow r$ *može se primeniti na term* t *ako postoji podterm* $t' \in Ter_s(\Sigma, X)$ *terma* t *i supstitucija* φ *takva da je* $l\varphi \equiv t'$*. Efekat primene pravila je zamenjivanje terma* t' *termom* $t'\varphi$ *u termu* t*.*

Uslov *3* iz prethodne definicije se ponekad (retko) izostavlja. Smisao tog uslova je u tome da se odbacuju pravila kao što je $f(x) \longrightarrow g(x, y)$. Primena takvih pravila može da stvara probleme zato što, u navedenom primeru, promenljiva y mora biti instancirana prilikom prezapisivanja ili prezapisivanje uvodi uvek nove promenljive. Pored uslova *3*, ponekad se izostavlja i uslov *2*, ali ipak najčešće se koristi navedena definicija pravila prezapisivanja.

Primetimo da se efekat jednostruke primene nekog pravila prezapisivanja oslikava na samo jednom podtermu terma t (a ne na svim takvim podtermovima). Taj efekat može se preciznije opisati ako je na raspolaganju i pozicija odgovarajućeg podterma u termu t.

Efekat primene pravila prezapisivanja možemo opisati i u vidu pravila izvođenja:

$$\frac{t[l\varphi] \quad l \longrightarrow r}{t[r\varphi]}$$

gde je sa $t[l\varphi]$ označen term t koji sadrži podterm $t' \equiv l\varphi$ (pri čemu se misli samo na jedan takav podterm i to se može precizirati označavanjem pozicije).

Ukoliko je pravilo prezapisivanja zasnovano na relaciji jednakosti, (tj. ako je pravilo prezapisivanja $l \longrightarrow r$ izvedeno iz neke jednakosti $l = r$) onda je

odgovarajuće pravilo izvođenja pojednostavljena varijanta paramodulacije [19]. Ta pojednostavljenost se ogleda pre svega u tome što se supstitucija φ primenjuje samo na term l, a ne i na term t. Ovakvu restrikciju unifikacije nazivamo *jednosmerno uparivanje* (eng. one way matching).

Primer B.2 *Pravilo prezapisivanja* $x + f(y) \longrightarrow x + c$ *može biti primenjeno na term* $(d + f(f(z))) * y$ *dajući rezultat* $(d + c) * y$ *(koristeći supstituciju* $\varphi = \{x \mapsto d, y \mapsto f(z)\}$*), ali ne može biti primenjeno na term* $w + z$ *(jer ne postoji supstitucija* φ *takva da je* $f(y)\varphi \equiv z$*) niti na term* $a + c$ *(jer pravilo ne može biti primenjeno u suprotnom smeru).*

Primer B.3 *Pravilo prezapisivanja* $\neg(A \vee B) \longrightarrow \neg A \wedge \neg B$ *može biti primenjeno na* $\neg(p \vee \neg q) \vee r$ *korišćenjem supstitucije* $\varphi = \{A \mapsto p, B \mapsto q\}$ *dajući* $(\neg p \wedge \neg\neg q) \vee r)$.

Definicija B.8 (Sistem za prezapisivanje termova) *Skup* R *pravila prezapisivanja nad signaturom* (Σ, X) *nazivamo* sistem za prezapisivanje termova (SPT) *(eng. term rewriting system (TRS)) ili* sistemom svođenja, sistemom redukcija *(eng. reduction system).*

U daljem tekstu, ukoliko ne naglasimo drugačije, pod sistemom za prezapisivanje podrazumevaćemo sistem za prezapisivanje sa konačno mnogo pravila pravila prezapisivanja.

Ako je $R \subseteq Ter(\Sigma) \times Ter(\Sigma)$, onda R nazivamo *baznim sistemom prazapisivanja* (eng. ground term rewriting system).

Ako je rezultat primene nekog pravila sistema za prezapisivanje R na term t_1 term t_2, onda to zapisujemo $t_1 \Longrightarrow_R t_2$ ili kraće $t_1 \Longrightarrow t_2$ ako je izbor sistema R jasan iz konteksta. Dakle, R indukuje relaciju $\Longrightarrow_R \subseteq Ter(\Sigma, X) \times Ter(\Sigma, X)$ koja zadovoljava sledeće svojsto: važi $t_1 \Longrightarrow_R t_2$ ako postoji pravilo prezapisivanja $l \longrightarrow r$ (iz R) takvo da je $l, r \in Ter_s(\Sigma, X)$, podterm $t' \in Ter_s(\Sigma, X)$ terma t i supstitucija φ takva da je $l\varphi \equiv t'$ i ako je t_2 term dobijen zamenjivanjem terma t' termom $t'\varphi$ u termu t.

Ako je $t_1 \Longrightarrow_R t_2 \ldots \Longrightarrow_R t_n$ i $n \geq 1$, onda to zapisujemo kraće $t_1 \Longrightarrow_R^* t_n$ (ili $t_1 \Longrightarrow^* t_2$ ako je izbor sistema R jasan iz konteksta) i kažemo da se t_1 prezapisuje u ili svodi na term t_n u sistemu R. Ako važi $t_1 \Longrightarrow^* t$ i $t_2 \Longrightarrow^* t$ za neki term t, onda to zapisujemo $t_1 \Downarrow t_2$. Ako važi $t_1 \Longrightarrow_R t_2$ ili $t_2 \Longrightarrow_R t_1$, onda to zapisujemo $t_1 \Longleftrightarrow_R t_2$. Ako važi

$$t_1 \Longleftrightarrow_R t_2 \Longleftrightarrow_R \ldots \Longleftrightarrow_R t_n$$

pri čemu je $n \geq 0$ onda to zapisujemo $t_1 \Longleftrightarrow_R^* t_n$. Primetimo da je relacija $\Longleftrightarrow^*$ refleksivno, simetrično i tranzitivno zatvorenje relacije $\Longleftrightarrow$.

Relacije $\Longrightarrow_R$, $\Longleftrightarrow_R$ i $\Longleftrightarrow_R^*$ moguće je uvesti i nešto formalnije (videti [96]) pravilima izvođenja poput

$$\frac{r \longrightarrow s}{r \Longrightarrow s} \qquad \frac{r \Longrightarrow s}{r \Longrightarrow^* s}$$

$$\frac{r \longrightarrow s \quad \varphi \text{ supstitucija}}{r\varphi \Longrightarrow^* s\varphi}$$

$$\frac{r \Longrightarrow^* s \qquad s \Longrightarrow^* t}{r\varphi \Longrightarrow^* t\varphi} \ .$$

Primer B.4 *Pravila prezapisivanja iz skupa* $\{x \cdot 0 \longrightarrow 0, 1 \cdot x \longrightarrow x, x^0 \longrightarrow 1, x + 0 \longrightarrow x\}$ *mogu biti iskorišćena u prezapisivanju terma* $a^{2\cdot 0} \cdot 5 + b \cdot 0$ *na sledeći način:* $a^{2\cdot 0}\cdot 5+b\cdot 0 \Longrightarrow a^0\cdot 5+b\cdot 0 \Longrightarrow 1\cdot 5+b\cdot 0 \Longrightarrow 5+b\cdot 0 \Longrightarrow 5+0 \Longrightarrow 5$. *Dakle, važi* $a^{2\cdot 0} \cdot 5 + b \cdot 0 \Longrightarrow^* 5$.

Definicija B.9 (Ekvivalentni sistemi za prezapisivanje) *Za dva sistema za prezapisivanje* R_1 *i* R_2 *kažemo da su ekvivalentna ako važi:*

$$t_1 \Longrightarrow^*_{R_1} t_2 \quad \textit{ako i samo ako} \quad t_1 \Longrightarrow^*_{R_2} t_2$$

B.1.4 Nesvodljiv term i normalna forma

Definicija B.10 (Nesvodljiv term) *Ako na term* t *nije moguće primeniti nijedno pravilo sistema* R *za prezapisivanje, onda kažemo da je term* t *nesvodljiv (eng. irreducible) u odnosu na* R.

Dakle, ukoliko je term t nesvodljiv u odnosu na sistem R onda ne postoji term t' takav da je $t \Longrightarrow t'$ (iako, primetimo, uvek važi $t \Longrightarrow^* t'$).

Definicija B.11 (Normalna forma) *Ako važi* $t_1 \Longrightarrow^*_R t_n$ *i* t_n *je nesvodljiv term u odnosu na* R, *onda kažemo da je* t_n normalna forma *(eng. normal form) terma* t_1 *u odnosu na* R. *Normalnu forma terma* t *označavamo sa* $t\downarrow$.

Naglasimo da, u opštem slučaju, normalna forma za neki term nije jedinstvena. Ukoliko je za neki term normalna forma jedistvena (u odnosu na neki sistem za prezapisivanje), onda nju nazivamo *kanonska forma* (eng. canonical form).

Definicija B.12 (Kanonska forma) *Ukoliko neki sistem za prezapisivanje* R *i za neki term* t *ne postoji beskonačno prezapisivanje* $t \Longrightarrow t_1 \Longrightarrow t_2 \Longrightarrow \ldots$ *i ukoliko su sve normalne forme terma* t *identične nekom termu* t', *onda taj term nazivamo* kanonskom formom *(eng. canonical form) terma* t.

B.1.5 Uslovno prezapisivanje

Ponekad obična pravila prezapisivanja nisu dovoljno moćna. Na primer, ponekad je potrebno uvesti pravilo prezapisivanja zasnovano na nekoj jednakosti $s = t$ koja je zadovoljena samo ako je ispunjen neki uslov C. U tim slučajevima koriste se *uslovni sistemi za prezapisivanje* u kojima svako pravilo ima pridružen uslov koji mora biti zadovoljen da bi pravilo bilo primenjeno. Na primer, pravilo $s \longrightarrow t$ sa uslovom C zapisuje se na sledeći način:

$$C \mid s \longrightarrow t$$

ili

$$s \longrightarrow t \textbf{ if } C$$

gde je C *čuvar* (eng. guard) pravila za prezapisivanje $s \longrightarrow t$. Uslov C može imati različite oblike; na primer, on može biti logička formula, jednakost ili nejednakost. Trivijalno, C može biti i uslov koji je uvek ispunjen ($C \equiv true$). Više o uslovnom prezapisivanju videti u [96].

B.2 Zaustavljanje i konfluentnost

B.2.1 Zaustavljanje

Definicija B.13 (Zaustavljanje) *Kažemo da je sistem za prezapisivanje R* zaustavljajući[4] *(*terminirajući, jako normalizujući*) (eng. terminating, strongly normalizing, Noetherian) ako ne postoji beskonačan niz termova t_1, t_2, t_3, ... takav da je*

$$t_1 \Longrightarrow_R t_2 \Longrightarrow_R t_3 \Longrightarrow_R \dots$$

Primer B.5

- *Sistem za prezapisivanje $\{x+y \longrightarrow y+x\}$ nije zaustavljajući jer za term $a + b$ postoji beskonačan niz svođenja: $a + b \Longrightarrow b + a \Longrightarrow a + b \dots$.*
- *Sistem za prezapisivanje $\{a \longrightarrow b, b \longrightarrow a\}$ nije zaustavljajući zbog beskonačnog niza $a \Longrightarrow b \Longrightarrow a \Longrightarrow_R \dots$, iako oba njegova pravila, pojedinačno određuju zaustavljajuće sisteme.*

Dokazano je da je u opštem slučaju zaustavljanje sistema za prezapisivanje neodlučivo. To svojstvo je neodlučivo čak i za sisteme za prezapisivanje koji imaju samo jedno pravilo [34, 35]. Ipak, postoje sistemi za koje je taj problem odlučiv. O jednoj klasi takvih sistema govori naredna teorema:

Teorema B.1 (Bazno zaustavljanje, Huet & Lankford 1978) *Za svaki bazni sistem za prezapisivanje odlučivo je da li je on zaustavljajući.*

Očigledno, ako je sistem za prezapisivanje zaustavljajući, onda svaki element skupa $Ter(\Sigma, X)$ ima (bar jednu) normalnu formu, ali ona ne mora biti jedinstveno određena. Naime, moguće je da normalna forma nekog terma zavisi od poretka primene pojedinih pravila prezapisivanja. Očigledno, ako sistem za prezapisivanje ima konačno mnogo pravila, onda svaki term ima konačno mnogo normalnih formi.

Zaustavljanje nekih sistema za prezapisivanje moguće je dokazati korišćenjem relacija uređenja nad termovima. O uređenima govori poglavlje B.3.

[4] Suštinski, misli se na to da je relacija $\Longrightarrow_R$ zaustavljajuća (i analogno konfluentna, lokalno konfluentna itd.), a ne sâm sistem R [54].

B.2.2 Konfluentnost i Church-Rosserovo svojstvo

Konfluentnost i Church-Rosserovo svojstvo inicijalno su bili definisani i razmatrani u širem kontekstu tzv. transformacionih sistema, ali mi ćemo ta svojstva i tvrđenja u vezi sa njima dati samo u kontekstu sistema za prezapisivanje.

Definicija B.14 (Konfluentnost) *Sistem za prezapisivanje je* konfluentan *ako iz*

$$t_1 \Longrightarrow^* t_2 \; i \; t_1 \Longrightarrow^* t_3$$

sledi da postoji term t_4 takav da je

$$t_2 \Longrightarrow^* t_4 \; i \; t_3 \Longrightarrow^* t_4 \; ,$$

tj. da važi $t_2 \Downarrow t_3$.

Primer B.6

- *Sistem za prezapisivanje $\{x + y \longrightarrow y + x\}$ je konfluentan.*
- *Sistem za prezapisivanje $\{a \longrightarrow b, b \longrightarrow a\}$ je konfluentan.*
- *Sistem $\{f(g(f(x))) \longrightarrow g(x)\}$ nije konfluentan jer term $f(g(f(g(f(a)))))$ može biti prezapisan i u term $f(g(a))$ i u $g(g(f(a)))$ pri čemu su oba terma u normalnoj formi i ne mogu biti prezapisani dalje.*

Ako je sistem za prezapisivanje R konfluentan, onda, očigledno, za svaki term sve njegove normalne forme (ako postoje) moraju biti identične.

Dokazano je da je u opštem slučaju konfluentnost sistema za prezapisivanje neodlučiva (videti npr. [64]). Ipak, postoje sistemi za koje je taj problem odlučiv. O nekim klasama takvih sistema govore naredne dve teoreme:

Teorema B.2 (Daucher & Tison 1984) *Konfluentnost je odlučiva za svaki bazni sistem za prezapisivanje.*

Teorema B.3 *Konfluentnost je odlučiva za svaki zaustavljajući sistem za prezapisivanje.*

Naglasimo da zaustavljanje nije neophodan uslov za konfluentnost: postoje sistemi koji su konfluentni iako nisu zaustavljajući. Takav je, na primer, sistem za prezapisivanje $\{a \longrightarrow b, a \longrightarrow c, c \longrightarrow a\}$.

Ekvivalentno svojstvu konfluentnosti je Church-Rosserovo svojstvo (koje je originalno formulisano u kontekstu apstraktnih smena ili transformacionih sistema).

Definicija B.15 (Church-Rosserovo svojstvo) *Kažemo da skup pravila prezapisivanja R ima* Church-Rosserovo svojstvo *ako za bilo koja dva terma t_1 i t_2 važi*

$$t_1 \Longleftrightarrow^* t_2 \; ako \; i \; samo \; ako \; t_1 \Downarrow t_2 \; .$$

Teorema B.4 *Sistem za prezapisivanje je konfluentan ako i samo ako ima Church-Rosserovo svojstvo.*

Dokaz videti npr. u [54] ili [96].

B.2.3 Lokalna konfluentnost

Definicija B.16 (Lokalna konfluentnost) *Sistem za prezapisivanje je lokalno konfluentan (ili* slabo konfluentan*) ako iz*

$$t_1 \Longrightarrow t_2 \; i \; t_1 \Longrightarrow t_3$$

sledi da postoji term t_4 takav da je

$$t_2 \Longrightarrow^* t_4 \; i \; t_3 \Longrightarrow^* t_4 \; ,$$

tj. da važi $t_2 \Downarrow t_3$.

Trivijalno se dokazuje da je svaki konfluentan sistem za prezapisivanje lokalno konfluentan. Za zaustavljajuće sisteme važi i obratno:

Teorema B.5 (Newmanova lema, 1942) *Ako je sistem za prezapisivanje R zaustavljajući, onda važi: R je konfluentan ako i samo ako je R lokalno konfluentan.*

Naglasimo da je uslov zaustavljanja u prethodnoj teoremi neophodan. Naime, ako sistem nije zaustavljajući, lokalna konfluentnost ne povlači nužno konfluentnost. Zaista, (nezaustavljajući) sistem $\{a \longrightarrow b, b \longrightarrow a, a \longrightarrow c, b \longrightarrow d\}$ je lokalno konfluentan, ali nije konfluentan.

Definicija B.17 (Kritični par) *Neka su $l \longrightarrow r$ i $l' \longrightarrow r'$ dva pravila prezapisivanja sistema R. Kažemo da l* preklapa *(eng. overlaps) l' ako postoji podterm t terma l' takav da t nije promenljiva i da postoji najopštija unifikacija φ za l i t. Tada se* kritični par *(eng. critical pair) formira na sledeći način: $t\varphi$ se zamenjuje termom $r\varphi$ u $l'\varphi$ dajući term l''; kritični par polazna dva pravila je par koji čine l'' i $r'\varphi$ i označavamo ga $\langle l'', r'\varphi \rangle$. (Tada kažemo i da se pravilo $l \longrightarrow r$ superponira (eng. superposes) sa pravilom $l' \longrightarrow r'$.*

Dakle, ako l preklapa l', onda se oba pravila $l \longrightarrow r$ i $l' \longrightarrow r'$ mogu primeniti na term $l'\varphi$. Prvo pravilo daje razultujući term l'', a drugo term $r'\varphi$, pa je $l'\varphi \Longrightarrow_R l''$ i $l'\varphi \Longrightarrow_R r'\varphi$.

Primer B.7 *Pravilo $x + c \longrightarrow c$ preklapa pravilo $(a + y) + z \longrightarrow y + z$ na dva podterma: $(a + y)$ i $(a + y) + z$.*

Za prvi term u odgovarajućoj najopštijoj unifikaciji $\varphi' = \{x \mapsto a, y \mapsto c\}$ unifikuju se termi $a + y$ i $x + c$, pa se zamenjivanjem terma $(a + y)\varphi' \equiv a + c$ termom $c\varphi' \equiv c$ u termu $((a+y)+z)\varphi' \equiv (a+c)+z$ dobija term $c+z$. S druge strane, važi $(y + z)\varphi' \equiv c + z$, pa je prvi kritični par $\langle c + z, c + z \rangle$.

Za drugi term u odgovarajućoj najopštijoj unifikaciji $\varphi'' = \{x \mapsto a+y, z \mapsto c\}$ unifikuju se termi $(a+y)+z$ i $x+c$, pa se zamenjivanjem terma $((a+y)+z)\varphi'' \equiv x + c$ termom $c\varphi' \equiv c$ u termu $((a + y) + z)\varphi'' \equiv x + c$ dobija term c. S druge strane, važi $(y + z)\varphi'' \equiv y + c$, pa je drugi kritični par $\langle c, y + c \rangle$.

Ako je sistem za prezapisivanje konačan (što je najšće slučaj), onda on ima konačno mnogo kritičnih parova i moguće ih je efektivno odrediti. Za zaustavljajući sistem za prezapisivanje termova, za bilo koja dva terma t_1 i t_2 moguće je efektivno ispitati da li važi $t_1 \Downarrow t_2$. Na osnovu toga i na osnovu sledeće teoreme, sledi da je za zaustavljajući sistem za prezapisivanje lokalna konfluentnost odlučiva:

Teorema B.6 (Lema o kritičnom paru; Knuth & Bendix 1970) *Sistem za prezapisivanje je lokalno konfluentan ako i samo ako važi $p \Downarrow q$ za svaki kritični par $\langle p, q\rangle$ sistema R.*

Navedena teorema važi i za zaustavljajuće i za nezaustavljajuće sisteme za prezapisivanje. Međutim njen značaj se pre svega ogleda u zaustavljajućim sistemima, za koje ona praktično daje proceduru odlučivanja za lokalnu konfluentnost (a time, na osnovu Newmanove leme, i za konfluentnost sistema za prezapisivanje).

B.2.4 Kanonski sistem

Definicija B.18 (Kanonski sistem) *Za sistem za prezapisivanje kažemo da je* kanonski *ako je konfluentan i zaustavljajući.*

Jednostavno se dokazuje da u kanonskom[5] sistemu važi da svaki term ima kanonsku formu tj. da su sve njegove normalne forme identične.

Značaj kanonskih sistema može biti ilustrovan na primeru sistema za prezapisivanje termova zasnovanom na skupu jednakosti. Ukoliko je neki skup jednakosti E moguće transformisati u zaustavljajući i konfluentan sistem za prezapisivanje termova, onda postoji procedura odlučivanja za ispitivanje da li neka jednakosti $s = t$ sledi iz sistema skupa jednakosti E. Naime važi $E \models s = t$ ako i samo ako $s \downarrow \equiv t \downarrow$.

Na osnovu navedenih teorema može se izvesti postupak (zasnovan na [70]) za proveravanje da li je neki sistem za prezapisivanje kanonski:

1. Dokazati da je sistem zaustavljajući (u opštem slučaju neodlučivo).

2. Naći sve kritične parove (za konačan skup pravila moguće efektivno uraditi u konačnom vremenu).

[5]Neki autori pojam kanonskog sistema za prezapisivanje uvode drugačije: npr. kao sistem za koji svaki term ima kanonsku formu ili kao sistem koji je zaustavljajući i lokalno konfluentan. No, može se dokazati da su tako uvedeni pojmovi ekvivalentni. Neki autori umesto termina kanonski sistem koriste termin *konvergentan sistem* (eng. convergent system) [40], a za sistem onda kažu da je kanonski, ako je konvergentan i redukovan. (sistem za prezapisivanje nazivamo *redukovanim* ili *svedenim* (eng. reduced) ako su u svakom pravilu sistema oba elementa nesvodljivi termi u odnosu na preostala pravila sistema).

U svom radu [70] Knuth i Bendix ne koriste termin kanonski sistem, već potpun (kompletan) sistem. Pored toga Knuth i Bendix umesto termina pravilo prezapisivanja koriste termin pravilo svođenja (eng. reduction rule).

3. Odrediti sve normalne forme svih elemenata svih kritičnih parova (kada je sistem zaustavljajući, to je moguće efektivno uraditi u konačnom vremenu).

4. Dokazati da nijedan od tih elemenata nema više od jedne normalne forme (trivijalno, na bazi prethodnog koraka).

5. Dokazati da u svakom paru oba elementa imaju istu normalnu formu.

Naglasimo da dati postupak nije potpun, tj. ukoliko se njime ne može dokazati da dati sistem za prezapisivanje nije kanonski (tj. ukoliko nije moguće uraditi neki od koraka), to još uvek ne znači da taj sistem nije kanonski.

Smisao opisanog postupka je sledeći: prvi korak proverava zaustavljanje, a preostali konfluentnost sistema (na bazi Newmanove leme). Ako je sistem zaustavljajući i konfluentan (tj. kanonski), onda svaki term ima jedinstvenu normalnu formu. Ako postoji term za koji to ne važi, onda sistem za prezapisivanja nije kanonski. Na osnovu teoreme B.6, to je dovoljno proveriti za elemente kritičnih parova. U petom koraku postupka proverava se suštinski uslov teoreme B.6.

Ako sistem za prezapisivanje nije kanonski, u nekim slučajevima moguće ga je transformisati u ekvivalentan kanonski sistem. Knuth/Bendixova procedura upotpunjavanja daje delimično rešenje tog problema. O toj proceduri govori poglavlje B.4.

Čak i kada je sistem za prezapisivanje kanonski, poredak primene pravila prezapisivanja može da bude relevantan sa aspekta efikasnosti (tj. broja primenjenih pravila prezapisivanja). Postoji više strategija prezapisivanja (iznutra-spolja (eng. inside out), spolja-iznutra (eng. outside in) i dr.), ali u ovom tekstu nećemo razmatrati ta pitanja (videti npr. [19, 96]).

B.2.5 Prezapisivanje po modulu

U primeru B.8 rečeno je da ne postoji zaustavljajući konfluentan sistem za prezapisivanje koji iskaznu formulu transformiše u konjuktivnu normalnu formu. Ta činjenica zasnivala se na svojstvima pravila komutativnosti koja uvek narušavaju zaustavljanje, ma kako ona bila orijentisana (npr. jednakost $x + y = y + x$ ne može biti orijentisana tako da sistem za prezapisivanje bude zaustavljajući). Isto važi i za pravila asocijativnosti. Zbog značaja ovih pravila u mnogim sistemima, pojam kanonskog sistema se proširuje i zamenjuje pojmom *kanonskog sistema po modulu asocijativnosti i/ili komutativnosti*. Sistem opisan u primeru B.8 je kanonski po modulu asocijativnosti i komutativnosti. Drugim rečima, svaka iskazna formula može biti transformisana u jedinstvenu formulu u konjuktivnoj normalnoj formi zanemarujući poredak disjunkata u konjuktima i poredak konjukata u dobijenoj formuli. Više o pojmu konfluentnosti po modulu videti npr. u [61].

B.3 Uređenja i zaustavljanje

Za dokazivanje zaustavljanja sistema za prezapisivanje i za konstruisanje kanonskih sistema (npr. primenom Knuth/Bendixove procedure upotpunjavanja) potreban je poredak koji je u određenom smislu saglasan sa prezapisivanjem termova. Postoji puno istraživanja na temu takvih uređenja (videti npr. [56, 39, 96]). Kako ne postoji odgovarajuće totalno uređenje nad skupom termova, pažnja se usredsređuje na uređenja koja mogu da pokriju što više termova.

U ovom poglavlju opisaćemo neke tehnike za dokazivanje da je sistem za prezapisivanje zaustavljajući. Te tehnike zasnivaju se na korišćenju dobro zasnovanih parcijalnih uređenja $>$ koja imaju svojstvo da $l \Longrightarrow r$ povlači $l > r$. Ako takvo uređenje postoji, onda je odgovarajući sistem zaustavljajući. U nastavku poglavlja biće izloženo nekoliko tipova zaustavljajućih uređenja. Pre toga biće uvedeno nekoliko potrebnih pojmova. Za širi pregled uređenja videti [96, 40, 61].

Definicija B.19 (Parcijalno uređenje) *Za relaciju R nad skupom S kažemo da je* parcijalno uređenje *(eng. partial ordering) skupa S ako je ona:*

- *refkleksivna: za svako $s \in S$ važi sRs;*
- *antisimetrična: ako važi s_1Rs_2, onda ne važi s_2Rs_1;*
- *tranzitivna: za sve $s_1, s_2, s_3 \in S$ važi da s_1Rs_2 i s_2Rs_3 povlači s_1Rs_3.*

Definicija B.20 (Strogo parcijalno uređenje) *Za relaciju R nad skupom S kažemo da je* strogo parcijalno uređenje *(eng. strict partial ordering) skupa S ako je ona:*

- *nerefkleksivna: ni za jedno $s \in S$ ne važi sRs;*
- *tranzitivna: za sve $s_1, s_2, s_3 \in S$ važi da s_1Rs_2 i s_2Rs_3 povlači s_1Rs_3.*

Ako za sve $s_1, s_2 \in S$ takve da je $s_1 \not\equiv s_2$ važi s_1Rs_2 ili s_2Rs_1, onda kažemo da je relacija R totalno uređenje *skupa S.*

Primetimo da je strogo parcijalno uređenje antisimetrična relacija. Dakle, nerefleksivni deo strogog parcijalnog uređenja je parcijalno uređenje. Proširivanje strogog parcijalnog uređenja refleksivnim delom daje parcijalno uređenje.

Parcijalna uređenja označavamo najčešće simbolom $\geq$ ili simbolom $\succeq$. Ako je $\geq$ parcijalno uređenje skupa S, onda relaciju $\sim$ uvodimo na sledeći način: važi $s \sim t$ ako i samo ako $s \geq t$ i $t \geq s$. Očigledno, relacija $\sim$ je relacija ekvivalencije. Nerefleksivni (ili strogi) deo relacije $\geq$, kao i stroga parcijalna uređenja uopšte označavaćemo najčešće sa $>$.

Definicija B.21 (Kvazi uređenje) *Za relaciju R skupa S kažemo da je* kvazi uređenje *(eng. quasi ordering) ako je refleksivna i tranzitivna.*

Definicija B.22 (Dobro zasnovana relacija) *Za relaciju R skupa S kažemo da je* dobro zasnovana *(eng. well-founded relation) ako svaki neprazan podskup skupa S ima minimalni element, tj. ako za svaki neprazan podskup W skupa S postoji element $u \in W$ takav da ne postoji element $v \in W$ za koji važi uRv.*

Ako je strogo parcijalno uređenje $>$ skupa S dobro zasnovana relacija (tj. *dobro zasnovano uređenje*) onda ne postoji beskonačan niz $s_1, s_2, s_3, \ldots$ elemenata skupa S takav da važi $s_1 > s_2 > s_3 > \ldots$. Obratno, ako za strogo parcijalno uređenje $>$ skupa S ne postoji beskonačan niz $s_1, s_2, s_3, \ldots$ elemenata skupa S takav da važi $s_1 > s_2 > s_3 > \ldots$ onda je ono dobro zasnovano.

B.3.1 Monotonost i stabilnost

Definicija B.23 (Monotono uređenje) *Za parcijalno uređenje $\geq$ skupa kažemo da je* monotono *(eng. monotonic) ili da ima* svojstvo zamenjivanja *(eng. replacement property) ako za svaka dva terma s i t signature (Σ, X) važi: $s > t$ povlači $f(t_1, \ldots, s, \ldots, t_n) > f(t_1, \ldots, t, \ldots, t_n)$, gde je f funkcijski simbol, t_i termovi signature, $a >$ je nerefleksivni (strogi) deo uređenja $\geq$.*

Teorema B.7 (Manna i Ness) *Sistem za prezapisivanje termova R je zaustavljajući ako i samo ako postoji monotono dobro–zasnovano uređenje $>$ nad skupom baznih termova takvo da za svako pravilo $l \longrightarrow r$ iz R važi $l\sigma > r\sigma$ za svaku supstituciju σ takvu da su $l\sigma$ i $r\sigma$ bazni termovi [77].*

Primena teoreme B.7 zahteva, u jednoj ili drugoj formi, ispitivanje beskonačno mnogo baznih termova (potrebno je proveriti dati uslov za sva pravila i za sve supstitucije φ, kojih obično ima beskonačno mnogo, takve da su $l\varphi$ i $r\varphi$ bazni termovi). To je glavni problem u efektivnoj primeni ove teoreme, a ne definisanje monotonog dobro–zasnovanog uređenja nad skupom baznih termova — takva uređenja postoje (štaviše, postoje i totalna takva uređenja). Jedan od načina primene navedene teoreme zasniva se na funkciji koja svakom baznom termu pridružuje neki prirodan broj (pri čemu se koristi i uobičajeno dobro uređenje nad prirodnim brojevima $>_{\mathbf{N}}$). To ilustruje sledeći primer.

Primer B.8 *Pravila za transformisanje iskazne formule u konjuktivnu normalnu formu mogu biti zadata kao sledeća pravila prezapisivanja:*

$$\neg\neg A \longrightarrow A$$

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

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

$$A \vee (B \wedge C) \longrightarrow (A \vee B) \wedge (A \vee C)$$

$$(B \wedge C) \vee A \longrightarrow (B \vee A) \wedge (C \vee A)$$

Razmotrimo funkciju $|.|$ koja svakoj iskaznoj formuli dodeljuje numeričku vrednost na sledeći način:

$$|\neg A| = 2^{|A|}$$

$$|A \vee B| = |A| \cdot |B|$$

$$|A \wedge B| = |A| + |B| + 1$$

$$|a| = 2$$

gde je a konstanta.

Uređenje $>$ definišemo na sledeći način: važi $l > r$ ako i samo ako $|l\varphi| >_{\mathbf{N}} |r\varphi|$ za svaku supstituciju φ za koju su $l\varphi$ i $r\varphi$ bazni termi.

Lako se pokazuje da iz $|x| >_{\mathbf{N}} |y|$ sledi $|\neg x| >_{\mathbf{N}} |\neg y|$ i slično za ostale logičke veznike, odakle sledi monotonost sistema.

U ovom uređenju važi $\neg\neg A > A$, jer je $|\neg\neg A\varphi| = 2^{|\neg A\varphi|} = 2^{2^{|A\varphi|}} >_{\mathbf{N}} |A\varphi|$. Slične nejednakosti (za sve vrednosti $m, n, p, q \geq 2$) važe za sva pravila sistema:

$$2^{(2^n)} >_{\mathbf{N}} n$$

$$2^{m \cdot n} >_{\mathbf{N}} 2^m + 2^n + 1$$

$$2^{m+n+1} >_{\mathbf{N}} 2^m \cdot 2^n$$

$$m \cdot (p + q + 1) >_{\mathbf{N}} (m \cdot p) + (m \cdot q) + 1$$

$$(p + q + 1) \cdot m >_{\mathbf{N}} (p \cdot m) + (q \cdot m) + 1$$

Svakom od pravila prezapisivanja odgovara po jedna od navedenih nejednakosti. Dakle, zadovoljeni su uslovi teoreme B.7, pa je time dokazano zaustavljanje datog sistema.[6]

Probleme koji se javljaju u efektivnoj primeni teoreme B.7 (potencijalno ispitivanje beskonačno mnogo termova) moguće je izbeći uvođenjem svojstva zatvorenosti za supstituciju (svojstvo pune invarijanse) koje redukuje taj prostor pretrage.

Definicija B.24 (Uređenje zatvoreno za supstituciju) *Za parcijalno uređenje $\geq$ skupa S kažemo da ima* svojstvo zatvorenosti za supstituciju *(eng. closed under substitutions), da ima* svojstvo pune invarijanse *(eng. full invariance property) ili da je* stabilno *(eng. stable) ako iz $s > t$ za svaku supstituciju φ sledi $s\varphi > t\varphi$.*

Definicija B.25 (Zaustavljajuće uređenje) *Za parcijalno, dobro zasnovano uređenje $\geq$ skupa $Ter(\Sigma, X)$ kažemo da je zaustavljajuće uređenje (eng. terminating ordering) ili* uređenje svođenja *(eng. reduction ordering) ako je monotono i zatvoreno za supstitucije.*

Primetimo, ako je $>$ zaustavljajuće uređenje i ako važi $s > t$, onda sve promenljive koje se pojavljuju u t moraju da se pojavljuju i u s. U suprotnom, na primer, iz $f(x) > g(x, y)$ na osnovu svojstva zatvorenosti za supstituciju sledilo bi $g(x, f(x)) > g(x, g(x, f(x)))$, dajući dalje $g(x, f(x)) > g(x, g(x, f(x))) > g(x, g(x, g(x, f(x)))) > \ldots$.

[6] U datom primeru pokazano je da je opisani sistem za prezapisivanje zaustavljajući. Međutim, ovaj sistem nije konfluentan (jer, na primer, $(A \vee B) \wedge (C \vee D)$ ima dve normalne forme). Bio bi "bolji" sistem koji uključuje pravila asocijativnosti i komutativnosti za $\vee$ (i pravila za eliminisanje izraza oblika $A \vee \neg A$. Međutim, ovo ne može biti urađeno bez izmene definicije prezapisivanja, jer komutativnost uvek narušava zaustavljanje. Čak i u slučaju da se dozvoli asocijativnost i komutativnost za $\wedge$ i $\vee$, dokazano je (vidi [109]) da ne postoji ekvivalentan zaustavljajući i konfluentan sistem. Videti i poglavlje B.2.5.

Teorema B.8 (Manna i Ness) *Sistem za prezapisivanje termova R je zaustavljajući ako i samo ako postoji zaustavljajuće uređenje $>$ takvo da za svako pravilo $l \longrightarrow r$ iz R važi $l > r$.*

Za razliku od teoreme B.7, u efektivnoj primeni teoreme B.8 osnovni problem je u definisanju zaustavljajućeg uređenja koje je dovoljno moćno da zadovoljava uslove teoreme za dati sistem za prezapisivanje.

Najjednostavnije vrste zaustavljajućih uređenja su ona zasnovana na "veličini". Neka je $|e|$ broj simbola (veličina) izraza e. Relaciju $>$ možemo definisati na sledeći način: važi $s > t$ ako za svaku supstituciju φ važi $|s\varphi| >_{\mathbf{N}} |t\varphi|$. Ovako definisano uređenje je efektivno izračunljivo; naime, važi $s > t$ ako i samo ako je $|s| >_{\mathbf{N}} |t|$ i nijedna promenljiva se u t ne javlja više puta nego u s (za sistem za prezapisivanje iz primera B.8 moguće je pokazati da je zaustavljajući korišćenjem ovako definisanog uređenja).

Za ovako definisano uređenje $>$ važi $f(x, y) > g(y)$, ali ne važi $h(x, a, b) > f(x, x)$ (jer npr. za $\varphi = \{x \mapsto g(a, a)\}$ ne važi $|h(g(a, a), a, b)| >_{\mathbf{N}} f(g(a, a), g(a, a)))$ niti $f(x, x) > h(x, a, b)$. Dakle, ovo uređenje nije totalno. Pored toga, ovo uređenje je preslabo za mnoge interesantne sisteme, između ostalih, za one koji sadrže pravila tipa $x*(y+z) \longrightarrow (x*y)+(x*z)$ (jer je desna strana ovog pravila "veća" od leve, a ima i više pojavljivanja promenljive x). Ovo uređenje može biti modifikovano tako da različitim simbolima dodeljuje različite težine. Međutim, i u tako modifikovanom sistemu ne može se uspostaviti odnos $>$ između $(x*y)*z$ i $x * (y * z)$ i za sistem koji sadrži pravilo $(x * y) * z \longrightarrow x * (y * z)$ ne može se dokazati zaustavljanje. S druge strane, to može biti prednost u radu sa sistemima koji sadrže pravila koja se odnose na komutativna i asocijativna pravila — u takvim sistemima često nije potrebno praviti razliku između termova koji su jedan od drugog dobijeni primenom komutativnih i asocijativnih pravila. Videti i poglavlje B.2.5.

B.3.2 Uređenja pojednostavljivanja

Jedan od najznačajnijih rezultata u vezi sa uređenjima i zaustavljanjem je Dershowitzeva teorema o uređenjima pojednostavljivanja koja daje dovoljne uslove da je neko uređenje uređenje zaustavljanja. Bez primene ove teoreme potrebno je neposredno pokazati da je uređenje dobro zasnovano, što je često veoma teško.

Definicija B.26 (Uređenje pojednostavljivanja) *Za strogo parcijalno uređenje $>$ kažemo da je* uređenje pojednostavljivanja *(eng. simplification ordering) ako zadovoljava sledeća svojstva:*

1. *$>$ je monotono uređenje;*
2. *$f(t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n) > t_i$;*

Ukoliko postoje funkcijski simboli koji imaju višestruku arnost (obično smatramo da nema takvih simbola), onda je potrebno dodati i sledeći uslov da bi uređenje bilo uređenje pojednostavljivanja: za svaki funkcijski simbol f višestruke arnosti važi: $f(t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n) > f(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$.

Za uređenje pojednostavljivanja važi Dershowitzeva teorema [37, 39] (zasnovana i na rezultatima D. S. Lankforda):

Teorema B.9 *Uređenje pojednostavljivanja je dobro zasnovano.*

Iz ove teoreme proističe i sledeće značajno tvrđenje:

Teorema B.10 *Uređenje pojednostavljivanja zatvoreno za supstituciju je zaustavljajuće uređenje.*

Uređenja pojednostavljivanja se često koriste i ona često vode do totalnog uređenja nad baznim termima. Međutim, totalna uređenja nad baznim termima zasnovana na uređenjima pojednostavljivanja ne mogu uvek biti iskorištena za proveravanje zaustavljanja sistema za prezapisivanje (čak ni baznih sistema za prezapisivanje). To ilustruje sledeći primer:

Primer B.9 *Sistem za prezapisivanje* $R = \{f(a) \longrightarrow f(b), g(b) \longrightarrow g(a)\}$ *je zaustavljajući, ali za njega ne postoji totalno uređenja nad baznim termima i odgovarajuće uređenje pojednostavljivanja* $>$ *takvo da postojanje pravila* $l \longrightarrow r$ *povlači* $l > r$*. Naime, ni* $a > b$ *ni* $b > a$ *ne mogu dovesti do traženog uređenja.*

U opštem slučaju neodlučivo je da li je sistem za prezapisivanje zaustavljajući u odnosu na neko uređenje pojednostavljivanja (to važi čak i za jednočlane sisteme za prezapisivanje [80]).

U mnogim situacijama zaustavljajuća uređenja se definišu induktivno nad strukturom termova, što ih često jednostavno čini monotonim i zatvorenim za supstitucije. Zahtev da postojanje pravila $l \longrightarrow r$ implicira $l > r$ je ono što vodi definisanje zaustavljajućeg uređenja.

B.3.3 Uređenje multiskup staze

Za efektivnu primenu pravila prezapisivanja (kako za primenu Knuth/Bendixove procedure upotpunjavanja tako i u dokazivačima teorema zasnovanim na prezapisivanju) važan je pojam *inkrementalnosti*. Kažemo da uređenje $>'$ proširuje uređenje $>$ ako $x > y$ povlači $x >' y$, ali ne i obratno (tada kažemo i da je uređenje $>'$ moćnije od uređenja $>$). Često je korisno definisati parcijalna uređenja i proširivati ih do moćnijih uređenja. Na prvi pogled to izgleda čudno, jer ako možemo da pokažemo da je sistem R zaustavljajući u odnosu na neko uređenje, onda možemo da pokažemo da je zaustavljajući i u odnosu na neko moćnije uređenje. Međutim, tokom primene Knuth/Bendixove procedure upotpunjavanja, sistem R nije poznat unapred; sistem R se gradi sukcesivno, inkrementalno dodajući nova pravila zasnovana na kritičnim parovima. Svojstvo inkrementalnosti dozvoljava postepeno proširivanje uređenja. Neka od takvih uređenja su tzv. uređenja rekurzivne staze. Postoje dve osnovne varijante ovog uređenja: uređenja multiskup staze[7] i uređenja leksikografske staze.

[7] Originalno definisano uređenje [39] rekurzivne staze zasnovano je na multiskup poređenjima i danas se naziva uređenjem multiskup staze. Termin uređenje rekuzivne staze se danas obično koristi za kombinacije uređenja multiskup staze i uređenja leksikografske staze.

Snagu uređenja multiskup staze oslikava činjenica da za svaku primitivno rekurzivno funkciju postoji sistem za prezapisivanje R čije se zaustavljanje može dokazati korišćenjem uređenja multiskup staze [94]. Važi i obratno: ako se zaustavljanje sistema za prezapisivanje R može dokazati korišćenjem uređenja multiskup staze, onda sistem R izračunava primitivno rekurzivnu funkciju (tj. funkcija koja polazni izraz preslikava u njegovu normalnu formu za sistem R je primitivno rekurzivna) [53]. Za mnoge uobičajene sisteme relativno je jednostavno naći uređenje multiskup staze dovoljno za dokaz zaustavljanja. Međutim, pravila poput $(x * y) * z \longrightarrow x * (y * z)$ zahtevaju uvođenje i pojma takozvanog "statusa" i kombinovanje sa uređenjem leksikografske staze.

Uređenje multiskup staze (kao i uređenje leksikografske staze) zasniva se na relaciji *prethođenja* koja je kvazi uređenje nad funkcijskim simbolima. Sa $g \succ f$ (ili sa $f \prec g$) označavamo da simbol f prethodi simbolu g ($\succ$ označava strogi deo kvazi uređenja $\succeq$). Uređenje multiskup staze $>_{mpo}$ definisaćemo potpunim skupom pravila izvođenja koja se mogu iskoristiti za dokazivanje veza oblika $s >_{mpo} t$. Naime, ukoliko važi $s >_{mpo} t$, onda se to može dokazati u datom sistemu izvođenja. Pri tome, moguće je efektivno konstruisati efikasnu proceduru odlučivanja za tvrđenja oblika $s >_{mpo} t$ korišćenjem datih pravila unazad u stilu dokaza vođenih ciljem (eng. goal directed manner).

Multiskup je, neformalno, skup u kojem se elementi mogu pojavljivati više puta. Formalno, multiskup S je funkcija iz nekog domena D u skup nenegativnih celih brojeva. Kažemo da je multiskup S konačan ako je skup $\{x \mid S(x) > 0\}$ konačan. Kažemo da važi $x \in S$ ako važi $S(x) > 0$. Vrednost $S(x)$ nazivamo višestrukošću elementa x u multiskupu S i ona predstavlja broj pojavljivanja elementa x u multiskupu S. Ako su S i T multiskupovi onda se $S \cup T$ definiše na sledeći način: $S \cup T(x) = S(x) + T(x)$ za svako x. Na osnovu parcijalnog uređenja $>$ nad D može se uvesti parcijalno uređenje $>^{mul}$ nad multiskupovima; to uređenje nazivamo i multiskup proširenjem (eng. a multiset extension) relacije $>$.

Definicija B.27 (Multiskup proširenje) *Ako je $>$ strogo parcijalno uređenje nad skupom D, onda za multiskupove S i T elemenata iz D važi $S >^{mul} T$ ako su zadovoljeni sledeći uslovi:*

1. *multiskupovi S i T su različiti, tj. $\exists x\ S(x) \neq T(x)$;*
2. *postoji multiskup V takav da je $S = S' \cup V$, $T = T' \cup V$ i za svako t iz T' postoji s iz S' takav da važi $s > t$.*

Ispitivanje da li važi $S >^{mul} T$ može se izvesti relativno efikasno brisanjem zajedničkih elemenata iz S i T i, dok je to moguće, ispitivanjem da li važi opisani odnos između S' i T' (uvedenih kao u prethodnoj definiciji). Može se dokazati da ako je relacija $>$ dobro zasnovana, onda je i relacija $>^{mul}$ dobro zasnovana.

Definicija B.28 (Uređenje multiskup staze) *Neka je (Σ, X) signatura i neka je $\succeq$ kvazi uređenje nad skupom funkcijskih simbola. Uređenje multiskup staze $\geq_{mpo}$ definišemo rekurzivno sledećim skupom pravila ($>_{mpo}$ označava strogi*

(nerefleksivni) deo relacije $\geq_{mpo}$, $\succ_{mpo}$ označava strogi (nerefleksivni) deo relacije $\succeq_{mpo}$, $\sim$ je relacija $\succeq \cap \succeq^{-1}$):

1. $$\frac{s_i \geq_{mpo} t \quad za\ neko\ i \in \{1, 2, \ldots, m\}}{f(s_1, \ldots, s_m) >_{mpo} t}$$

2. $$\frac{f \succ g \quad f(s_1, \ldots, s_m) >_{mpo} t_i \quad za\ sve\ t_i,\ i \in \{1, 2, \ldots, m\}}{f(s_1, \ldots, s_m) >_{mpo} g(t_1, \ldots, t_n)}$$

3. $$\frac{f \sim g \quad \{s_1, \ldots, s_m\} >_{mpo}^{mul} \{t_1, \ldots, t_n\}}{f(s_1, \ldots, s_m) >_{mpo} g(t_1, \ldots, t_n)}$$

4. $$\frac{true}{s \geq_{mpo} s}$$

Primer B.10 *Pretpostavimo da je $* > +$. Tada se može dokazati da važi $x * (y + z) >_{mpo} x * y + x * z$ na sledeći način:*

- *da bismo dokazali da važi $x*(y+z) >_{mpo} x*y+x*z$ dovoljno je (na osnovu drugog pravila) dokazati da važi $x*(y+z) >_{mpo} x*y$ i $x*(y+z) >_{mpo} x*z$:*
  - *da bismo dokazali da važi $x*(y+z) >_{mpo} x*y$ dovoljno je (na osnovu trećeg pravila) dokazati da važi $\{x, y + z\} >_{mpo}^{mul} \{x, y\}$. To se može dokazati korišćenjem tvrđenja $y + z >_{mpo} y$, koje na osnovu prvog pravila sledi iz $y \geq_{mpo} y$. Na osnovu četvrtog pravila sledi $y \geq_{mpo} y$.*
  - *analogno se dokazuje da važi $x * (y + z) >_{mpo} x * z$.*

Teorema B.11 (Dershowitz 1982) *Ako je $\succeq$ kvazi uređenje nad skupom funkcijskih simbola, onda je odgovarajuće uređenje multiskup staze zaustavljajuće uređenje i uređenje pojednostavljivanja.*

Za dati sistem za prezapisivanje R i za dato uređenje multiskup staze $>_{mpo}$, na osnovu navedene teoreme (i na osnovu teoreme B.8) sledi da je za zaustavljanje sistema R dovoljno proveriti da za svako pravilo $l \longrightarrow r$ važi $l >_{mpo} r$.

Primer B.11 *Treba dokazati da je sledeći skup pravila zaustavljajući:*

1. $\neg(\neg(x) \longrightarrow x$
2. $\neg(x \vee y) \longrightarrow (\neg x \wedge \neg y)$
3. $\neg(x \wedge y) \longrightarrow (\neg x \vee \neg y)$
4. $x \wedge (y \vee z) \longrightarrow (x \wedge y) \vee (x \wedge z)$
5. $(x \vee y) \wedge z \longrightarrow (x \wedge z) \vee (y \wedge z)$

Kako je svako uređenje multiskup staze $\geq_{mpo}$ uređenje pojednostavljivanja, dovoljno je dokazati da važi $l \geq_{mpo} r$ za svako od datih pet pravila $l \longrightarrow r$.

Za prvo pravilo to je lako, jer je desna strana podterm leve strane i uređenje nad funkcijskim simbolima još uvek nije potrebno. Za drugo pravilo treba da poredimo $\neg(x \vee y)$ i $(\neg x \wedge \neg y)$, što znači treba da poredimo $\neg$ i $\wedge$. Definisanjem $\neg \succ \wedge$ dobija se željeni rezultat: $\neg(x \vee y) >_{mpo} (\neg x \wedge \neg y)$. Kod trećeg pravila uvodimo $\neg \succ \vee$. U četvrtom pravilo treba da poredimo $\wedge$ i $\vee$. Ako definišemo $\wedge \succ \vee$ onda treba dokazati $x \wedge (y \vee z) \geq_{mpo} (x \wedge y)$ i $x \wedge (y \vee z) \geq_{mpo} (x \wedge z)$. U oba slučaja koreni funkcijski simboli su jednaki i treba porediti multiskupove odgovarajućih termova. Dakle, treba proveriti $\{x, (y \vee x)\} \geq^{mul}_{mpo} \{x, y\}$, što je jednostavno, s obzirom da važi $(y \vee z) \geq_{mpo} y$. Drugi potproblem je sličan, pa važi $l \geq_{mpo} r$ i za ovo pravilo. Za peto pravilo analogno se dokazuje da važi $l \geq_{mpo} r$.

Navedeni primer ilustruje kako se definisanje relacije prethođenja može automatizovati. Prednost uređenja multiskup staze je u tome što se relacija prethođenja najčešće nalazi prirodno. Na primer, pretpostavimo da je funkcija f definisana tako da se $f(s)$ definiše preko termova oblika $f(t)$ pri čemu je t jednostavnije od s (npr. t je podterm terma s). Tada je $f(s) = H(f(t_1), \ldots, f(t_n))$. Pretpostavimo da se H može izraziti kao kompozicija već ranije definisanih funkcija. Za sve te funkcije g možemo dodefinisati relaciju $\succ$ tako da je $f \succ g$. Iz $s >_{mpo} t_i$ za sve i, sledi $f(s) >_{mpo} f(t_i)$ za sve i, pa je, na osnovu uređenja multiskup staze i $f(s) >_{mpo} H(f(t_1), \ldots, f(t_n))$. Time je dokazano zaustavljanje skupa pravila koje sadrži pravilo $f(s) \longrightarrow H(f(t_1), \ldots, f(t_n))$

Primer B.12 *Razmotrimo sistem za prezapisivanje koji računa faktorijel funkciju:*

$$fact(s(x)) \longrightarrow times(s(x), fact(x))$$

$$fact(0) \longrightarrow s(0)$$

$$times(0, x) \longrightarrow 0$$

$$times(s(x), y) \longrightarrow plus(y, times(x, y))$$

$$plus(0, x) \longrightarrow x$$

$$plus(s(x), y) \longrightarrow s(plus(x, y))$$

Za ovaj sistem relaciju prethođenja $\succ$ nad funkcijskim simbolima definišemo na sledeći način: $fact \succ times \succ plus \succ s \succ 0$. Ovako definisanje je prirodno jer je $'fact'$ definisano preko $'times'$, $'times'$ preko $'plus'$, a $'plus'$ preko $'s'$ i $'0'$. Korišćenjem ove relacije prethođenja i odgovarajućeg uređenja multiskup staze, za sva pravila $l \longrightarrow r$ datog sistema može se dokazati da važi $l > r$, što povlači da je dati sistem za prezapisivanje zaustavljajući.

Postoje zaustavljajući sistemi za prezapisivanje za koje se zaustavljanje ne može dokazati korišćenjem nekog uređenja multiskup staze. Standardni primer je definicija Akermanove funkcije:

1. $ack(0, y) \longrightarrow succ(y)$
2. $ack(succ(x), 0) \longrightarrow ack(x, succ(0))$
3. $ack(succ(x), succ(y)) \longrightarrow ack(x, ack(succ(x), y)$

Poznato je da Akermanova funkcija nije primitivno rekurzivna [31]. Dakle, postoji izvođenje čija dužina ne može biti ograničena ni jednom primitivno rekurzivnom funkcijom.

Uređenje multiskup staze ima svojstvo inkrementalnosti u odnosu na relaciju prethođenja $\succ$ nad funkcijskim simbolima. Naime, neka je $>_{mpo(\succ)}$ uređenje multiskup staze za uređenje $\succ$ nad funkcijskim simbolima; tada, ako je $\succ'$ proširenje relacije $\succ$, onda je $>_{mpo(\succ')}$ proširenje relacije $>_{mpo(\succ)}$. Ovo svojstvo je veoma važno u primeni Knuth/Bendixove procedure upotpunjavanja kao i u dokazivačima zasnovanim na prezapisivanju: kada se uvodi novo pravilo prezapisivanja moguće je relaciju prethođenja dodefinisati kako je potrebno ne narušavajući pri tom svojstva do tada izgrađenog sistema.

B.3.4 Uređenje leksikografske staze

Uređenje multiskup staze nije dovoljno za sisteme koji sadrže pravila tipe $(x * y) * z \longrightarrow x * (y * z)$ ili za sisteme koji odgovaraju primitivno rekurzivnim funkcijama. U tom slučaju uređenje je potrebno modifikovati tako da se termovi porede "leksikografski".

Definicija B.29 (Leksikografsko proširenje) *Ako je $>$ uređenje, onda se njegovo leksikografsko proširenje $>^{lex}$ nad torkama fiksne dužine definiše sledećim pravilima:*

1. $$\frac{s_1 > t_1}{(s_1, \ldots, s_m) >^{lex} (t_1, \ldots, t_n)}$$
2. $$\frac{s_1 = t_1 \quad (s_2, \ldots, s_m) >^{lex} (t_2, \ldots, t_n)}{(s_1, \ldots, s_m) >^{lex} (t_1, \ldots, t_n)}$$
3. $$\frac{true}{(s_1, \ldots, s_m) >^{lex} ()}$$

Može se dokazati da je relacija $>^{lex}$ dobro zasnovana, ako je relacija $>$ dobro zasnovana.

Definicija B.30 (Uređenje leksikografske staze) *Neka je $\Sigma = (S, F, \tau)$ signatura, $s, t \in Ter(\Sigma, X)$ termi i $\succeq$ kvazi uređenje nad skupom fukcijskih simbola. Uređenje leksikografkske staze $>_{lpo}$ definiše se rekurzivno sledećim skupom pravila ($>$ označava strogi (nerefleksivni) deo relacije $\geq$, $\succ$ označava strogi (nerefleksivni) deo relacije $\succeq$, $a \sim$ je relacija $\succeq \cap \succeq^{-1}$):*

1.

$$\frac{s_i \geq_{lpo} t \quad za\ neko\ i \in \{1,2,\ldots,m\}}{f(s_1,\ldots,s_m) >_{lpo} t}$$

2.

$$\frac{f \succ g \quad f(s_1,\ldots,s_m) \geq_{lpo} t_i \quad za\ sve\ i \in \{1,2,\ldots,m\}}{f(s_1,\ldots,s_m) >_{lpo} g(t_1,\ldots,t_n)}$$

3.

$$\frac{f \sim g\ (s_1,\ldots,s_m) >^{lex}_{lpo} (t_1,\ldots,t_n)\ f(s_1,\ldots,s_m) \geq_{lpo} t_j\ za\ sve\ j \in \{1,2,\ldots,m\}}{f(s_1,\ldots,s_m) >_{lpo} g(t_1,\ldots,t_n)}$$

4.

$$\frac{true}{s \geq_{lpo} s}$$

5.

$$\frac{x \in X \quad t \not\equiv x}{t \geq_{lpo} x}$$

U drugom i trećem pravilu uslov $f(s_1,\ldots,s_m) \geq_{lpo} t_i \quad$ za sve $i \in \{1,2,\ldots,m\}$ može da izgleda čudno i prejako. Ipak, iako se termovi u osnovi porede leksiografski, ne želimo da važi npr. $f(h(x),x) > f(x,f(h(x),x))$ samo zbog toga što je $h(x) > x$. Ovaj primer ilustruje razloge za uvođenje pomenutog uslova.

Teorema B.12 (Dershowitz 1982) *Ako je $\succeq$ kvazi uređenje nad skupom funkcijskih simbola, onda je uređenje leksikografske staze uređenje pojednostavljivanja i zaustavljajuće uređenje, samo ako svaki funkcijski znak ima fiksnu arnost.*

Zanimljivo je da je korišćenjem uređenja leksikografske staze moguće dokazati zaustavljanje izračunavanja Ackermannove funkcije, što znači da u ovom slučaju (za razliku od uređenja multiskup staze) ne stoji ograničenje na primitivno rekurzivne funkcije.

B.3.5 Uređenje rekurzivne staze sa statusom

Postoji više uređenja koja su između uređenja multiskup staze i uređenja leksikografske staze. Ona se obično nazivaju uređenja rekurzivne staze sa *statusom*. To su uređenja pojednostavljivanja zasnovana na relaciji prethođenja $\succeq$ nad funkcijskim simobolima i na status funkciji τ kao parametrima. Za svaki funkcijski simbol f njegov status određuje uređenje po kojem će njegovi termi $f(s_1,s_2,\ldots,s_m)$ i $f(t_1,t_2,\ldots,t_m)$ biti poređeni. Različite mogućnosti su obično multiskup proširenje i leksikografsko proširenje uređenja (pri čemu se podtermi mogu porediti sleva nadesno ili zdesna nalevo). Uređenja rekurzivne staze sa statusom koriste se u većini dokazivača koji se u manjoj ili većoj meri oslanjaju na sisteme za prezapisivanje. U daljem tekstu opisaćemo ukratko uređenje koje se koristi u sistemu *Oyster/Clam* [18].

Relaciju prethođenja $\succeq$ nad funkcijskim simobolima i status funkciju τ zajedno nazivamo *registrom* (eng. registry) i označavamo sa $\rho = (\succeq, \tau)$. Uređenje nad termovima indukovano registrom ρ označavamo sa $>_\rho$ (strogi deo), odnosno sa $\geq_\rho$ (kada je uključen i refleksivni deo). Registri se mogu određivati inkrementalno — to znači da nije nužno odrediti registar unapred: novo pravilo prezapisivanja može da se doda u sistem za prezapisivanje i da se pri tom proširi registar ako i kada je potrebno u cilju održavanja zaustavljanja sistema.

Kao što je već rečeno, status funkcija je preslikavanje iz skupa funkcijskih simbola u skup status indikatora. Obično postoje bar dva status indikatora:

- $\otimes$ koji označava multiskup proširenje)
- δ koji označava leksikografsko proširenje i to kao jednu funkciju permutacije argumenata. Obično se koriste dve funkcije permutacije:
  - $\oplus$ — identička permutacija;
  - $\ominus$ — permutacija u kojoj su argumenti poređani zdesna nalevo.

Dodatno, dozvoljavamo i nedefinisan status $\odot$, i onda je funkcija τ preslikavanje iz skupa funkcijskih simbola u skup $\{\otimes, \oplus, \ominus, \odot\}$. Status određuje kako se porede torke argumenata za neki funkcijski simbol; on određuje oblik torke argumenata $(t_1, t_2, \ldots, t_n)$ i to na način opisan sledećom definicijom.

Definicija B.31 (Status funkcija)

$$(t_1, t_2, \ldots, t_n)^{\oplus} = (t_1, t_2, \ldots, t_n)$$

$$(t_1, t_2, \ldots, t_n)^{\ominus} = (t_n, t_{n-1}, \ldots, t_1)$$

$$(t_1, t_2, \ldots, t_n)^{\otimes} = [t_1, t_2, \ldots, t_n]$$

Definicija B.32 (Konzistentnost registra) *Registar $\rho = (\succeq, \tau)$ ako i samo ako važe sledeći uslovi:*

1. *ako f i g imaju definisan status i ako važi $f \sim g$, onda važi $\tau(f) = \tau(g)$;*
2. *iz $f \succeq g$, $g \succeq h$ i $f \not\sim g$ (ili $g \not\sim h$) sledi $f \not\sim h$.*

Za dati registar ρ možemo definisati odgovarajuće uređenje $\geq_\rho$ (i njegov strogi deo $>_\rho$) nad baznim termovima rekurzivno, na sličan na koji je to urađeno za uređenje multiskup i uređenje leksikografske staze. Dva terma $s = f(s_1, s_2, \ldots, s_m)$ i $t = g(t_1, t_2, \ldots, t_n)$ i $f \succeq g$ se mogu porediti samo ako koreni funkcijski simboli f i g imaju definisan isti status.

Definicija B.33 ($\geq_\rho$, $>_\rho$) *Za dati konzistentan registar $\rho = (\succeq, \tau)$ definišemo uređenje $>_\rho$ nad baznim termovima na sledeći način. Ako je $s = f(s_1, s_2, \ldots, s_m)$ i $t = g(t_1, t_2, \ldots, t_n)$ i $f \succeq g$, onda važi $s >_r$ hot ako i samo ako je ispunjen jedan od sledećih uslova:*

- *$s_i \geq_\rho t$ za neko $i \in \{1, 2, \ldots, m\}$;*

- $f \succ g$ *i* $s >_\rho t_i$ *za svako* $i \in \{1, 2, \ldots, n\}$*;*
- $f \sim g$, $\tau(f) = \tau(g) = \otimes$ *i* $[s_1, s_2, \ldots, s_m] >_\rho^{mul} [t_1, t_2, \ldots, t_n]$*;*
- $f \sim g$, $\tau(f) = \tau(g) = \oplus$, $(s_1, s_2, \ldots, s_m) >_\rho^{lex} (t_1, t_2, \ldots, t_n)$ *i* $s >_\rho t_i$ *za svako* $i \in \{1, 2, \ldots, n\}$*;*
- $f \sim g$, $\tau(f) = \tau(g) = \ominus$, $(s_m, s_{m-1}, \ldots, s_1) >_\rho^{lex} (t_n, t_{n-1}, \ldots, t_1)$ *i* $s >_\rho t_i$ *za svako* $i \in \{1, 2, \ldots, n\}$*;*

U praktičnoj primeni, kada je potrebno da se sistem za prezapisivanje inkrementalno proširuje novim pravilima (a da se pri tome zadržava zaustavljanje) navedena definicija se proširuje još jednim slučajem koji objedinjuje slučajeve $f \sim g$ i $f \succ g$. Ukoliko je prihvatljivo uvođenje bilo koje od veza $f \sim g$ ili $f \succ g$, onda se uz korišćenjem parcijalne informacije $f \succeq g$, proverava odgovarajuća unija uslova.

Navedeno uređenje definisano je samo za bazne termove i ono može da se proširi do stabilnog uređenja nad svim termovima: sve promenljive smatraju se posebnim konstantama koje nisu uporedive u datom uređenju. U tom proširenju datog uređenja treba da za svaku promenljive x važi i $x \sim x$, $\tau(x) = \otimes$.

Korišćenjem Dershowitzevih teorema za uređenja multiskup i leksikografske može se dokazati da važi naredna teorma:

Teorema B.13 *Ako je* $\succeq$ *kvazi uređenje nad skupom funkcijskih simbola, onda je uređenje rekurzivne staze sa statusom pojednostavljivanja i zaustavljajuće uređenje.*

U praktičnim primenama, registar se računa dinamički, kako se sistemu dodaju nova pravila prezapisivanja. Počinje se sa nekim inicijalnim registrom i on se dinamički proširuje dodeljivanjem statusa funkcijskim simbolima koje ga nisu imale i/ili proširivanjem relacije prethođenja $\succeq$. Inicijalno τ ima vrednost $\odot$ za sve funkcijske simbole (izuzev za simbole arnosti 0 koji predstavljaju promenljive). Registar se može proširiti samo ako pri tome on zadržava svojstvo konzistentnosti. Tačke izbora u stablu dokaza se javljaju kada je

- moguće odabrati $f \sim g$ ili $f \succ g$;
- treba dodeliti neki status funkcijskim simbolima f i g.

Očigledno, često je to moguće uraditi na više načina. Zbog toga, u cilju sužavanja prostora pretraživanja može se koristiti svojstvo minimalnosti. Proširenje e_1 registra je manje nego proširenje e_2 ako e_1 može biti prošireno do e_2. Međutim, izračunavanje minimalnog proširenja je skupo, pa se pretraživanje obično pojednostavljuje uvođenjem jednostavnog zahteva da se pokušava prvo sa proširivanjem relacije $\succeq$, a onda sa dodefinisanjem funkcije τ. U celom postupku, u slučaju raspolaganja delimičnom informacijom, $\succeq$ se obrađuje kao konjunkcija dva slučaja ($\succ$ i *sim*). Slično, $\odot$ se tretira kao konjunkcija $\otimes$, $\oplus$ i $\ominus$. U oba slučaja, ukoliko konjunkcija ne može biti potvrđena, potpuna određenost (uvođenje ili $\sim$ ili $\succ$ za relaciju uređenja) je neophodna kako bi dokaz bio nastavljen.

B.3.6 Knuth/Bendixovo uređenje

Jedan od značajnih tipova uređenja nad termovima je Knuth/Bendixovo uređenje opisano u okviru njihove procedure upotpunjavanja [70]. To uređenje je uopšteno uređenje po dužini termova i to uopštenje ogleda se u težinama dodeljenim funkcijskim simbolima.

Definicija B.34 (Knuth/Bendixovo uređenje) *Neka je $\Sigma = (S, F, \tau)$ signatura i neka je $>$ relacija prethođenja nad funkcijskim simbolima.*

Knuth/Bendixovo uređenje $>_{KB}$ definiše se na osnovu relacije $>$ i težinske funkcije $\phi : (F \cup X) \mapsto \mathbf{N}$ koja ima osobine: $\phi(f_n) = 0$ za samo jedan izdvojeni funkcijski simbol $f_n \in F$ arnosti jedan i za sve ostale funkcijske simbole $f \in F$ važi $f_n > f$. Za neku fiksiranu vrednost $\mu \in \mathbf{N}$ težinska funkcija ϕ mora da zadovoljava uslove:

1. *$\phi(x) = \mu > 0$ za svaku promenljivu $x \in X$;*
2. *$\phi(f) \geq \mu$, ako je f konstanta (tj. ako je $f \in F_0$);*
3. *$\phi(f) > 0$ ako je simbol $f \in F$ arnosti 1 i različit od f_n.*
4. *$\phi(f_n) \geq 0$ ako je $f_n \in F$ arnosti 1 i za sve ostale funkcijske simbole $f \in F$ važi $f_n > f$.*
5. *$\phi(f) \geq 0$ u svim ostalim slučajevima.*

Vezom $\phi(f(t_1, t_2, \ldots, t_n) = \phi(f) + \phi(t_1) + \ldots + \phi(t_n)$ preslikavanje ϕ se kanonski produžava na termove $s, t \in Ter(\Sigma, X)$.

Za termove $s = f(s_1, s_2, \ldots, s_m)$ i $t = g(t_1, t_2, \ldots, t_n)$ važi $s >_{KB} t$ ako i samo ako je zadovoljen jedan od sledećih uslova:

(a) za svaku promenljivu $x \in X$ njen broj pojavljivanja je u termu s jednak ili veći nego u termu t i $\phi(s) > \phi(t)$.

(b) za svaku promenljivu $x \in X$ njen broj pojavljivanja je u termu s jednak je broju pojavljivanja u termu t i $f > g$.

(c) za svaku promenljivu $x \in X$ njen broj pojavljivanja je u termu s jednak je broju pojavljivanja u termu t, važi $f = g$ i $(s_1, s_2, \ldots, s_m) >_{KB}^{lex} (t_1, t_2, \ldots, t_n)$.

Može se dokazati da je uređenje $>_{KB}$ uređenje zaustavljanja. Ono je uspešno u mnogim standardnim situacijama. Ipak, postoje i primeri u kojima ovo uređenje nije pogodno. Razmotrimo jedan takav primer: pretpostavimo da treba pokazati da je sistem za prezapisivanje $\{x * ((-y) * y) \longrightarrow -(y * y) * x\}$ (ili $\{*(x, *(-y, y)) \longrightarrow *(-(*(y, y)), x)\}$ u prefiksnom zapisu) zaustavljajući. Jedino uslov (c) iz date definicije uređenja $>_{KB}$ može biti iskorišćen, ali bi onda moralo da važi i $(x, *(-y, y)) >_{KB}^{lex} (-(*(y, y)), x)$, odakle sledi da bi moralo da važi i $x >_{KB}^{lex} -(*(y, y))$ što je nemoguće (zbog uslova o promenljivama u definiciji leksikografksog uređenja). Postoji više varijanti originalnog Knuth/Bendixovog uređenja.

B.4 Knuth/Bendixova procedura upotpunjavanja

Kanonski sistemi za prezapisivanje su značajni jer mogu da služe kao procedure odlučivanja za jednakosne teorije. Naime, ukoliko je neki kanonski sistem za prezapisivanje zasnovan na skupu jednakosti E, onda važi $E \models s = t$ ako i samo ako termovi s i t imaju iste kanonske forme. Osnovni problem je u tome što nije za svaki skup jednakosti moguće konstruisati odgovarajući kanonski sistem za prezapisivanje[8]. Čak i kada je to teorijski moguće, postavlja se pitanje efektivne izgradnje takvog sistema. Pored toga, postavlja se i pitanje da li je (i kada je) moguće transformisati sistem za prezapisivanje koji nije kanonski u njemu ekvivalentan kanonski sistem. Knuth/Bendixova procedura upotpunjavanja daje delimičan odgovor na ova pitanja. Ta procedura je vođena idejom dodavanja novih pravila sistemu, pri čemu treba da se čuva zaustavljanje i, ukoliko je moguće, dobije konfluentnost sistema.

Na osnovu leme o kritičnom paru, za ispitivanje konfluentnosti zaustavljajućeg sistema za prezapisivanje dovoljno je ispitivanje njegovih kritičnih parova. Ako postoji kritični par $\langle p, q\rangle$ takav da su normalne forme izraza p i q različite, onda se u sistem dodaje ili pravilo $p' \longrightarrow q'$ ili pravilo $q' \longrightarrow p'$ (gde su p' i q' neke normalne forme izraza p i q) i proverava se da li je novodobijeni sistem zaustavljajući. Nakon toga se ceo proces ponavlja iz početka. Opisani postupak ne mora uvek da staje. Ukoliko postupak staje, onda je dobijen sistem kanonski.

Najpre dajemo pojednostavljenu verziju Knuth/Bendixove procedure upotpunjavanja (ulaz je skup jednakosti E nad signaturom (Σ, X) i zaustavljajuće uređenje $>$, a izlaz odgovarajući kanonski sistem za prezapisivanje) [61, 55]:

1. Za svaku jednakost $s = t$ iz E uradi sledeće:
   - ako je $s > t$ dodaj pravilo $s \longrightarrow t$ u skup pravila R_0 (skup R_0 je inicijalno prazan);
   - ako je $t > s$ dodaj pravilo $t \longrightarrow s$ u skup pravila R_0;
   - inače, prekini izvršavanje procedure sa neuspehom.
2. Za sve kritične parove (t_1, t_2) sistema R_i izračunaj normalne forme $t_1 \downarrow$ i $t_2 \downarrow$ korišćenjem pravila iz R_i. Neka je $s = t_1 \downarrow$ i $t = t_2 \downarrow$.
   - ako su termovi s i t identični, ne radi ništa;
   - ako je $s > t$ dodaj pravilo $s \longrightarrow t$ u skup pravila R_i^c (skup R_i^c je inicijalno prazan);
   - ako je $t > s$ dodaj pravilo $t \longrightarrow s$ u skup pravila R_i^c;
   - inače, prekini izvršavanje procedure sa neuspehom.

   Sva pravila dobijena na opisani način čine skup R_i^c i skup pravila R_{i+1} je definisan na sledeći način $R_{i+1} = R_i \cup R_i^c$.

[8] Očigledno, ne postoji odgovarajući kanonski sistem za prezapisivanje u onim slučajevima kada je jednakost zasnovana na polaznom skupu jednakosti neodlučiva.

3. Ako je $R_i^c = \emptyset$, rezultat procedure je kanonski sistem $R = R_i$, inače idi na korak 2.

Ako je skup jednakosti E konačan, onda opisana procedura može da stane uspešno (dajući kanonski sistem R) ili neuspešno (tada se može probati sa nekim drugim uređenjem). Ako algoritam ne staje, onda je beskonačna unija $\bigcup_{i\geq 0} R_i$ odgovarajući kanonski sistem za prezapisivanje. Dakle, algoritam može da uspe (eng. succeed), da ne uspe (eng. fail) ili da se ne zaustavlja (eng. loop). Upotpunjavanje ne uspeva ako neku jednakost nije moguće usmeriti u skladu sa datim uredjenjem; u tom slučaju može se koristiti sledeća varijanta algoritma (koju zovemo *nepropadajuće upotpunjavanje* (eng. unfailing completion)): ako neki kritični par (s, t) nije moguće orijentisati, onda se jednakost $s = t$ dodaje u sistem umesto pravila prezapisivanja. Nakon toga, novi kritični parovi se formiraju i na osnovu postojećih pravila prezapisivanja i na osnovu jednakosti uvršćenih u sistem.

Korektnost opisane procedure dokazuje se jednostavno: za svako pravilo $l \longrightarrow r \in R_i$ važi $l > r$ na osnovu konstrukcije pravila, pa je sistem $\bigcup_{i\geq 0} R_i$ zaustavljajući. Za svako pravilo $l \longrightarrow r \in R_i$ važi $E \models l = r$, pa to važi za svako pravilo iz R. Za svaki kritični par $\langle l, r\rangle$ važi $l \downarrow\equiv r \downarrow$, pa je sistem lokalno konfluentan, odakle sledi da je i kanonski.

Navedeni postupak izložen je prvi put u radu [70] i u tom radu uglavnom je primenjivan na teoriju grupa. Razmotrimo standardni primer, aksiome teorije grupa, za ilustraciju Knuth/Bendixove procedure.

Neka je dat skup jednakosti E:

1. $e \cdot x = x$ ("Postoji levi neutral, e.")
2. $x^{-1} \cdot x = e$ ("Za svako a postoji levi inverz u odnosu na e.")
3. $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ ("Operacija $\cdot$ je asocijativna.").

Korišćenjem Knuth/Bendixovog uređenja $>_{KB}$ sa ${}^{-1} \succ \cdot \succ e$ i $\phi({}^{-1}) = \phi(\cdot) = 0$ navedene jednakosti daju sledeći skup pravila za prezapisivanje:

1. $e \cdot x \longrightarrow x$
2. $x^{-1} \cdot x \longrightarrow e$
3. $(x \cdot y) \cdot z \longrightarrow x \cdot (y \cdot z)$

Prva dva pravila nužno su orijentisana na navedeni način, bez obzira na izabrano uređenje (u prvom pravilu levu stranu ne bi mogla da predstavlja promenljiva, a u drugom pravilu skup promenljivih na desnoj strani pravila mora da bude podskup skupa promenljivih na levoj strani pravila). Orijentacija trećeg pravilo indukovana je Knuth/Bendixovim uređenjem.

Prva dva pravila ne mogu se superponirati, pa dakle, ne daju kritični par. Pravila 1 i 3 daju kritični par $\langle x \cdot z, e \cdot (x \cdot z)$. Međutim, primenom pravila 1 term $e \cdot (x \cdot z)$ se prezapisuje u $x \cdot z$ (dakle, par $\langle 1, 3\rangle$ je konvergentan, pa se nijedno pravilo ne dodaje u sistem.

Pravila 2 i 3 imaju kritični par $\langle e \cdot z, x^{-1} \cdot (x \cdot z)\rangle$. U sistemu R_0 elementi ovog para imaju normalne forme z i $x^{-1} \cdot (x \cdot z)$, pa se dodaje novo pravilo

4 $x^{-1} \cdot (x \cdot z) \longrightarrow z$

Pravilo 4 se dodaje skupu R_0^c i skup R_1 sadrži pravila 1, 2, 3 i 4. Procedura dalje daje:

5 $(y^{-1})^{-1} \cdot e \longrightarrow y$ (iz 2 i 4)

6 $e^{-1} \cdot z \longrightarrow z$ (iz 1 i 4)

7 $(y^{-1})^{-1} \cdot x) \longrightarrow y \cdot x$ (iz 3 i 5)

8 $y \cdot e \longrightarrow y$ (iz 7 i 5)

9 $(y^{-1})^{-1} \longrightarrow y$. Sada leva i desna strana pravila 5 mogu da se svedu na istu normalnu formu, pa pravilo 5 može da bude izostavljeno. Slično, može da bude izostavljeno i pravilo 7.

10 $e^{-1} \longrightarrow e$ (iz 8 i 6). Može da bude poništeno pravilo 6.

11 $y \cdot y^{-1}$ (iz 9 i 2)

12 $y \cdot (y^{-1} \cdot x) \longrightarrow e$ (iz 3 i 11).

13 $x \cdot (y \cdot (x \cdot y)^{-1}) \longrightarrow e$ (iz 3 i 11)

14 $y \cdot (x \cdot y)^{-1} \longrightarrow x^{-1}$ (iz 4 i 13). Može da bude poništeno pravilo 13.

15 $(x \cdot y)^{-1} \longrightarrow y^{-1} \cdot x^{-1}$ (iz 4 i 14). Može da bude poništeno pravilo 14.

Rezultat primene Knuth/Bendixove procedure je sledeći sistem prezapisivanja:

1 $e \cdot x = x$

2 $x^{-1} \cdot x = e$

3 $(x \cdot y) \cdot z = x \cdot (y \cdot z)$

4 $x^{-1} \cdot (x \cdot z) \longrightarrow z$

6 $e^{-1} \cdot z \longrightarrow z$

8 $y \cdot e \longrightarrow y$

9 $(y^{-1})^{-1} \longrightarrow y$

10 $e^{-1} \longrightarrow e$

11 $y \cdot y^{-1}$

12 $y \cdot (y^{-1} \cdot x) \longrightarrow e$

15 $(x \cdot y)^{-1} \longrightarrow y^{-1} \cdot x^{-1}$

Navedeni sistem je kanonski i logički je ekvivalentan polaznom sistemu. Dobijeni konfluentan sistem je dovoljan za rešavanje problema reči za slobodne grupe bez jednakosti: za dve reči formirane operatorima $\cdot$ i $^{-}$ i konstantom e može se dokazati da su jednake kao posledica polaznih aksioma ako i samo se primenom novodobijenog sistema svode na istu nesvodljivu reč (bez obzira na poredak primene pravila).

U prethodnom primeru brisana su ona pravila za koje može da se pokaže da su suvišna. Navedena, osnovna verzija procedure upotpunjavanja ne sadrži brisanja suvišnih pravila, ali se ono obično uključuje u sve implementacije procedure upotpunjavanja.

Efekat procedure upotpunjavanja opisuje sledeća teorema:

Teorema B.14 *Ako se procedura upotpunjavanja koja kreće od sistema za prezapisivanje R i koristi uvek isto uređenje (ili uređenje čija moć raste) ne zaustavlja sa neuspehom i ako važi $R^{=} \models s = t$, onda će ona u konačnom broju koraka proizvesti "dolinski dokaz" za $s = t$. Dodatno, ako je S skup pravila koji je generisan (i nije brisan) tokom upotpunjavanja, onda je on konfluentan i ekvivalentan sistemu R.*

U navedenoj teoremi $R^{=}$ označava skup jednakosti koje odgovaraju pravilim sistema R. "Dolinski dokaz" je dokaz oblika $\longrightarrow\longrightarrow\longrightarrow \ldots \longleftarrow\longleftarrow\longleftarrow$ i navedeno tvrđenje da u S postoji "dolinski dokaz" za $s = t$ znači da se s i t mogu prezapisati u isti term. Kako je S konfluentan, to znači da s i t imaju jedinstvene S-normalne forme koje su identične.

Moguće je transformisati kanonski sistem za prezapisivanje koji nije redukovan u redukovani sistem. O tome govori naredno tvrđenje koje može biti iskorišćeno nakon upotpunjavanja.

Teorema B.15 (Procedura međusvođenja) *Neka je R kanonski sistem za prezapisivanje i neka su skupovi R_0 i R_1 definisani na sledeći način:*

$R_0 = \{l \longrightarrow r \downarrow \mid l \longrightarrow r \in R\}$

$R_1 = \{l \longrightarrow r \mid l \longrightarrow r \in R_0$ *i ne postoji pravilo $l' \longrightarrow r'$ iz R_0 takvo da je $l'\phi$ podterm terma l za neku supstituciju ϕ.*

Tada važi: R_1 je svedeni kanonski sistem ekvivalentan sistemu R.

Standardne implementacije procedure upotpunjavanja rade međusvođenje tokom upotpunjavanja, a ne tek nakon njega. Pored toga, mnoge varijante procedure upotpunjavanja su efikasnije od navedene, osnovne varajante. Neke od njih koriste posebne strategije za izbor pravila koje daju kritične parove ili koriste međusvođenje pravila pre traganja za novim kritičnim parovima. Kao ilustraciju složenijih varijanti procedure upotpunjavanja navodimo sledeću proceduru (ulaz je skup jednakosti E nad signaturom (Σ, X) i zaustavljajuće uređenje $\geq$, a izlaz odgovarajući kanonski sistem za prezapisivanje):

```
R:=0
while E ≠ 0 do
  izaberi jednakost s = t iz E;
  svedi s i t na njihove normalne forme s↓ i t↓
  pravilima iz tekućeg skupa R;
  if s↓= t↓ then
    E = E \ {(s,t)}
  else
    case of
      s↓> t↓ then
        α = s↓ i β = t↓;
      t↓> s↓ then
        α = t↓ i β = s↓;
      else prekini sa neuspehom
    end case
  end if
  R = {γ ⟶ δ↓ | γ ⟶ δ ∈ R i δ↓ je nesvodljiv moduo R ∪ {α ⟶ β}}
;
  CP = {(p,q) | ⟨p,q⟩ je kritični par nekog pravila iz R i α ⟶ β};
  E = E ∪ CP ∪ {(γ,δ)|γ ⟶ δ ∈ R i γ je svodljiv pravilom α ⟶ β}\
{(s,t)};
  R = R ∪ {α ⟶ β} \ {γ ⟶ δ|γ je svodljiv pravilom α ⟶ β}
end while
```

Od različitih pristupa procesu upotpunjavanja izdvojimo i onaj koji ne opisuje upotpunjavanje kao proceduru, već opisuje svaki korak procesa upotpunjavanja kao primenu nekog pravila izvođenja [5]. Na taj način moguće je preciznije rezonovati o proceduri upotpunjavanja i o njenim osobinama kao što je saglasnost. Ta preciznost se ogleda u podizanju tog rezonovanja na apstraktniji nivo koji ne zavisi od konkretne implementacije.

B.5 Kontekstualno prezapisivanje

Konteksualno prezapisivanje (eng. contextual rewriting) [124] je uopštenje uslovnog prezapisivanje i predstavlja moćno pravilo izvođenja. Da bi se pojednostavio term t kontekstualnim prezapisivanjem, pored skupa uslovnih pravila prezapisivanja R, koristi se i skup jednakosti C koji zovemo *kontekst* (eng. context) terma t. Ako je kontekst prazan, kontekstualno prezapisivanje se svodi na uslovno prezapisivanje. U automatskom dokazivanju teorema, kada se dokazuje nezadovoljivost konjunkcije litetala, u prezapisivanju jednog literala kao kontekst se koriste preostali literali; u direktnom dokazu, kada se dokazuje da važi disjunkcija literala, u prezapisivanju jednog literala kao kontekst se koriste negacije preostalih literala. U nastavku dajemo formalnu definiciju kontekstualnog prezapisivanja.

Definicija B.35 Konstantno kongruentno zatvorenje $\simeq_C$ *generisano skupom jednakosti* C *je minimalna relacija ekvivalencije koja zadovoljava sledeće uslove:*

- *za svaku jednakost* $t_1 = t_2$ *of* C*, važi* $t_1 \simeq_C t_2$*;*
- *ako je* $t_1 \simeq_C t_2$*, onda važi* $f(\ldots t_1 \ldots) \simeq_C f(\ldots t_2 \ldots)$*.*

Definicija B.36 *Za dati skup jednakosti* C *i za skup uslovnih pravila prezapisivanja* R*, prezapisivanje u kontekstu* C *se definiše rekurzivno na sledeći način: term* t *se prezapisuje u term* t' *kontekstualnim prezapisivanjem korišćenjem* R *i* C *(i pišemo* $t \longrightarrow_{C,R} t'$*) ako važi:*

- $t \simeq_C t'$ *ili*
- *postoji podterm* t'' *terma* t*, pravilo prezapisivanja* $l \longrightarrow r$ **if** $\{p_1, p_2, \ldots, p_k\}$ *u skupu* R *i supstitucija* σ *takvi da je*
  - $l\sigma = t''$
  - *za svako* p_i*,* $1 \leq i \leq k$*,* $p_i\sigma \rightarrow^*_{C,R} true$
  - *term* t' *jednak je termu* t *u kojem je podterm* t'' *zamenjen termom* $r\sigma$*.*

  Kažemo da je pravilo $l \longrightarrow r$ **if** $\{p_1, p_2, \ldots, p_k\}$ primenljivo *ako su zadovoljena pravila (1) i (2).*

Kontekstualno prezapisivanje može da se koristi i u deduktivnom i u induktivnom dokazivanju teorema. U deduktivnom dokazivanju, kontekstualno prezapisivanje je moćno pravilo koje strogo uključuje nekoliko pravila pojednostavljivanja kao što su brisanje tautologija, uključivanje, demodulacija, ali i pravila kao što je augmentacija (ili njegove varijante).

Primer B.1 *Neka* R *sadrži definicije za* $+$*,* $\cdot$*,* nzd *za prirodne brojeve, uključujući pravilo* $(\mathbf{r}_1)$ $nzd(u \cdot v, u) \longrightarrow u$*. Neka je kontekst* C *jednak* $\{z = (x \cdot y)\}$ *i neka je dat term* $nzd(z, x)$*. Tada važi* $nzd(z, x) \longrightarrow_{C,R} nzd(x \cdot y, z)$ *jer je* $z \simeq_C (x \cdot y)$*. Primenjujući pravilo* $(\mathbf{r}_1)$ *na* $nzd(x \cdot y, z)$*, dobijamo* x*. Dakle,* $nzd(z, x) \longrightarrow_{C,R} nzd(x \cdot y, z) \longrightarrow_{C,R} x$*.*

Saglasnost kontekstualnog prezapisivanja se može formalno dokazati. Moguće je nametnuti ograničenja na pravila prezapisivanja (u terminima uređenja svođenja) takva da je kontekstualno prezapisivanje zaustavljajuće. Potpunost kontekstualnog prezapisivanja (u njegovom osnovnom obliku) ne može biti garantovana i to ilustruje sledeći primer:

Primer B.2 *Neka je* $\mathcal{L}$ *skup koji sadrži sledeća dva primera:*

$$q \Rightarrow \neg p(a, x)$$

$$\neg q \Rightarrow \neg p(x, b)$$

Literal $\neg p(a, b)$ *je logička posledica skupa* $\mathcal{L}$*, ali nezadovoljivost literala* $p(a, b)$ *ne može biti dokazana korišćenjem niti prvog niti drugo pravila.*

Detaljan opis kontekstualnog prezapisivanja i njegovih različitih varijanti dat je u radu [124]).

B.6 Term rewriting, ordering and termination: Summary

Term rewriting systems have their origin in the seminal work of Knuth and Bendix [70]. That work was primarily intended for solving one class of the word problems, but later it found many other applications.

A *rewrite rule* is an oriented equation of the form $l \to r$. The rule $l \to r$ rewrites a term t to another term s if there is a subterm t' of t and a substitution ϕ such that $l\phi$ is equal to t' and s is t with the subterm t' replaced by $r\phi$ (and we denote it $t \Longrightarrow t'$. A *rewrite system* $\mathcal{R}$ is a finite set of rewrite rules. If a term can be rewritten by a term rewriting system $\mathcal{R}$, we say that t is in normal form. A term rewriting system $\mathcal{R}$ is terminating if there does not exist an infinite sequence of terms t_1, t_2, t_3, ... such that $t_1 \Longrightarrow_R t_2 \Longrightarrow_R t_3 \Longrightarrow_R \ldots$. A term rewriting system is confluent if $t_1 \Longrightarrow^* t_2$ and $t_1 \Longrightarrow^* t_3$ imply that there is a term t_4 such that $t_2 \Longrightarrow^* t_4$ and $t_3 \Longrightarrow^* t_4$. A term rewriting system is canonical if it is confluent and terminating. If a set of equality E can be transformed into a confluent and terminating term rewriting system, then there exists a decision procedure for determining whether some equality is implied by a given set E. Every set of ground equalities can be transformed into a canonical rewriting system. A partial ordering $\geq$ is monotonic if for every two terms s and t it holds: $s > t$ implies $f(t_1, \ldots, s, \ldots, t_n) > f(t_1, \ldots, t, \ldots, t_n)$, where $>$ is strict part of the ordering $\geq$. A partial ordering $\geq$ is closed under substitutions if $s > t$ implies $s\varphi > t\varphi$ (where φ is an arbitrary substitution). A partial, well-founded ordering $\geq$ is reduction ordering if it is monotonic ordering and closed under substitutions. A rewrite system $\mathcal{R}$ is terminating if and only if there is a reduction ordering $\prec$ such that it holds that $r \prec l$ for each rule $l \to r$. When a precedence relation on function and predicate symbols is given, Dershowitz' recursive path ordering [37] extends that relation to a reduction ordering $\prec$ on terms. A *conditional rewrite rule* is a rule of the form $l \to r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$, where the conditions $p_1, p_2, \ldots, p_k$ are literals. The rule $l \to r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$ rewrites a term t to another term s if there is a subterm t' of t and a substitution ϕ such that $l\phi$ is equal to t', s is t with the subterm t' replaced by $r\phi$ and each of the conditions $p_i\phi$ reduces to $\top$ (also by rewriting using rules). A system $\mathcal{R}$ of conditional rewrite rules is terminating if there is a reduction ordering $\prec$ such that for each rule $l \to r$ **if** $p_1 \wedge p_2 \wedge \ldots \wedge p_k$, it holds $r \prec l$ and $p_i \prec l$ $(1 \leq i \leq k)$.

Detailed account on term rewriting, ordering and termination can be found in [96, 40, 61].

Dodatak C

Jednakosni sistemi

I u računarstvu i u matematici uopšte često se razmatra pitanje da li neka jednakost sledi iz datog skupa jednakosti. Na primer, ako su date jednakosti $x + y = y + x$, $(x + y) + z = x + (y + z)$ i $-(-(x + y) + -(x + -y))$, može se razmatrati da li je jednakost $-(-x + y) + -(-x + -y))$ logička posledica datih jednakosti. Ovakvi sistemi imaju veliki značaj u raznim granama računarstva, a pre svega u automatskom dokazivanju teorema i u funkcionalnom programiranju (funkcionalni program je, suštinski, skup jednakosti i izvršavanje takvog programa je vrsta jednakosnog rezonovanja). Jednakosno rezonovanje od vitalnog značaja u mnogim aplikacijama uključujući prevodilačke optimizatore, funkcionalne jezike, rezonovanje o bazama podataka i, najvažnije, rezonovanje o različitim aspektima softvera i hardvera — o kolima, programima i specifikacijama. Algoritmi za izračunavanje kongruentnog zatvorenja nad skupom baznih jednakosti imaju veliki značaj u jednakostnom rezonovanju. Od sredine sedamdesetih do sredine osamdesetih opisano je više algoritama za kongruentno zatvorenje [87, 104, 8].

Algoritmi za kongruentno zatvorenje veoma često se koriste u dokazivačima teorma kao vezivno tkivo za raznorodne procedure odlučivanja.

Neka su signatura i skup termova nad signaturom definisani kao u poglavlju B.1.1. Koristimo slova $f, g, h, \ldots$ za *funkcijske simbole*, slova $a, b, c, \ldots$ za *individualne konstante*, slova $x, y, z, \ldots$ za *individualne promenljive*, slova $r, s, t, \ldots$ za termove i simbol $=$ za simbol jednakosti. *Jednakost* je izraz oblika $s = t$ gde su s i t termovi. *Jednakosni sistem* (eng. equational system) je skup jednakosti.

C.1 Semantika jednakosnih sistema

Semantika jednakosnih sistema može biti definisana analogno semantici za logiku prvog reda. *Strukturu* M čine neprazan domen D i dodela "značenja" f^M svakom funkcijskom simbolu f (uključujući individualne konstante). Funkcijskom simbolu f dodeljuje se značenje $f^M : D^n \to D$, tj. funkcija koja preslikiva n-torke elemenata D u D. Struktura M uključuje i dodele značenja

promenljivama: promenljivoj x dodeljuje se značenje $v(x)$ koje je element skupa D. Značenje t^M terma t definiše se na sledeći način: ako je t promenljiva x, onda t^M je $v(x)$; ako je t oblika $f(t_1, t_2, \ldots, t_n)$ onda t^M je $f^M(t_1^M, t_2^M, \ldots, t_n^M)$. Ako je jednakost $s = t$ tačna u strukturi M (tj. ako su s^M i t^M isti elementi skupa D), onda to zapisujemo $M \models s = t$. Kažemo da struktura M zadovoljava skup jednakosti E ako M zadovoljava sve jednakosti iz skupa E; to zapisujemo $M \models E$ i kažemo da je strukura M *model* skupa jednakosti E. Ako je svaka od jednakosti u nekom sistemu E tačna u svakom modelu, onda to zapisujemo $\models E$. Ako su E_1 i E_2 dva jednakosna sistema i ako je svaki model sistema E_1 takođe model i sistema E_2, onda to zapisujemo $E_1 \models E_2$ i kažemo da je E_2 logička posledica E_1.

Često, i u različitim kontekstima, postavlja se pitanje da li za neke sisteme E_1 i E_2 važi $E_1 \models E_2$. Obično se E_2 sastoji samo od jedne jednakosti. Tada se razmatra problem da li je ta jednakost logička posledica sistema E_1. Intuitivno, treba ispitati da li ta jednakost važi ako važe sve jednakosti iz E_1. S obzirom na to da je najčešće veoma teško proveriti da li je neka jednakost tačna u svakom modelu, veliki značaj ima pojam dokazivosti. U cilju rešavanja ovog problema razvijeni su mnogi sistemi dokazivanja i mnogi od njih su prilagođeni računarskim primenama. O vezi semantički i sintaksno uvedenog pojma jednakosti govori Birkohoffova teorema.

Ponekad su od interesa i jednakosti sa egzistencijalnim kvantifikatorima. Semantika egzistencijalnog kvantifikatora definiše se na sledeći način: važi $M \models (\exists x A)$ ako postoji struktura M' koja je jednaka strukturi M, sa izuzetkom značenja dodeljenog promenljivoj x i takva da važi $M' \models A$.

C.2 Sintaksa jednakosnih sistema

U prethodnom poglavlju uvedena je semantika jednakosnih sistema, a u ovom poglavlju uvodimo pojam sintaksne jednakosti. Sintaksna jednakost se obično uvodi jednakosnim sistemima izvođenja i izvodivost (dokazivost) se označava sa $\vdash$. Ako je E jednakosni sistem i ako je $t_1 = t_2$ dokazivo u teoriji određenoj tim sistemom, onda to zapisujemo $E \vdash t_1 = t_2$.

Definicija C.1 (E-jednakost) *Neka je (Σ, X) signatura i $E \subseteq Ter(\Sigma, X) \times (\Sigma, X)$ skup jednakosti. Sledećih šest pravila definiše relaciju izvodivosti ili dokazivosti u oznaci $\vdash$:*

1. *ako je $(s, t) \in E$, onda važi $E \vdash s = t$.*
2. *$E \vdash s = s$.*
3. $$\frac{E \vdash s = t \quad \phi \text{ supstitucija}}{E \vdash s\phi = t\phi}$$
4. $$\frac{E \vdash s_1 = t_1,\ E \vdash s_2 = t_2,\ \ldots, E \vdash s_n = t_n}{E \vdash f(s_1, s_2, \ldots, s_n) = f(t_1, t_2, \ldots, t_n)}$$

5.
$$\frac{E \vdash t_1 = t_2 \; E \vdash t_2 = t_3}{E \vdash t_1 = t_3}$$

6.
$$\frac{E \vdash t_1 = t_2}{E \vdash t_2 = t_1}$$

Ako važi $E \vdash t_1 = t_2$, onda za termove t_1 i t_2 kažemo da su jednaki (mod E) *i zapisujemo i u obliku $t_1 =_E t_2$.*

Teorema Birkohoffa daje vezu između semantički i sintaksno uvedenog pojma jednakosti (naglasimo da ona ne važi u slučaju kada su dozvoljene i prazne sorte).

Teorema C.1 (Birkhoff, 1935) *Neka je (Σ, X) signatura (bez praznih sorti) i neka je $E \subseteq Ter(\Sigma, X) \times Ter(\Sigma, X)$ skup jednakosti. Tada važi:*

$$E \vdash t_1 = t_2 \;\; \textit{ako i samo ako} \;\; E \models t_1 = t_2$$

Teorema Birkhoffa može se formulisati i na sledeći način: važi $E \models s = t$ ako i samo ako postoji konačan niz termova $u_1, u_2, \ldots, u_n$ takav da je term u_1 identičan termu s, term u_n identičan termu t, a za svako i, term u_{i+1} je dobijen iz terma u_i zamenjivanjem podterma v terma u_i termom u gde je jednakost $v = u$ ili jednakost $u = v$ element sistema E. Ovo tvrđenje daje i metod za uspitivanje semantičke jednakosti na osnovu datih pravila izvođenja. Međutim, taj sistem je, očigledno, neefikasan. Poželjno je pravila izvođenja koristiti na neki restrikovan način, ali tako da ona i dalje imaju svojstvo potpunosti. To je upravo i bila motivacija za pojavu sistema za prezapisivanje.

Originalna teorema Birkhoffa odnosi se samo na jednakosne teorije, ali ovde ćemo je formulisati i u terminima sistema za prezapisivanje termova. Neka je E skup jednakosti, neka su sve jednakosti orijentisane u pravila prezapisivanja (dajući sistem R) i neka je relacija $\Longleftrightarrow^*_R$ definisana kao u delu o sistemima prezapisivanja; onda važi:

$$E \models s = t \quad \text{ako i samo ako} \quad s \Longleftrightarrow^*_R t \; .$$

Za dokaz videti [56] ili [96].

Na osnovu teoreme Birkhoffa, ukoliko je sistem za prezapisivanje R zasnovan na skupu jednakosti E konfluentan, za ispitivanje da li važi $E \models t = t'$, dovoljno je proveriti da li su u sistemu R kanonske forme termova t i t' jednake.

C.3 Kongruentno zatvorenje

Ako je E skup jednakosti, problem $E \models s = t$ može biti rešavan tehnikama prezapisivanja i Knuth/Bendixovom procedurom upotpunjavanja. Najčešće je, međutim, od tog pristupa efikasniji metod *kongruentnog zatvorenja* [87, 104, 8]. Ovaj metod zasnovan je na ideji primenjivanja aksioma jednakosti, ali suženih na one termove koji se pojavljuju u E, uključujući i njihove podtermove. Metod je, između ostalog, zasnovan na sledećoj teoremi:

Teorema C.2 *Ako postoji dokaz jednakosti $a = b$ iz jednakosne teorije E korišćenjem refleksivnosti, simetričnosti, tranzitivnosti i supstitucije, onda postoji i dokaz za $a = b$ iz E u kojem su sve primene tranzitivnosti svedene na slučaj kada je term sečenja (eng.* cut term*) leva ili desna strana neke jednakosti iz E.*

Neka je, na primer, dat skup jednakosti E jednak $\{w = g(f(a)), f(a) = a\}$ i neka je potrebno dokazati da važi $w = g(a)$. Metod kongruentnog zatvorenja konstruiše graf čiji čvorovi odgovaraju termovima i podtermovima termova iz skupa $\{w, g(f(a)), g(a)\}$. Relacija kongruentnog zatvorenja povezuje (smeštanjem u iste klase ekvivalencije) one čvorove koji odgovaraju datim jednakostima tako da su termovi w i $g(f(a))$ identifikovani, kao i termovi $f(a)$ i a. Dalje, bilo koja dva čvora sa identičnom oznakom čiji sledbenici su identični su takođe identični. Identifikovanje termova a i $f(a)$, dakle, indukuje identifikovanje termova $g(a)$ i $g(f(a))$. Kao rezultat, termovi w i $g(a)$ su u istoj klasi ekvivalencije.

Neka je $G = (V, D)$ označeni usmereni graf za koji su $\lambda(v)$ i $\delta(v)$ redom oznaka i izlazni stepen čvora v (iz skupa V). Neka je $v[i]$ čvor u takav da je (v, u) i-ta grana sa početnim čvorom v. Ako je α neki skup čvorova, neka je $use(\alpha)$ skup $\{v \in V \mid (\exists i : v[i] \in \alpha)\}$. Za datu binarnu relaciju R nad V, ekvivalentno zatvorenje relacije R (označeno sa R^*) je refleksivno, simetrično i tranzitivno zatvorenje relacije R. Kongruentno zatvorenje $\hat{R}$ binarne relacije R je najmanje proširenje relacije R^* takvo da za svaka dva čvora u, v ako važi $\lambda(u) = \lambda(v)$, $\delta(u) = \delta(v)$ i $u[i]\hat{R}v[i]$ za $1 \leq i \leq \delta(u)$, onda važi i $u\hat{R}v$.

Neka je S skup termova koji je zatvoren za podtermove. Skup S možemo reprezentovati grafom $G = (V, D)$ takvim da je svaki term x_i (svaki simbol varijable) list (tj. nema sledbenika) i svaki term $f(a_1, a_2, \ldots, a_n)$ je reprezentovan čvorom v takvim da je $\lambda(v) = f$, $\delta(v) = n$ i čvorovi $v[1]$, $v[2]$, …, $v[n]$ reprezentuju redom termove $a_1, a_2, \ldots, a_n$. Ako je G graf koji reprezentuje S i a term iz skupa S, neka je $\nu(a)$ čvor iz G koji reprezenutuje a. Ako je E bazna jednakosna teorija oblika $a_1 = b_1$, $a_2 = b_2$, …, $a_n = b_n$ takva da su svi termovi a_i i b_i u skupu S, onda neka je R_E binarna relacija nad S koja sadrži parove $(\nu(a_i), \nu(b_i))$ i zatvorena za refleksivnost, simetričnost i tranzitivnost.

Teorema C.3 *Neka je E bazna jednakosna teorija, S skup termova zatvoren za podtermove koji sadrži sve termove iz E i neka graf $G = (V, D)$ reprezentuje skup S. Tada za dva terma a i b iz S važi:*

$$E \models a = b \text{ ako i samo ako } E \vdash a = b \text{ ako i samo ako } \nu(a)\hat{R}_T\nu(b) \ .$$

C.4 Nelson/Oppenov algoritam za kongruentno zatvorenje

U nastavku je dat pseudokôd za osnovni algoritam za kongruentno zatvorenje Nelsona i Oppena [87, 32].

Procedura *Makuse* je jednostavna i ovde nije definisana: ona inicijalizuje strukturu koja odgovara funkciji *use* tako da ona pokriva sve termove iz E

ili, alternativno, sve termove iz unapred zadatog skupa termova koji mora da uključuje sve termove iz E (za ovaj algoritam karakteristično je, dakle, da skup termova koji se obrađuju mora da bude poznat unapred). U okviru procedure $Makeuse$, za svaki dati term t (uključujući i sve podtermove) određuje se lista $use(t)$ i pamti uređeni par $(t, use(t))$.

Glavna NO petlja obrađuje listu jednakosti E primenjivanjem procedure $Merge$ koja koristi primitive za uniju za održavanje klasa ekvivalencija: *union* spaja dve klase ekvivalencije, a $find$ vraća kanonskog predstavnika određene klase ekvivalencije. Ove primitive obično se efikasno implementiraju na osnovu opisa iz [115]. Generalno, $find$ je moguće implementirati na bazi strukture parova $(t, find(t))$ koja za svaki obrađeni term sadrži i kanonskog predstavnika klase ekvivalencije kojoj pripada. U tom pristupu, nema eksplicitnog memorisanja klasa ekvivalencija, pronalaženje kanonskog predstavnika je jednostavno i efikasno, ali je nešto složenije izvršavanje operacije *union*. Alternativno, moguće je klase ekvivalencije memorisati kao uređene parove (c, C) gde je C tekuća lista svih termova u toj klasi, a c njen kanonski predstavnik; u tom pristupu ne postoji struktura koja neposredno odgovara funkciji $find$, već se ona izračunava na osnovu činjenice da za svaki element a iz C, važi $find(a) = c$. Po konvenciji, pri određivanju unije dve klase ekvivalencije C_1 i C_2, za kanonskog predstavnika nove klase uzima se kanonski predstavnik klase C_2. Dakle, u pomenutoj reprezentaciji, operacija *union* nad elementima a i b redom iz klasa reprezentovanih sa (c_1, C_1) i (c_2, C_2) zamenila bi te dve klase novom klasom reprezentovanom sa $(c_2, C_1 \cup C_2)$. U oba slučaja, ukoliko term t nije u zadatom, polaznom skupu termova, funkcija $find$ vraća sam taj term.

Ključna procedura algoritma je $Merge$ koja spaja dve klase ekvivalencije i prosleđuje spajanje svim prethodničkim čvorovima ove dve klase koji takođe postaju ekvivalentni.

```
Merge(u,v) =
   LET P_u = U { use(u') | find(u') = find(u) };
   LET P_v = U { use(v') | find(v') = find(v) };
   DO union(u,v);
   FOR x IN P_u DO
      FOR y IN P_v DO
         IF find(x) <> find(y) AND Congruent(x,y)
            THEN Merge(x,y)
         ENDIF
```

```
Congruent(u,v) =
( delta(u)=delta(v)
  AND (FORALL i: 1<= i <= delta(u):
       find(u[i]) = find(v[i])))
```

```
Nelson_Oppen(E) =
   Makeuse(E);
   CASES E OF
       nil : RETURN
[a=b : E'] : Nelson_Oppen(E');
             IF find(a)= find(b)
                THEN RETURN
                ELSE merge(a,b)
             ENDIF;
   ENDCASES
```

Lako se dokazuje da se data procedura zaustavlja (jer se broj klasa ekvivalencija smanjuje svakim pozivanjem procedure $Merge$). Sledeća teorema [87, 32] tvrdi da procedura NO generiše kongruentno zatvorenu kolekciju klasa ekvivalencija termova. Neka $find_E$ predstavlja operaciju $find$ koja proizilazi iz algoritma $NO(E)$.

Teorema C.4 (Korektnost procedure NO) *Neka je E jednakosna teorija i S skup termova koji je zatvoren za podskupove i sadrži sve termove iz E. Tada za svaka dva terma a i b iz S važi*

$$find_E(a) = find_E(b) \;\; \textit{ako i samo ako} \;\; \nu(a)\hat{R}_T\nu(b) \; .$$

C.5 Shostakov algoritam za kongruentno zatvorenje

Shostakova verzija algoritma za kongruentno zatvorenje [104] optimizuje osnovnu, Nelson/Oppenovu verziju u tri vida (od kojih je treći najznačajniji [32]):

- Izračunavanje skupova termova $\mathtt{P_u}$ i $\mathtt{P_u}$ u proceduri $Merge$ je optimizovano tako da je uvek zadovoljeno $use(u) \subseteq use(find(u))$.
- Izračunavanje vrednosti $Congruent$ je optimizovano održavanjem strukture podataka $sig(u)$ za svaki term u koja zadovoljava $sig(f(u_1, u_2, \ldots, u_n)) = f(find(u_1), find(u_2), \ldots, find(u_n))$.
- Eliminisanje unapred zadatog skupa termova; to znači da skup termova ne mora da bude poznat unapred i da algoritam uvodi nove termove "u hodu" (i ažurira strukturu use). Zbog toga je uvedena dodatna funkcija $canon$ koja vraća kanonsku formu za do tada nepoznate termove na bazi kanonskih formi za podtermove dobijene strukturom $find$. Kako je u Shostakovom algoritmu skup termova otvoren, ne važi uvek $find_{Sh}(u) = find_{NO}(u)$ kada dva algoritma rade paralelno.

U nastavku je dat pseudokôd Shostakovog algoritma za kongruentno zatvorenje. Procedura $Sh(E)$ obrađuje svaku jednakost iz datog skupa jednakosti E korišćenjem procedure $Merge$ i gradi strukturu kongruentnog zatvorenja na bazi strukura za $find$, use i sig. Struktura use ne mora da inicijalno oslikava sve

termove datih jednakosti, veće nove termove i njihove neposredne veze ažurira tokom uvođenja novih termova; struktura $find$ je inicijalizovana tako da je $find(t) = t$ za sve termove, a struktura sig se tokom algoritma ažurira tako da uvek važi $sig(f(u_1, \ldots, u_n)) = f(find(u_1), \ldots, find(u_n))$. U odnosu na Nelson/Oppenovu varijantu, uvedene su i funkcije *canon* i *canonsig*. Procedura primenjuje *canon* na obe strane jednakosti kako bi bile dobijene njihove normalne forme zasnovane na do tada poznatim jednakostima. U slučaju terma t koji je već bio obrađen, *canonsig* osigurava da je njegova normalna forma upravo $find(t)$. To je zbog toga što već ranije obrađen term t koji je argument za *canonsig* jeste ili varijabla ili konstanta (u kom slučaju se vraća $find(t)$ čak i ako t nije ranije obrađen; kao što je rečeno u prethodnom delu, funkcija $find$ za term koji nije poznat vraća sâm taj term) ili je oblika $f(t_1, \ldots, t_n)$ i jednak $sig(u)$ (jer su svi t_i normalizovani) za neko u iz $use(t_1)$ (umesto t_1 može biti iskorišćen i bilo koji term t_i, $1 \leq i \leq n$). Nakon toga, poziva se $Merge$ na rezultujuće normalne forme kako bi spojio klase ekvivalencija za dva terma i onda spojio klase ekvivalencije za kongruentne prethodničke čvorove (pri čemu se struktura *sig* koristi za brzi test kongruentnosti).

```
Shostak(E) =
   CASES E oF
       nil : RETURN
[a=b : E'] : Merge ( canon(a), canon(b));
             Shostak(E')
   ENDCASES


Merge(a,b) =
   UNLESS a=b DO
      union(u,v);
      FOR u IN use(a) DO
        replace a by b in the argument list of sig(u);
        FOR v IN use(b) WHEN sig(v)=sig(u) DO
           Merge(find(u),find(v));
        use(b):= use(b) U {u}


canon(t) =
   CASES t OF
     f(t_1,...,t_n): canonsig(f(canon(t_1,...,canon(t_n))),
     ELSE: canonsig(t)
   ENDCASES


canonsig(t) =
   CASES t OF
      f(t_1,...,t_n):
```

```
        IF (FORSOME u IN use(t_1): t = sig(u))
           THEN RETURN find(u)
           ELSE
              FOR i FROM 1 to n DO use(t_i):=use(t_i) U {t};
              sig(t) := t;
              use(t) := {};
              RETURN t
        ENDIF
     ELSE: find(t);
  ENDCASES
```

Za ilustraciju razlike između Nelson/Oppenovog i Shostakovog algoritma za kongruentno zatvorenje, razmotrimo skup termova $\{x, y, z, f(x), f(y)\}$ i skup jednakosti $\{x = y, z = f(x)\}$. U obe varijante, nakon što je jednakost $z = f(x)$ obrađena, važi $find_{NO}(z) = find_{Sh}(z) = f(x)$. Nakon što je $x = y$ obrađeno, u Nelson/Oppenovoj verziji važi $find_{NO}(x) = y$ i $find_{NO}(f(x)) = f(y)$, dok je u Shostakovoj verziji $find_{Sh}(f(x)) = f(x)$ jer $f(y)$ nije eksplicitno prisutno ni u jednost od jednakosti obrađenih do tog trenutka. U Shostakovoj varijanti, operacija *canon* može da radi sa do tada nekorišćenim termovima, pa je $canon(f(x)) = f(y)$. Ovo ne znači da uvek važi $canon(t) = find_{NO}(t)$: kada je druga jednakost obrađena, važi $canon(z) = find_{Sh}(z) = f(x)$, dok je $find_{NO}(z) = f(y)$. Na drugoj strani, $canon(f(z)) = f(f(x))$, dok je $find_{NO}(f(z)) = f(z)$ jer $f(z)$ nije u datom skupu termova. Ipak, za sve termove iz term univerzuma S, *canon* i $find_{NO}$ održavaju iste klase ekvivalencije, pa je $find_{NO}(a) = find_{NO}(b)$ ako i samo ako je $canon(a) = canon(b)$. Više o uporednoj analizi Nelson/Oppenovog i Shostakovog algoritma videti u [32].

Dokaz korektnosti Shostakovog algoritma proističe iz korektnosti Nelson/Oppenovog algoritma i sledeće invarijante ($canon*^T$ je isto što i *canon* u Shostakovom algoritmu za zadati skup jednakosti T, ali bez bočnih efekata izmene struktura *sig* i *use*; $find^T_{NO}(t)$ je rezultat procedure $find$ iz Nelson/Oppenovog algoritma za skup jednakosti T) [32]:

Teorema C.5 *Za svaka dva terma a i b iz term univerzuma algoritma za Nelson/Oppenov algoritam važi:*

$$canon *^T (a) = canon *^T (b) \;\; ako\ i\ samo\ ako \;\; find^T_{NO}(a) = find^T_{NO}(b)\ .$$

Efekat primene Shostakovog algoritma ilustrujmo sledećim primerom [67]:

Primer C.1 *Razmotrimo skup jednakosti $\{f(a) = g(f(a))\}$. Kanonska forma terma $f(a)$ određuje se na osnovu praznog skupa jednakosti. Najpre se određuje kanonska forma terma a, što je upravo a. Zatim se određuje kanonska forma terma $f(a)$ – najpre se ispituje da li se $f(a)$ već pojavljuje među termovima u kojima je a prisutno kao argument (koristeći strukturu use). Odgovor je "ne", pa se use(a) ažurira tako da uključuje $find(a)$; $sig(f(a))$ se takođe ažurira tako da bude jednako $f(a)$ i $f(a)$ je kanonska forma za $f(a)$. U izračunavanju kanonske forme terma $g(f(a))$, najpre se računa kanonska forma terma a što je a; zatim se*

određuje kanonska forma terma $f(a)$ *što je* $find(f(a)) = f(a)$*, jer* $use(a)$ *sada uključuje* $f(a)$ *i* $f(a) = sig(f(a))$*. Konačno, dok se određuje kanonska forma za* $g(f(a))$*, skup* $use(f(a))$ *se ažurira tako da uključuje* $\{g(f(a))\}$*;* $sig(g(f(a))$ *dobija vrednost* $g(f(a))$ *i, konačno, kanonska forma za* $g(f(a))$ *je* $g(f(a))$*.*

Sada, kada su strukture use i sig inicijalizovane, zadata jednakost se obrađuje najpre spajanjem klasa kongruencije što vodi najpre do primenjivanja operacije union na $f(a)$ *i* $g(f(a))$ *i zatim zamenjivanja terma* $f(a)$ *termom* $g(f(a))$ *u sig u svakom podtermu koji se pojavljuije u* $use(f(a))$*. (Kako je* $use(f(a)) = \{g(f(a))\}$*,* $f(a)$ *je element* $sig(g(f(a))) = g(f(a))$ *se zamenjuje termom* $g(f(a))$ *dajući* $g(g(f(a)))$*. Skup* $use(g(f(a))$ *dobija vrednost* $\{g(f(a))\}$*. Dakle, nakon obrađivanja svih jednakosti važi:*

$use(a) = \{f(a)\},\ sig(a) = a,\ find(a) = a,$
$use(f(a)) = \{g(f(a))\},\ sig(f(a)) = f(a),\ find(f(a)) = g(f(a)),$
$use(g(f(a))) = \{g(f(a))\},\ sig(g(f(a)) = g(g(f(a))),\ find(g(f(a))) = g(f(a)).$

Sa navedenim određenim strukturama, mogu se odrediti kanonske forme:

$canon(a) = a,$
$canon(f(a)) = g(f(a)),$
$canon(g(a)) = g(a),$
$canon(g(f(a)) = g(f(a)),$
$canon(f(f(a))) = f(g(f(a))),$
$canon(f(g(f(g(a))))) = f(g(f(a))),$
i tako dalje.

C.6 Kongruentno zatvorenje kao upotpunjavanje

Još od sedamdesetih godina zna se da kongruentno zatvorenje nad skupom baznih jednakosti može da se odredi korišćenjem procedure upotpunjavanja. Osnovna ideja je u tome da se jednakosti orijentišu u zaustavljajuća pravila prezapisivanja i da se zatim dobijeni sistem za prezapisivanje svede (tj. normalizuje). Za razliku od opšteg slučaja, bazne jednakosti se uvek mogu orijentisati u zaustavljajući sistem za prezapisivanje.

Postoji značajan broj radova o generisanju kompletnog sistema prezapisivanja (koji čine samo bazna pravila) od skupa baznih jednakosti u polinomijalnom vremenu [108]. Ovi algoritmi najpre koriste algoritam za kongruentno zatvorenje nad grafom podtermova (indukovanim zadatim skupom jednakosti) za određivanje klasa kongruencija. Jedinstveni predstavnik iz svake klase se zatim odabira za generisanje kompletnog, svedenog skupa pravila prezapisivanja. Ako je dozvoljeno signaturu proširiti novim konstantama, generisanje sistema za prezapisivanje je relativno lako [108]. Tek nedavno je razvijen algoritam polinomijalne složenosti za generisanje kompletnog sistema za prezapisivanje direktno iz zadatih baznih jednakosti bez eksplicitnog konstruisanja kongruentnog zatvorenja korišćenjem grafovski orijentisanih algoritama za kongruentno zatvorenje [95]. Shostakov algoritam za kongruentno zatvorenje moguće je interpretirati kao upotpunjavanje [67]. Slično važi i za apstrahovani algoritam za kongruentno zatvorenje [8].

C.7 Varijante i složenost

Postoji više algoritama za računanje kongruentnog zatvorenja i njihovih varijanti, počev od algoritama opisanih u prethodnom delu [87, 104, 32]. Izračunavanje kongruentnog zatvorenja moguće je u vremenu $O(n \log n)$ (gde je n broj jednakosti i ima najviše n različitih termova), pri čemu se koristi i prostor $O(n \log n)$ [11]. Na bazi Shostakovog algoritma moguće je generisati (bez konstruisanja odgovarajućeg grafa) odgovarajući konfluentan bazni sistem za prezapisivanje u vremenu $O(n^2)$ [67]. To vreme moguće je smanjiti na $O(n \log n)$ ako se umesto linearnog prostora koristi prostor $O(n \log n)$ [11]. Za dodatne varijante algoritma za kongruentno zatvorenje videti [11, 9].

C.8 Equational Systems: Summary

Equality reasoning is often vital component of a theorem prover. Equality reasoning can be performed via equality axioms, but much more efficient by some other techniques. Reasoning about ground equalities can be handled by Knuth/Bendix completion procedure (which always give canonical term rewriting system for a set of ground equalities) or, more efficiently, by congruence closure algorithms [87, 104]. Ground completion and congruence closure are substantially the same thing and congruence closure algorithms can be interpreted as completion [67, 8].

Dodatak D

Prezburgerova aritmetika

Rezonovanje o problemima specifikacije i opisa generičkih komponenti koje se javljaju u dizajnu softvera i hardvera, najčešće zahteva rezonovanje o prirodnim i celim brojevima. To je posebno izraženo u kontekstu linearnih struktura podataka kao što su nizovi. Rezonovanje o prirodnim i celim brojevima je često esencijalno i u problemima koji uključuju "mere" definisane za razne strukture podataka (npr. veličina, dužina, dubina itd.) Kako je cela aritmetika nodlučiva [82], veoma značajno mesto pripada njenim odlučivim fragmentima kao što su Prezburgerova aritmetika ili strogo multiplikativna aritmetika [97, 107, 84]. Zbog njenog značaja u verifikacijskim problemima, ovom deo teksta bavi se Prezburgerovom aritmetikom, kako u funkciji ispitivanja svojstava ove teorije i relevantnih algoritama, tako i u funkciji ilustracije rezultata koji mogu biti primenjeni i na druge teorije.

Neka je $\langle \mathbf{N}, + \rangle$ struktura prirodnih brojeva sa sabiranjem. Teoriju $Th(\langle \mathbf{N}, + \rangle)$ zovemo Prezburgerovom aritmetikom ili teorijom sabiranja prirodnih brojeva i označavamo[1] sa PNA. Jezik ove teorije sadrži samo jednu sortu (koju ćemo označavati sa $\mathbf{N}$), nema predikatskih simbola (osim simbola $=_{\mathbf{N}}$), ima samo jedan funkcijski simbol $(+)$ i njemu odgovara sorta $(\mathbf{N}, \mathbf{N}, \mathbf{N})$. Teoriju $Th(\langle \mathbf{Z}, + \rangle)$ (teoriju sabiranja celih brojeva; ona ima jednu sortu koju označavamo sa $\mathbf{I}$) označavamo sa PIA, a teoriju $Th(\langle \mathbf{Q}, + \rangle)$ (teoriju sabiranja racionalnih brojeva[2]; ona ima jednu sortu koju označavamo sa $\mathbf{Q}$) sa PRA. Ove teorije mogu biti zasnovane i aksiomatski. Jedan od mogućih sistema aksioma za PNA je sistem S [44] koji čini sledećih pet aksioma i jedna beskonačna shema aksioma (u daljem tekstu nećemo označavati sortu u simbolu jednakosti kada je ona jasna iz konteksta):

[1] Često se ova teorija označava sa PAR. Mi ćemo, međutim, razmatrati "aritmetiku sabiranja" i na strukturama celih i racionalnih brojeva, pa oznakom PNA naglašavamo da mislimo na teoriju koja je vezana za strukturu prirodnih brojeva.

[2] Naglasimo da za teoriju sabiranja nad realnim brojevima ($Th(\langle \mathbf{R}, + \rangle)$) važi $Th(\langle \mathbf{R}, + \rangle) = Th(\langle \mathbf{Q}, + \rangle)$ (do na oznake sorti u formulama).

$S1$. $(\forall x)(\forall y)(x + y = y + x)$
$S2$. $(\forall x)(\forall y)(\forall z)(x + (y + z) = (x + y) + z)$
$S3$. $(\forall x)(\forall y)(\forall z)(x + y = x + z \to y = z)$
$S4$. $(\forall x)(\forall y)((x \leq y \vee y \leq x) \wedge (x \leq y \wedge y \leq x \to x = y))$
$S5$. $(\exists x)(x = 1)$
$S6_n$. $(\forall x)(\exists d)(\forall v)(x = nd + v \wedge \bigvee_{i=0}^{n-1} v = i)$

U formulaciji aksioma korišćene su sledeće definicije:

$$x \leq y \stackrel{def}{\Leftrightarrow} (\exists z)(x + z = y)$$

$$x < y \stackrel{def}{\Leftrightarrow} x \leq y \wedge x \neq y$$

$$x = 0 \stackrel{def}{\Leftrightarrow} x + x = x$$

$$x = 1 \stackrel{def}{\Leftrightarrow} x \neq 0 \wedge (\forall z)(\exists t)(z = 0 \vee x + t = z)$$

Koristićemo i kraće oznake tipa $x = n$ gde je n prirodan broj. Te skraćene zapise uvodimo induktivno. Ako je skraćenica $x = n$ već uvedena, neka je

$$x = n + 1 \stackrel{def}{\Leftrightarrow} (\exists u)(\exists v)(u = n \wedge v = 1 \wedge x = u + v)$$

Koristimo i sledeću skraćenu oznaku (n je prirodan broj):

$$nx \stackrel{def}{\Leftrightarrow} \underbrace{x + x + \ldots + x}_{n},$$

pa kažemo i da je u Prezburgerovoj aritmetici dozvoljeno množenje prirodnim brojem.

Može se pokazati da je teorija $Th(S)$ potpuna [44]. Kako su, pored toga, sve aksiome teorije $Th(S)$ tačne u strukturi $\langle \mathbf{N}, + \rangle$, sledi da su teorije $Th(S)$, $Th(\langle \mathbf{N}, + \rangle)$ i PNA identične. Teorija $Th(S)$ je potpuna i njen skup aksioma je rekurzivan, pa na osnovu teoreme A.1, sledi da je teorija $Th(S)$ odlučiva. Nama će od većeg značaja biti dokazi odlučivosti Prezburgerove aritmetike zasnovani na eliminaciji kvantifikatora koji indukuju odgovarajuće procedure odlučivanja. Neke od procedura odlučivanja za Prezburgerovu aritmetiku (pogodnih za implementaciju) zasnovani na eliminaciji kvantora su Cooperova procedura za PIA [29] i Hodesova za PRA [52] (koja je suštinski zasnovana na Furijeovom metodu za rešavanje linearnih nejednakosti nad racionalnim brojevima [74]). Pored procedura zasnovanih na eliminaciji kvantora, značajna je i grupa SUP-INF procedura za PRA koju je razvio Bledsoe [13] i kasnije usavršavao Shostak [105, 106], kao i procedura odlučivanja za PIA zasnovana na Furijeovom metodu koja se koristi u sistemu TECTON [68]. Vredno je pomenuti i pionirsku Davisovu implementaciju originalnog Presburgerovog algoritma [36]; taj program napisan 1954. godine zasnivao se na mnogo *ad hoc* rešenja i nije bitno uticao na razvoj automatske dedukcije. Ipak, smatra se da je to prvi program koji je automatski dokazivao teoreme i zbog toga mu pripada posebno mesto u oblasti.

Procedure za PRA su u automatskom rezonovanju često korišćene kako bi se izbegla znatno veća kompleksnost procedura odlučivanja za PIA. Lako se,

međutim, može videti da postoje formule koje su tačne u PRA, a nisu tačne u PIA i obratno. Na primer, formula $(\exists x)2x = 3$ je tačna u strukturi racionalnih brojeva, ali ne i u strukturi celih. Takođe, formula $(\forall x)(x \leq 1 \vee x \geq 2)$ je tačna u strukturi celih brojeva, ali nije tačna u strukturi realnih. Dakle, nije moguće koristiti proceduru za PRA kao proceduru odlučivanja (pa čak ni kao proceduru poluodlučivanja) za PIA niti obratno. Međutim, ako je univerzalno kvantifikovana formula tačna u PRA, onda je onda tačna i u PIA i u PNA (obratno ne važi), pa procedura odlučivanja za PRA može u tom smislu biti upotrebljiva i za PIA i PNA (kao procedura koja je nepotpuna čak i za univerzalno kvantifikovan fragment).

Ideja korišćenja procedura odlučivanja za PRA i za PIA je osnovna u SUP-INF metodu [13] i ona se može smatrati početkom tradicije korišćenja nepotpunih procedura za PIA. Ta tradicija nastavljena je i Shostakovim radovima [105, 106] u kojima je on unapredio SUP-INF metod tako da je bilo moguće razrešiti i neke formule koje nisu teoreme u PIA. Ipak, klasa PIA formula za koju je Shostakov SUP-INF metod procedura odlučivanja nije opisana sintaksno, već je ona na specifičan način semantički karakterisan fragment teorije PIA.

Boyer i Moore su takođe sledili tradiciju korišćenja nepotpunih procedura za PIA u svom dokazivaču NQTHM [17] iako nisu koristili SUP-INF metod, već Hodesovu proceduru. Ovaj izbor je relativno prirodan, jer je logika sistema NQTHM bez kvantifikatora, pa je saglasnost primene Hodesove procedure za PIA obezbeđena. Čudno je, međutim, da Boyer i Moore tvrde [17] da je efikasnost sâme procedure odlučivanja malo važna u kontekstu ugradnje u heuristički dokazivač teorema[3]: u tom slučaju zašto nije upotrebljena Cooperova procedura, čime bi bila obezbeđena kompletnost za PIA? U tom smislu, značajno je pitanje u kojoj meri neki heuristički dokazivač teorema može da iskoristi negativan odgovor dobijen od procedure odlučivanja (npr. u sistemu *Clam* [18] ta informacija može se upotrebiti u kontrolisanju generalizacije i u drugim heuristikama koje ne čuvaju ekvivalenciju).

D.1 Hodesova procedura odlučivanja za PRA

Procedura koju ćemo u daljem tekstu zvati Hodesova procedura zasnovana je suštinski na Furijeovom metodu za rešavanje linearnih nejednakosti nad racionalnim brojevima [74]. U matematičkoj literaturi ovaj metod je, ponekad uz određene varijacije, više puta iznova "otkrivan" od strane raznih autora. Autor prve verzije prilagođene automatskoj primeni i implementirane je Hodes [52]. U daljem tekstu dajemo Hodesov opis procedure odlučivanja za PRA[4]. U

[3] Heuristički dokazivač teorema sa integrisanom procedurom odlučivanja obično ima širi domen nego sâma ta procedura. To je slučaj i sa sistemom NQTHM.

[4] Sâm Hodes u svom tekstu ovu teoriju naziva EAR (the elementary theory of addition on reals), mada precizira da se zapravo radi o elementarnoj teoriji uređenih gustih Abelovih grupa bez krajnjih tačaka. Boyer i Moore [17] zovu univerzalno kvantifikovan fragment teorije PRA linearnom aritmetikom (pri čemu Hodesovu proceduru primenjuju na PNA). Ponegde se na teoriju PRA referiše i kao na Bledsoeovu realnu aritmetiku.

poglavlju D.5.3 Hodesova procedura biće opisana preciznije i to u terminima pravila prezapisivanja.

1. Neka je PRA rečenica čiju dokazivost treba ispitati u preneks normalnoj formi; ako rečenica nema kvantifikatora/promenljivih (tj. ako je ona bazna), njena dokazivost može trivijalno biti ispitana. Ako je njen unutrašnji kvantifikator univerzalni, onda ga možemo zameniti egzistencijalnim koristeći vezu $(\forall x)g \equiv \neg(\exists x)\neg g$.

2. Koristimo naredne korake da odredimo formulu $g(x_1, x_2, \ldots, x_n)$ bez kvantifikatora ekvivalentnu formuli $(\exists y)F(x_1, x_2, \ldots, x_n, y)$; nakon toga zamenjujemo $(\exists y)F(x_1, x_2, \ldots, x_n, y)$ formulom $g(x_1, x_2, \ldots, x_n)$ i idemo na korak 1.

3. Eliminišemo sva pojavljivanja ekvivalencije i implikacije u formuli $(\exists y)f$ koristeći identitete:

$$f_1 \leftrightarrow f_2 \equiv (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$$

$$f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2$$

4. Koristimo identitete

$$\neg(f_1 \wedge f_2) \equiv (\neg f_1 \vee \neg f_2)$$

$$\neg(f_1 \vee f_2) \equiv (\neg f_1 \wedge \neg f_2)$$

$$\neg\neg f_1 \equiv f_1$$

   za eliminisanje svih simbola $\neg$, osim onih koji direktno dominiraju atomičkim formulama.

5. Preostale negacije eliminišemo koristeći sledeće veze:

$$\neg(t_1 = t_2) \equiv (t_1 < t_2) \vee (t_2 < t_1)$$

$$\neg(t_1 < t_2) \equiv (t_1 = t_2) \vee (t_2 < t_1)$$

   Označavamo novodobijenu formulu sa $(\exists y)f$. Formula f sadrži samo veznike $\wedge$ i $\vee$

6. Koristimo distributivni zakov

$$f_1 \wedge (f_2 \vee f_3) \equiv (f_1 \wedge f_2) \vee (f_1 \wedge f_3)$$

   za transformisanje formule f u disjunktivnu normalnu formu

$$f \equiv f_1 \vee f_2 \vee \ldots \vee f_n$$

   gde je f_i $(i = 1, 2 \ldots, n)$ konjunkcija atomičkih formula.

7. Egzistencijalni kvantifikator "prolazi" kroz disjunkciju, pa važi:

$$(\exists y)f \equiv (\exists y)(f_1 \vee f_2 \vee \ldots \vee f_n \equiv (\exists y)f_1 \vee (\exists y)f_2 \vee \ldots \vee (\exists y)f_n$$

Ako u f_i nema atomičkih formula koje sadrže y onda je

$$(\exists y)f_i \equiv f_i \ ,$$

čime je eliminisan kvantifikator $\exists y$ iz $(\exists y)f_i$. U suprotnom, formula f_i može biti napisana u obliku $f_i' \wedge f_i''$ gde je f_i' konjunkcija svih atomičkih formula iz f_i koje sadrže y. Kako f_i'' ne sadrži y važi

$$(\exists y)(f_i' \wedge f_i'') \equiv f_i'' \wedge (\exists y)f_i'$$

8. Potrebno je eliminisati kvantifikatore $\exists y$ iz svih formula $(\exists y)f_i'$. Svaka od ovih formula je oblika

$$(\exists y)(t_1^{(i)} = s_1^{(i)} \wedge \ldots \wedge t_{j_i}^{(i)} = s_{j_i}^{(i)} \wedge q_1^{(i)} < r_1^{(i)} \wedge \ldots \wedge q_{k_i}^{(i)} < r_{k_i}^{(i)})$$

Za $1 \leq i \leq n$, $j_i \geq 0$ je broj atomičkih formula oblika $t_1 = t_2$ i k_i je broj atomičkih formula oblika $t_1 < t_2$ u formuli f_i' (pri čemu važi $j_i + k_i > 0$). Svaka od ovih atomičkih formula može biti transformisana tako da se y i sa leve i sa desne strane pojavljuje najviše po jednom.

Najpre razmatramo slučaj $j_i > 0$, tj. slučaj da postoji bar jedna jednakost. Ako je $j_i = 1$ i $k_i = 0$, onda $(\exists y)(t_1^{(i)} = s_1^{(i)})$ može biti zamenjeno logičkom konstantom $\top$. Inače, rešimo jednakost "po" y i zamenimo dobijenu vrednost za y u preostalih $j_i + k_i - 1$ atomičkih formula. Konjunkcija dobijenih atomičkih formula može da zameni formulu $(\exists y)f_i'$ (čime je eliminisan kvantifikator $\exists y$).

Razmotrimo slučaj $j_i = 0$. Formula $(\exists y)f_i'$ je oblika

$$(\exists y)(q_1^{(i)} < r_1^{(i)} \wedge \ldots \wedge q_{k_i}^{(i)} < r_{k_i}^{(i)})$$

Ako se y u svakoj nejednakosti nalazi sa desne strane; tada zamenjujemo $(\exists y)f_i'$ logičkom konstantom $\top$. Slično, ako se y u svakoj nejednakosti nalazi sa leve strane; zamenjujemo $(\exists y)f_i'$ konstantom $\top$. Inače, za svaki par m, p takav da se y nalazi sa raznih strana u nejednakostima $q_m^{(i)} < r_m^{(i)}$ i $q_p^{(i)} < r_p^{(i)}$, odgovarajući par nejednakosti zamenjujemo novom nejednakošću

$$C_p q_m^{(i)} + C_m q_p^{(i)} < C_p r_m^{(i)} + C_m r_p^{(i)}$$

pri čemu je C_m koeficijent uz y u m-toj, a C_p koeficijent uz y u p-toj nejednakosti (pa je koeficijent uz y sa obe strane nejednakosti jednak $C_p C_m$); tako dobijene nejednakosti vezane su konjunkcijama. Ako u formuli f_i' ima j nejednakosti sa promenljivom y na levoj k sa promenljivom y na desnoj strani, onda novodobijena formula ima ukupno jk nejednakosti (vezanih konjunkcijama) iz kojih se promenljiva y trivijalno može eliminisati.

Korake 1—8 treba ponavljati dok se ne dobije formula bez kvantifikatora, kada u koraku 1 ona može biti svedena na $\top$ ili $\bot$.

Interesantno je da se u literaturi ne komentariše jedno jednostavno, ali veoma značajno unapređenje originalnog Hodesovog algoritma za PRA. Kritična tačka u Hodesovom algoritmu je konverzija u disjunktivnu normalnu formu (što je eksponencijalne složenosti). Kada su dva kvantifikatora koja se uzastopno eliminišu egzistencijalnog tipa, onda tu konverziju nije neophodno uraditi oba puta. Slično važi i za univerzalne kvantifikatore, ali je dodatno potrebno opisati eliminaciju univerzalnog kvantifikatora (analogno eliminaciji egzistencijalnog). Modifikacija Hodesovog algoritma se onda svodi na opisivanje eliminacije oba tipa kvantifikatora (za razliku od osnovne verzije u kojoj se eliminacija univerzalnog kvantifikatora svodi na eliminaciju egzistencijalnog). U opisu eliminacije univerzalnog kvantifikatora suštinski će se razlikovati nekoliko koraka: umesto disjunktivne normalne forme, formula treba da bude transformisana u konjuktivnu normalnu formu (korišćenjem pravila kao što je $f_1 \vee (f_2 \wedge f_3) \equiv (f_1 \vee f_2) \wedge (f_1 \vee f_3)$); univerzalni kvantifikator "prolazi" kroz konjunkciju, pa za formulu f u konjuktivnoj normalnoj formi važi: $(\forall y)f \equiv (\forall y)(f_1 \wedge f_2 \wedge \ldots \wedge f_n) \equiv (\forall y)f_1 \wedge (\forall y)f_2 \wedge \ldots \wedge (\forall y)f_n$; ako je formula $(\forall y)f_i'$ oblika $(\forall y)(q_1^{(i)} < r_1^{(i)} \vee \ldots \vee q_{k_i}^{(i)} < r_{k_i}^{(i)})$, tada za svaki par m, p takav da se y nalazi sa raznih strana u nejednakostima $q_m^{(i)} < r_m^{(i)}$ i $q_p^{(i)} < r_p^{(i)}$, odgovarajući par nejednakosti zamenjujemo novom nejednakošću $C_p q_m^{(i)} + C_m q_p^{(i)} < C_p r_m^{(i)} + C_m r_p^{(i)}$ pri čemu je C_m koeficijent uz y u m-toj, a C_p koeficijent uz y u p-toj nejednakosti; tako dobijene nejednakosti vezane su disjunkcijama. Ovo unapređenje Hodesove procedure zasnovano je na izbegavanju nepotrebnih transformacija u konjuktivno ili disjuntivno normalnu formu. Kako se te transformacije ne koriste u Cooperovoj proceduri, ona ne može biti suštinski unapređena na ovaj način.

Ukoliko se Hodesova procedura koristi za teoriju PIA bolje je sve relacije nejednakosti svesti na relaciju $\leq$ nego na $<$ (o tome će više biti reči u D.5.1).

D.2 Cooperova procedura odlučivanja za PIA

Prva verzija Cooperove procedure odlučivanja za PIA [28] bila je zasnovana na standardnoj proceduri eliminacije kvantifikatora za ovu teoriju [51, 72] uz neznatna prilagođavanja neophodna za automatsku primenu. Druga verzija Cooperove procedure [29] predstavljala je kvalitativno unapređenje, jer je uočeno da nije neophodno formulu koja se dokazuje transformisati u disjunktivnu normalnu formu što je tu verziju procedure učinilo znatno efikasnijom (postojala su, u toj verziji, još neka, manje značajna poboljšanja).

U ovom tekstu opisaćemo samo drugu verziju Cooperove procedure [29]; u poglavlju D.5.4 Cooperova procedura biće opisana preciznije i to u terminima prezapisivanja. U opisu Cooperove procedure koriste se i sledeće dodatne definicije (δ je pozitivan ceo broj):

$$\delta | \alpha \overset{def}{\Leftrightarrow} (\exists x)\alpha = \delta x$$

$$\delta \not| \alpha \overset{def}{\Leftrightarrow} \neg((\exists x)\alpha = \delta x)$$

$$x - y = z \overset{def}{\Leftrightarrow} x = y + z\ .$$

U PIA formuli bez promenljivih, moguće je sve atomičke formule zameniti vrednostima $\bot$ i $\top$, nakon čega se i cela formula može svesti na jednu od ove dve vrednosti. Univerzalni kvantifikator može biti zamenjen egzistencijalnim na osnovu sledeće veze $(\forall x)F \equiv \neg(\exists x)\neg F$. Egzistencijalni kvantifikator iz $(\exists x)F$ (gde je F formula bez kvantifikatora) može biti eliminisan sledećim postupkom:

1. eliminiši negaciju koristeći veze $\neg\alpha < \beta \ \equiv\ \beta < \alpha + 1$ i $\neg(\delta|\alpha) \equiv \delta \not| \alpha$.

2. pojednostavi svaku atomičku formulu grupisanjem termova x tako da za svaku atomičku formulu važi da ona ili ne sadrži x ili je jednog od oblika:

   - $\lambda_i x < \alpha_i$
   - $\beta_i < \mu_i x$
   - $\delta_i | \nu_i x + \gamma_i$
   - $\delta_i \not| \nu_i x + \gamma_i$

   gde su λ, μ, ν i δ pozitivni celi brojevi, a α, β i γ su izrazi koji ne sadrže x.

3. neka je δ najmanji zajednički sadržalac brojeva λ, μ i ν; množenjem obe strane svih atomičkih formula pogodnim konstantama, svi koeficijenti uz promenljivu x mogu da budu δ; zameni $(\exists x)F(\delta x)$ formulom $(\exists x)(F(x) \wedge \delta|x)$; nakon toga, formula $(\exists x)(F(x) \wedge \delta|x)$ ne sadrži nijedan znak negacije i za svaku atomičku formulu koju sadrži važi da ona ili ne sadrži x ili je jednog od oblika:

   (A) $x < a_i$

   (B) $b_i < x$

   (C) $\delta_i | x + c_i$

   (D) $\varepsilon_i \not| x + d_i$

4. neka je $F_{-\infty}(x)$ formula dobijena od F zamenjivanjem sa $\top$ svih atomičkih formula oblika a i zamenjivanjem sa $\bot$ svih atomičkih formula oblika b; zameni $(\exists x)F(x)$ formulom

$$\bigvee_{j=1}^{\delta} F_{-\infty}(j) \vee \bigvee_{j=1}^{\delta} \bigvee_{b_i} F(b_i + j)$$

Cooperova procedura je saglasna i potpuna za PIA [29]: ona vraća vrednost $\top$ ako i samo ako je polazna formula teorema teorije PIA (a inače vraća $\bot$).

Ekstremno visoka vremenska složenost Cooperove procedure uslovljena je osobinama teorije PIA. Dokazano je da Cooperova procedura može da razreši svaku PIA rečenicu dužine n u najviše $2^{2^{2^{pn}}}$, koraka gde je p neka konstanta za koju je $p > 1$ [90] (može se dokazati da je dovoljan prostor manji za jedan

eksponent). S druge strane, dokazano je da postoji konstanta $c > 0$ takva da za svaku proceduru odlučivanja za PIA postoji prirodan broj n_0 takav da za svaki prirodan broj n za koji je $n > n_0$ postoji PIA rečenica dužine n za koju procedura odlučivanja mora da potroši više od $2^{2^{cn}}$ koraka nedeterminističkog algoritma da bi je razrešila [45]. Između navedene donje i gornje granice postoji jedan eksponent razlike. Veruje se da cena simulacije nedeterminističkog algoritma na determinističkom ima jedan eksponent (tj. veruje se da je P≠NP), pa se čini malo verovatnim da je moguće popraviti ijednu od ove dve granice (više o složenosti Prezburgerove aritmetike može se videti u [50]). Dakle, može se smatrati da je složenost Cooperove procedure *u najgorem slučaju* $2^{2^{2^{pn}}}$. Eksperimentalni rezultati (poglavlje D.4), međutim, govore da Cooperova procedura nije, u praktičnim primenama, neefikasna koliko bi to moglo da se očekuje na osnovu analize najgoreg slučaja.

D.3 Procedura odlučivanja za PNA a la Cooper

Cooperova procedura odlučivanja za PIA može biti iskorišćena i kao procedura odlučivanja za PNA. Naime, PNA formula F može biti transformisana u PIA formulu F' zamenom promenljivih sorte **N** promenljivama sorte `I` i za svaku promenljivu dodavanjem uslova nenegativnosti (npr. $(\forall x)A(x)$ biće zamenjeno sa $(\forall x')(x' \geq 0 \rightarrow A(x'))$. Nije teško dokazati da je onda F teorema teorije PNA ako i samo ako je F' teorema teorije PIA.

Dodatno, Cooperov algoritam može biti modifikovan tako da bude procedura odlučivanja za PNA, ali i da pri tom jedina sorta koju koristi bude **N** (takav zahtev je opravdan jer tokom primene eliminacije kvantifikatora, za svaki međurezultat može biti primenjena i neka od drugih raspoloživih dokazivačkih strategija za PNA). Između ostalog, to bi znači da više nije raspoloživa operacija oduzimanja $(-)$ definisana kao u prethodnom poglavlju (jer skup **N** nije zatvoren za tako definisano oduzimanje).

Modifikovan Cooperov algoritam za PNA u većini koraka identičan je ili potpuno analogan sa algoritmom za PIA. Slično kao u verziji za PIA, formulu $(\exists x)F(x)$ koja se dokazuje moguće je napisati u obliku $(\exists x)(F(x) \wedge \delta | x)$ koja je sadrži nijedan znak negacije i za svaku atomičku formulu koju sadrži važi da ona ili ne sadrži x ili je jednog od oblika:

(A) $a'_i + x < a''_i$

(B) $b'_i < b''_i + x$

(C) $\delta_i | x + c_i$

(D) $\varepsilon_i \nmid x + d_i$

Nakon toga, formulu $(\exists x)f(x)$ moguće je zameniti ekvivalentnom formulom

$$\bigvee_{j=0}^{\delta-1} f(j) \vee \bigvee_{b'_i} \left(\bigvee_{j=0}^{\delta-1} f(b'_i - b''_i + j) \wedge (b''_i \leq b'_i) \right)$$

što eliminiše kvantifikator $\exists x$.

Poslednji korak sadrži i dodatne korake indukovane činjenicom da skup $\mathbf{N}$ nije zatvoren za oduzimanje: ako u formuli $f(x)$ postoji atomička formula $a'_i + x \leq a''_i$, onda će u $f(b'_i - b''_i + j)$ odgovarajuća atomička formula da bude $a'_i + b'_i + j \leq a''_i + b''_i$; ako u formuli $f(x)$ postoji atomička formula $\delta_i | c_i + x$, onda će u $f(b'_i - b''_i + j)$ odgovarajuća atomička formula da bude $\delta_i | c_i + b'_i + (\delta_i - 1)b''_i + j$ (analogno važi i za preostala dva tipa atomičkih formula). Time se obezbeđuje da je nakon svakog koraka primene algoritma, tekuća formula takođe PNA formula.

D.4 Uporedna analiza Hodesove i Cooperove procedure

Tokom više od dvadeset pet godina mnogo istraživanja uloženo je u modifikacije procedura PRA koje bi aproksimirale procedure odlučivanja za PIA. Boyer i Moore tvrde [17]:

> ... procedure odlučivanja za cele brojeve su veoma komplikovane u poređenju sa mnogim procedurama odlučivanja za linearne nejednakosti nad racionalnim brojevima. [...]. Zato, sledeći tradiciju verifikacije programa, mi smo se odlučili za proceduru za racionalne brojeve ...

Upravo ta "tradicija" je motivisala izvođenje eksperimenata opisanih u ovom poglavlju. Dugo i široko rasprostranjeno uverenje da procedure odlučivanja za PIA ne mogu biti praktično upotrebljive zasnovano je jedino na rezultatima analize najgoreg slučaja [90] i ne na podacima o bilo kakvoj eksperimentalnoj uporednoj analizi. Rezultati opisanih eksperimenata su, iz te perspektive, iznenađujući i dovode u pitanje dugu tradiciju nekorišćenja procedura odlučivanja za PIA: naime, Cooperova procedura (i jedna njena varijanta) bila je efikasnija od Hodesove procedure.

Za eksperimente su korišćena dva korpusa:

- *induktivni korpus* od 121 teoreme iz induktivnih verifikacijskih dokaza;
- *generisani korpus* od 10 000 slučajno generisanih formula.

Ispitivani su veličina formula, broj promenljivih, dokazivost/nedokazivost i CPU vreme utrošeno od strane različitih procedura.

Generisanje Prezburgerovih formula Korpus od 10 000 PNA formula generisan je korišćenjem stohastičke gramatike[5] date u tabeli D.4. Svako pravilo gramatike korišćeno je u skladu sa verovatnoćom datom u desnoj koloni. Generisane su formule bez kvantifikatora, a onda su generisana njihova univerzalna

[5] *Stohastička kontekst–slobodna gramatika* je petorka $G_s = (N, \Sigma, S, R, \phi)$ gde je N konačan skup neterminalnih simbola, Σ konačan alfabet, S izdvojeni početni simbol, R konačan skup produkcionih pravila oblika $A \to \beta$, gde je $A \in N$, $\beta \in (N \cup \Sigma)^*$ i ϕ je funkcija iz R u $[0, 1]$. $\phi(r)$ je verovatnoća koja je pridružena pravilu r iz R i ona "kontroliše" primenu tog

zatvorenja. Skup promenljivih imao je pet elemenata i svaki od njih imao je istu verovatnoću.

#	*Pravilo*	*Verovatnoća*
1.	$\langle\text{formula}\rangle := \langle\text{atomička formula}\rangle$	0.75
2.	$\langle\text{formula}\rangle := (\neg\ \langle\text{formula}\rangle)$	0.10
3.	$\langle\text{formula}\rangle := (\langle\text{formula}\rangle \vee \langle\text{formula}\rangle)$	0.05
4.	$\langle\text{formula}\rangle := (\langle\text{formula}\rangle \wedge \langle\text{formula}\rangle)$	0.05
5.	$\langle\text{formula}\rangle := (\langle\text{formula}\rangle \Rightarrow \langle\text{formula}\rangle)$	0.05
6.	$\langle\text{atomička formula}\rangle := \langle\text{term}\rangle = \langle\text{term}\rangle$	0.20
7.	$\langle\text{atomička formula}\rangle := \langle\text{term}\rangle < \langle\text{term}\rangle$	0.20
8.	$\langle\text{atomička formula}\rangle := \langle\text{term}\rangle \leq \langle\text{term}\rangle$	0.20
9.	$\langle\text{atomička formula}\rangle := \langle\text{term}\rangle > \langle\text{term}\rangle$	0.20
10.	$\langle\text{atomička formula}\rangle := \langle\text{term}\rangle \geq \langle\text{term}\rangle$	0.20
11.	$\langle\text{term}\rangle := (\langle\text{term}\rangle + \langle\text{term}\rangle)$	0.20
12.	$\langle\text{term}\rangle := s(\langle\text{term}\rangle)$	0.20
13.	$\langle\text{term}\rangle := 0$	0.20
14.	$\langle\text{term}\rangle := x \mid y \mid z \mid u \mid v$	0.40

Tabela D.1: Stohastička gramatika za Prezburgerovu aritmetiku korišćena za generisanje eksperimentalnog korpusa.

Razmatrani algoritmi Pored Hodesove i Cooperove procedure, razmatrane su i dve njihove kombinacije sa heuristikom koja brzo odbacuje netačna tvrđenja (zvaćemo je QR heuristika (*quick reject*). Heuristika je definisana na sledeći način: da bismo opovrgli rečenicu $(\forall\vec{x})\Phi(\vec{x})$ dovoljno je dokazati da je neka njena instanca $\Phi(\vec{c})$ netačna. Dakle, instanciramo sve univerzalno kvantifikovane promenljive u formuli $(\forall\vec{x})\Phi(\vec{x})$ određenim konstantnim vrednostima (recimo 0 i 100) na sve načine. Na taj način dobijamo bazne formule $\Phi(\vec{c})$, za koje je dokazivost trivijalno ispitati. Ova jednostavna heuristika je očigledno saglasna, ali nije potpuna. Ipak, rezultati opisanih eksperimenata govore da ova heuristika može biti veoma korisna.[6]

Poredili smo sledeće četiri procedure:

Hodesova procedura. Procedura odlučivanja za PRA. Saglasna (ali ne i kompletna) za univerzalno kvantifikovani fragment teorije PNA i teorije PIA.

pravila. Stohastička gramatika $G_s = (N, \Sigma, S, R, \phi)$ određuje gramatiku $G = (N, \Sigma, S, R)$ koju nazivamo *karakterističnom gramatikom* stohastičke gramatike G_s. Kontekst–slobodna gramatika $G = (N, \Sigma, S, R)$ i funkcija verovatnoće $\phi : R \mapsto [0, 1]$ određuju stohastičku gramatiku $G_s = (N, \Sigma, S, R, \phi)$. Stohastička gramatika je *prâva* ako za svaki neterminalni simbol A važi $\sum_{\beta\in(N\cup\Sigma)^*} \phi(A \to \beta) = 1$. Više o stohastičkim gramatikama videti u [62, 73], a više o složenijim varijantama za sintaksno verovatnosno modelovanje videti npr. i [12, 63, 102].

[6]Ova heuristika može biti korišćena za sve tipove formula (ne samo za univerzalno kvantifikovane): korišćenjem ove procedure možemo transformisati tj. pojednostaviti (saglasno, ali ne i kompletno) datu formulu u egzistencijalno kvantifikovanu formulu i pokušati da je opovrgnemo korišćenjem neke procedure odlučivanja za Prezburgerovu aritmetiku.

U eksperimentima je modifikovana tako da radi sa sortom **N** umesto sa sortom `real`. Razmatrana je procedura implementirana na osnovu originalnog opisa [52] (a ne varijanta sa posebno implementiranim eliminacijama egzistencijalnog i univerzalnog kvantifikatora opisana u poglavlju D.1).

Cooperova procedura. Procedura odlučivanja za PNA. Korišćena je druga, unapređena verzija Cooperove procedure [29] modifikovana za PNA.

DP-A . Procedura odlučivanja za PNA; Definisana je na sledeći način: datu formulu pokušaj da opovrgneš korišćenjem QR heuristike; ako to uspe, formula nije teorema; inače, primeni Cooperovu proceduru; ako ona vrati $\top$, formula je teorema, a inače nije.

DP-B . Procedura odlučivanja za PNA; Definisana je na sledeći način: datu formulu pokušaj da opovrgneš korišćenjem QR heuristike; ako to uspe, formula nije teorema; inače, ako je formula univerzalno kvantifikovana, primeni Hodesovu proceduru; ako ona vrati $\top$, zadata formula je teorema, ako ona vrati $\bot$ ili zadata formula nije univerzalno kvantifikovana, primeni Cooperovu proceduru; ako ona vrati $\top$, zadata formula je teorema, a inače nije.

Opisane procedure primenjene su na sve formule iz dva korpusa. Primenjivan je vremenski limit od 100 sekundi. Rezultati eksperimenata dati su u sledeće četiri tabele; CPU vreme je mereno u milisekundama[7].

Rezultati za induktivni korpus Za sve formule iz ovog korpusa (121) bilo je poznato da su teoreme. Cooperova procedura za sve njih utvrdila je da su teoreme, za prosečno vreme 22ms. Hodesova procedura razrešila je sve formule za prosečno vreme od 110ms, ali 9 odgovora je bilo netačno (zbog nepotpunosti Hodesove procedure za PIA).

	CPU vreme (ms)				
Procedura	$< 10^2$	10^2–10^3	10^3–10^4	10^4–10^5	*Ukupno*
Induktivni korpus					
Hodes	86/22	32/260	3/1100		121/110
Cooper	118/14	3/290			121/20
Generisani korpus					
Hodes	5427/21	3256/316	864/3548	355/32871	9902/1604
Cooper	9480/12	327/360	74/2560	29/31937	9910/136
DP-A	9765/8	201/200	7/2350	8/35000	9981/42
DP-B	9333/8	363/385	184/3470	85/33500	9965/370

Tabela D.2: Broj razrešenih formula (uključujući nepouzdane odgovore Hodesove procedure). (Svako polje je oblika broj razrešenih formula/prosečno CPU vreme).

[7]Programi su napisani u Quintus Prologu; eksperimenti su vršeni na računaru 64Mb Sun SPARC Ultra I.

Doprinos QR heuristike Procedure DP-A i DP-B su korišćenjem QR heuristike brzo odbacile sve osim 70 formula od 7891 formula iz generisanog korpusa za koje se zna da nisu teoreme. Eksperimenti su potvrdili izabrani broj od dve konstante za QR heuristiku: dodatne konstante usporavaju heuristiku, a ne doprinose joj značajno; na drugoj strani, korišćenjem samo konstante 0 heuristika QR odbacila je manje od 5000 formula).

Efekat broja promenljivih Tabela D.4 pokazuje da, kao što je i očekivano, efikasnost svih procedura opada sa porastom broja promenljivih. Zbog transformisanja u disjuntivno normalnu formu, ovaj pad izraženiji je kod Hodesove procedure nego kod Cooperove. Ponašanje procedura DP-A i DP-B je znatno bolje jer slučajno generisane formule sa više promenljivih najčešće nisu bile teoreme i mogle su biti lako odbačene heuristikom QR. Na primer, 55% formula bez promenljivih bile su teoreme, ali najviše 25% formula (19% razrešenih i 6% nerazrešenih) sa 5 promenljivih bile su teoreme teorije PIA.

Procedura	Broj promenljivih/broj formula sa toliko promenljivih					
Induktivni korpus	*0/0*	*1/42*	*2.48*	*3/28*	*4/3*	*5/0*
Hodes		100/17	100/39	100/130	100/683	
Cooper		100/42	100/169	100/1202	100/3608	
Generisani korpus	*0/598*	*1/3362*	*2/3603*	*3/1503*	*4/693*	*5/241*
Hodes	100/0.5	100/10	100/100	100/1000	98.8/9900	63.7/47000
Cooper	100/0.5	100/4	100/12	99.9/70	96.8/1250	72.6/2015
DP-A	100/1.6	100/4.5	100/7.7	100/36	99.3/350	94.2/3592
DP-B	100/1.7	100/6	100/18.1	100/237	99.3/2500	87.9/7700

Tabela D.3: Efekat broja promenljivih na CPU vreme (svako polje u tabeli je oblika procenat uspešno razrešenih formula/prosečno CPU vreme).

Efekat veličine formula Veličina formula (i termova) definisana je kao za 1 uvećan zbir veličina podformula (podtermova) i kao vrednost 1 za promenljive i konstante. Tabela D.4 pokazuje efekat veličine formule na utrošeno CPU vreme. Vreme utrošeno od strane Cooperove procedure raste nešto sporije nego vreme utrošeno od strane Hodesove. Uspešnost procedura sa QR heuristikom opadala je sporije jer verovatnoća da formula nije teorema raste sa njenom veličinom.

Procedura	*Veličina/broj formula*					
Generisani korpus	*1–20/8785*	*21–40/1029*	*41–60/150*	*61–80/29*	*81–100/5*	*101–120/2*
Hodes	100/9	100/232	99.8/3824	88.2/24171	54.8/46990	27.8/64466
Cooper	100/3	100/15	99.5/455	85.6/19317	75.0/1314	50/4256
DP-A	100/4	100/9	99.8/147	98.6/442	89.3/1342	83.3/1258
DP-B	100/5	1002/33	99.8/900	96.9/5082	81.0/9864	77.8/7466

Tabela D.4: Efekat veličine formule na CPU vreme. (Svako polje je oblika procenat razrešenih formula/prosečno CPU vreme; Početna vrsta sadrži brojeve formula u pojedinim grupama.).

Efekat dokazivosti Tabela D.4 pokazuje broj formula razrešenih u okviru datog vremenskog limita (sa prosečnim CPU) vremenom a tabela D.4 sumira rezultate u vezi sa dokazivošću. (Primetimo da ova tabela uključuje negativne odgovore date od strane Hodesove procedure koji nisu "pouzdani" zbog nepoptunosti ove procedure za teoriju PIA).

U induktivnom korpusu i Cooperova i Hodesova procedura razrešile su sve formule, pri čemu je Cooperova procedura bila brža. Hodesova procedura razrešila je netačno 9 od 121 formule (jer te formule nisu tačne u strukturi racionalnih brojeva, a jesu tačne u strukturi celih brojeva).

U generisanom korpusu, od 10 000 formula, bilo je 1968 teorema, 8013 formula nisu bile teoreme, dok preostalih 19 teorema nije razrešeno u okviru datog vremenskog limita. Nasuprot induktivnog korpusa, u ovom skupu formula nepotpunost Hodesove procedure nije došla do izražaja: odgovori Cooperove i Hodesove procedura bili su isti za sve formule razrešene od strane obe procedure. Cooperova procedura nije uspela da razreši 90 formula a Hodesova 98. Ni jednom ni drugom procedurom nije razrešeno 34 formule. Od 90 formula koje nije razrešio Cooperov algoritam, Hodesov algoritam je za 4 utvrdio da su teoreme, a za 52 da nisu (što nije pouzdano zbog nepotpunosti algoritma). Od 98 formula koje nije razrešio Hodesov algoritam, Cooperov je za 13 utvdio da su teoreme, a za 51 da nisu.

Procedura DP-A popravila je rezultate Cooperove procedure: od 90 nerazrešenih, za 71 je utvrđeno da nisu teoreme (nalaženjem kontraprimera). Procedura DP-B bila je nešto bolja utvrdivši da od 90 formula 71 nisu teoreme, a 4 jesu. QR heuristika je uspešno primenjena na većinu ne-teorema i procedure DP-A i DP-B veoma brzo su razrešile ne-teoreme koje nisu mogle da razreše ni Hodesova ni Cooperova procedura.

Procedura DP-B razmatrana je zbog pretpostavke da ona može da uspešno iskoristi Hodesovu proceduru za formule koje su tačne i za racionalne i za cele brojeve (za koje je odgovor Hodesove procedure pouzdan) i pretpostavke da taj dobitak preteže nad gubicima vremena u drugim slučajevima. Bilo je očekivano da prosečno procedura DP-B bude brža od procedure DP-A. Međutim, iznenađujuće, pokazalo se da je Hodesova procedura bila neefikasnija od Cooperove u ovoj grupi formula. Zbog toga DP-B je neefikasnija od DP-A u svakoj grupi formula razvrstanih po dokazivosti. Naravno, DP-B (kao procedura odlučivanja za PIA) ne može iskoristiti negativan odgovor Hodesove procedure, jer je Hodesova procedura nepotpuna; u tim slučajevima mora da bude pozvana Cooperova procedura i vreme koje je potrošio Hodesov algoritam je potrošeno uzalud.

Primeri Ovo je jedna od formula koju je u okviru datog vremenskog limita razrešila Hodesova, ali ne i Cooperova procedura:

$$(\forall y)(\forall z)(\forall x)(\forall u)(\forall v)(\neg(y > ((0 + s(v)) + v) \vee$$

$$((z \leq 0 \vee \neg v < 0) \vee s(u) < ((v + 0) + u)) \wedge s(z) < 0)) \vee$$

$$s(((x + s(u)) + 0)) < (v + 0).$$

Dokazivost	$\models_{pra} F$ $\models_{pia} F$	$\not\models_{pra} F$ $\models_{pia} F$	$\not\models_{pra} F$ $\not\models_{pia} F$	?
Induktivni korpus	*112*	*9*	*0*	*0*
Hodes	107	153		0
Cooper	22			0
Generisani korpus	*1968*	*0*	*8013*	*19*
Hodes	1556		1625	19
Cooper	78		151	98
DP-A	98		3	19
DP-B	1575		90	34

Tabela D.5: Efekat dokazivosti na CPU vreme. (U svakom polju je prosečno CPU vreme za formule iz odgovarajuće grupe; broj formula u tim grupama dat je u početnim vrstama; kolona ? označava nerazrešene formule).

Ovo je jedna od formula koju je u okviru datog vremenskog limita razrešila Cooperova, ali ne i Hodesova procedura:

$$(\forall x)(\forall z)(\forall u)(\forall y)(\forall v)(\neg((s((x+z)) > s(0) \vee (s(0)+v) > s(0)) \vee \neg(x+0) = z)) \Rightarrow$$

$$(x = (0 + (s((z + s(s(u)))) + s(s((y + u))))) \wedge y < v) \vee (0 + 0) < s(v).$$

Zaključak Na induktivnom korpusu, vreme utrošeno od strane Cooperove i Hodesove procedure je uporedivo i štaviše, neočekivano, Cooperova procedure je utrošila manje CPU vremena za ove probleme ("probleme iz stvarnog sveta"). Nepotpunost Hodesove procedure potvrđena je na ovom korpusu.

Teško je izvući opšte zaključke na bazi rezultata koji se odnose na generisani korpus zato što je moguće da su generisani problemi (iz nekog razloga) lakši za Cooperovu nego za Hodesovu proceduru. Eksperimenti bazirani na fazama prelaska [24, 48, 83, 46, 57] bi mogli da pomognu u detektovanju najtežih problema za jednu i drugu proceduru. To bi moglo da bude postignuto menjanjem verovantosnih parametara stohastičke gramatike za Prezburgerovu aritmetiku. Važan je i sledeći faktor: u generisanom korpusu nema mnogo velikih konstanti, a one usporavaju oba algoritma i to posebno Cooperov (na efikasnost SUP-INF algoritma, međutim, ne utiče veličina konstanti).

Za dublje upoznavanje ponašanja Cooperove i Hodesove procedure potrebno je još eksperimenata na različitim korpusima, a posebno na velikim korpusima sa "problemima iz stvarnog sveta" (takva analiza, ipak, nije cilj ovog teksta). Uz sve navedene ograde, mogu se izvesti sledeći zaključci:

- vreme utrošeno od strane Cooperove i Hodesove procedure uporedivo je za realne probleme;
- nepotpunost Hodesove procedure manifestovana je samo na induktivnom korpusu (7% formula pogrešno je razrešeno). Negativan rezultat Hodesove

procedure mora da bude preispitan nekim drugim postupkom. Za većinu PIA ne-teorema, verovatno je i spora procedura odlučivanja brža i robusnija od heurističkih strategija (kao što je npr. indukcija);

- mnoge ne-teoreme mogu jednostavno i brzo da budu odbačene baznim instanciranjem nad malim skupom konstanti;

- analiza najgoreg slučaja može da daje pogrešnu sliku: eksperimentalni rezultati mogu da budu korisni za procenu efikasnosti algoritama za probleme koji se javljaju u praksi.

Na bazi opisanih algoritama, zaključujemo da je Cooperova procedura za PIA, pojačana jendostavnom QR heuristikom prihvatljive efikasnosti (i pri tome je potpuna za teoriju PIA za razliku od Hodesove procedure). Za dublje poznavanje prirode algoritama za PIA potrebni su dodatni eksperimenti i teorijska analiza, ali se čini da Cooperovoj proceduri može da pripadne znatno značajnije mesto u automatskom dokazivanju teorema.

Hodesova procedura, implementirana na osnovu originalnog Hodesovog opisa, pokazala se, iznenađujuće, neefikasnijom od Cooperove procedure na opisanim korpusima. Modifikacija Hodesove procedure opisana u poglavlju D.1 bi dobijene rezultate mogla znatno da popravi na generisanom korpusu, jer ga čine univerzalno kvantifikovane formule i za svaku od njih bila bi dovoljna samo jedna transformacija u konjuktivnu normalnu formu (opisana modifikacija nema efekta ako kvantifikatori u preneks normalnoj formi alterniraju). Ove osobine Hodesove procedure dobrim delom objašanjavaju njeno mesto u dokazivačima teorema — u sistemima koji rade sa logikom bez kvantifikatora, najslabije strane Hodesove procedure ne dolaze do izražaja. S druge strane, modifikacija analogna ovoj ne bi mogla značajno da unapredi Cooperov algoritam.

U kontekstu ugradnje procedura odlučivanja u dokazivače teorema, važan zaključak koji na osnovu uporedne analize nekoliko procedura za PIA može da se izvede je sledeći: za različite klase problema najpogodnije mogu da budu različite procedure; zbog toga njihova ugradnja treba da bude što fleksibilnija — treba da bude omogućeno što jednostavnije ugrađivanje više procedura za istu teoriju i eventualno njihovo dinamičko biranje (u zavisnosti od formule koja se dokazuje). Ovaj kriterijum bio je jedan od ključnih u osnovnim konceptima na kojima su zasnovana rešenja za kombinovanje i ugradnju procedura odlučivanja opisana u prethodnom tekstu.

D.5 Hodesova i Cooperova procedura kao sistemi za prezapisivanje

Procedure odlučivanja za Prezburgerovu aritmetiku moguće je implementirati primenom sistema za prezapisivanje [22, 23]. Slêdeći Bundyjeve ideje iz rada [22], u ovom poglavlju dajemo opis Hodesove i Cooperove procedure kao sistema za prezapisivanja.

PIA formule možemo sintaksno definisati kontekst–slobodnom gramatikom ili korišćenjem Backus-Naurove forme. U Backus-Naurovoj formi (BNF) PIA formulu možemo definisati na sledeći način:

$$\begin{array}{rcl} f & := & af|\neg f|f \vee f|f \wedge f|f \rightarrow f|f \leftrightarrow f|(\exists var)f|(\forall var)f \\ af & := & t = t|t < t|t > t|t \leq t|t \geq t|t \neq t \\ t & := & var|int| - t|t + t|s(t) \\ var & := & x|y|z|\ldots \\ int & := & nat| - nat \\ nat & := & 0|s(nat) \end{array}$$

U navedenoj definiciji f označava formulu, af atomičku formulu, t term, var promenljivu, int ceo a nat prirodan broj. Cooperova procedura koristi dve dodatne relacije: $\delta|\alpha$ označava da je α deljivo pozitivnim celim brojem δ i $\delta \not| \alpha$ označava da α nije deljivo pozitivnim celim brojem δ. Dodatno, pišemo 1 umesto $s(0)$, 2 umesto $s(s(0))$ itd, pa PIA formulu definišemo na sledeći način:

$$\begin{array}{rcl} f & := & af|\neg f|f \vee f|f \wedge f|f \rightarrow f|f \leftrightarrow f|(\exists var)f|(\forall var)f \\ af & := & t = t|t < t|t > t|t \leq t|t \geq t|t \neq t|\ posint|t\ |\ posint \not| t \\ t & := & var|int|t + t|\ int \cdot t \\ var & := & x|y|z|\ldots \\ int & := & nat| - nat \\ nat & := & 0|s(nat)|1|2|\ldots \\ posint & := & s(nat)|1|2|\ldots \end{array}$$

D.5.1 $<$ ili $\leq$?

I Hodesova i Cooperova procedura koriste redukovan broj relacija: Hodesova procedura koristi $<$ i $=$ (ili $\leq$ i $=$) a Cooperova $<$ (ili $\leq$). Cooperov algoritam je procedura odlučivanja za PIA i sva pravila prezapisivanja čuvaju ekvivalenciju (bez obzira na to da li koristi relacija $<$ ili $\leq$). S druge strane, ako se Hodesova procedura koristi za PIA, onda je pogodno (zbog jednostavnosti i efikasnosti) koristiti neka pravila prezapisivanja prilagođena teoriji PIA (na primer, $\neg(t_1 < t_2) \Rightarrow t_2 < t_1 + 1$ može da se koristi umesto $\neg(t_1 < t_2) \Rightarrow\ t_2 < t_1 \vee t - 1 = t_2$). Ovakva prezapisivanja su saglasna, ali ne čuvaju ekvivalenciju u teoriji PRA, pa postoje formule koje su valjane i u PIA i u PRA ali ne mogu da budu dokazane:

$$(\forall x)(1 < x \vee x < 2)$$

$$\Downarrow$$

$$\neg(\exists x)\neg(1 < x \vee x < 2)$$

$$\Downarrow$$

$$\neg(\exists x)(x < 2 \wedge 1 < x)$$

$$\Downarrow$$

$$\neg(\exists x)(1 < 2)$$

$$\Downarrow$$

$$\neg(1 < 2)$$

$$\Downarrow$$

$$2 < 2$$

$$\Downarrow$$

$$false$$

Za istu formulu, ako se koristi samo relacija $\leq$, Hodesova procedura (sa pravilima prezapisivanja $t_1 < t_2 \Rightarrow t_1 + 1 \leq t_2$ i $\neg(t_1 \leq t_2) \Rightarrow t_2 + 1 \leq t_1$) je uspešna:

$$(\forall x)(1 < x \vee x < 2)$$

$$\Downarrow$$

$$(\forall x)(2 \leq x \vee x \leq 1)$$

$$\Downarrow$$

$$\neg(\exists x)\neg(2 \leq x \vee x \leq 1)$$

$$\Downarrow$$

$$\neg(\exists x)(x \leq 1 \wedge 2 \leq x)$$

$$\Downarrow$$

$$\neg(\exists x)(2 \leq 1)$$

$$\Downarrow$$

$$\neg(2 \leq 1)$$

$$\Downarrow$$

$$2 \leq 2$$

$$\Downarrow$$

$$true$$

Relacija $\leq$ se koristi i u implementaciji Hodesove procedure u okviru sistema NQTHM [17], pa ćemo i mi prihvatiti taj izbor kao pogodan. Dodatno, koristićemo tu relaciju i u opisu Cooperove procedure kako bismo istražili sličnosti u strukturi ova dva algoritma.

D.5.2 Normalizacije

U ovom potpoglavlju opisujemo normalizatore potrebne u Hodesovoj i Cooperovoj proceduri. U duhu Bundyjevog pristupa [22], možemo ih smatrati instancama planova dokaza [21, 20, 23]. Normalizacije se odvijaju u etapama i u svakoj etapi se primenjuje jedan zaustavljajući skup pravila za prezapisivanje. Zaustavljanje pojedinačnih podsistema za prezapisivanje (tj. pojedinačnih normalizatora) nije teško dokazati. S druge strane, za neke od tih normalizacija ne postoje konfluentni zaustavljajući sistemi (kao, na primer, za transformaciju u disjunktivnu normalnu formu).

U nastavku navodimo potrebna pravila prezapisivanja i BNF opise ulaznih i izlaznih klasa za neke normalizatore.

Remove *Remove* je plan dokaza koji eliminiše određeni funkcijski ili predikatski simbol ili kvantifikator iz formule. On iscrpno primenjuje jedno pravilo prezapisivanje na tekuću formulu dok ne eliminiše sva pojavljivanja određenog simbola. Za Hodesovu i Cooperovu proceduru potrebna su nam sledeća takva pravila prezapisivanja[8]:

$$\begin{array}{lcl} f_1 \to f_2 & \Rightarrow & \neg f_1 \vee f_2 \\ f_1 \leftrightarrow f_2 & \Rightarrow & (\neg f_1 \vee f_2) \wedge (f_1 \vee \neg f_2) \\ \alpha > \beta & \Rightarrow & \beta + 1 \leq \alpha \\ \alpha < \beta & \Rightarrow & \alpha + 1 \leq \beta \\ \alpha = \beta & \Rightarrow & \alpha \leq \beta \wedge \beta \leq \alpha \\ \alpha \neq \beta & \Rightarrow & \alpha + 1 \leq \beta \vee \beta + 1 \leq \alpha \\ \alpha \geq \beta & \Rightarrow & \beta \leq \alpha \end{array}$$

Na primer, *remove* može da eliminiše logički veznik $\to$ iz formule koristeći pravilo prezapisivanja $f_1 \to f_2 \;\Rightarrow\; \neg f_1 \vee f_2$. Odgovarajuća BNF definicija ulazne klase f je:

$$f \quad := \quad af|\neg f|f \vee f|f \wedge f|f \to f|f \leftrightarrow f|(\exists var)f|(\forall var)f$$

dok je BNF definicija izlazne klase f':

$$f' \quad := \quad af|\neg f'|f' \vee f'|f' \wedge f'|f' \leftrightarrow f'|(\exists var)f'|(\forall var)f' \; .$$

Navedena pravila prezapisivanja omogućavaju nam da transformišemo polaznu BNF definiciju formula Prezburgerove aritmetike u sledeću definiciju:

$$\begin{array}{rcl} f' & := & af'|\neg f'|f' \vee f'|f' \wedge f'|(\exists var)f'|(\forall var)f' \\ af' & := & t < t|\;\; int|t \;\;|\;\; int \not| t \\ t & := & var|int|-t|t+t \\ var & := & x|y|z|\ldots \\ int & := & nat|-nat \\ nat & := & 0|s(nat) \end{array}$$

Plan *remove* može da ima i složenije oblike; on može da koristi pravila prezapisivanja kao što su:

$$\begin{array}{lcl} \neg(t_1 \leq t_2) & \Rightarrow & t_2 + 1 \leq t_1 \\ \neg(\delta|a) & \Rightarrow & \delta \not| a \\ \neg(\delta \not| a) & \Rightarrow & \delta|a \end{array}$$

[8] Ako bismo koristili relaciju $<$ bila bi potrebna neka druga pravla.

Stratify *Stratify* je plan dokaza koji razdvaja klasu u dva nivoa koji koriste samo neke[9] specifične veznike, predikatske ili funkcijske simbole. Suštinski, *stratify* "pomera" neke simbole "ispod" nekih drugih simbola.

U Hodesovoj i Cooperovoj proceduri koristimo plan *stratify* za konvertovanje formule u preneks normalnu formu, za pomeranje negacija ispod disjunkcija i konjuncija, za pomeranje konjunkcija ispod dijsunkcija (samo za Hodesov algoritam) i za pomeranje množenja ispod sabiranja.

Za konvertovanje formule u preneks normalnu formu, ulazna BNF klasa je:

$$f \quad := \quad af|\neg f|f \vee f|f \wedge f|(\exists var)f|(\forall var)f \ ;$$

ona je razdvojena u dve izlazne klase:

$$\begin{aligned} f' \quad &:= \quad f''|(\exists var)f'|(\forall var)f' \ , \\ f'' \quad &:= \quad af|\neg f''|f'' \vee f''|f'' \wedge f'' \end{aligned}$$

i odgovarajuća pravila prezapisivanja su:

$$\begin{aligned} \neg(\forall x)f_1 \quad &\Rightarrow \quad (\exists x)\neg f_1 \\ \neg(\exists x)f_1 \quad &\Rightarrow \quad (\forall x)\neg f_1 \\ (\forall x)f_1 \wedge f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \wedge f_2) \\ (\exists x)f_1 \wedge f_2 \quad &\Rightarrow \quad (\exists x)(f_1 \wedge f_2) \\ f_1 \wedge (\forall x)f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \wedge f_2) \\ f_1 \wedge (\exists x)f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \wedge f_2) \\ (\forall x)f_1 \vee f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \vee f_2) \\ (\exists x)f_1 \vee f_2 \quad &\Rightarrow \quad (\exists x)(f_1 \vee f_2) \\ f_1 \vee (\forall x)f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \vee f_2) \\ f_1 \vee (\exists x)f_2 \quad &\Rightarrow \quad (\forall x)(f_1 \vee f_2) \end{aligned}$$

Za pomeranje negacija ispod disjunkcija i konjunkcija, ulazna BNF klasa je oblika:

$$f \quad := \quad af|\neg f|f \vee f|f \wedge f \ ;$$

ona je razdvojena u dve izlazne klase:

$$\begin{aligned} f' \quad &:= \quad f''|f' \vee f'|f' \wedge f' \\ f'' \quad &:= \quad af|\neg f'' \ ; \end{aligned}$$

i odgovarajuća pravila prezapisivanja su:

$$\begin{aligned} \neg(f_1 \vee f_2) \quad &\Rightarrow \quad \neg f_1 \wedge \neg f_2 \\ \neg(f_1 \wedge f_2) \quad &\Rightarrow \quad \neg f_1 \vee \neg f_2 \end{aligned}$$

Za pomeranje konjunkcija ispod disjunkcija, ulazna BNF klasa je:

$$f \quad := \quad f'''|f \vee f|f \wedge f \ ;$$

ona je razdvojena u dve izlazne klase:

$$\begin{aligned} f' \quad &:= \quad f''|f' \vee f' \\ f'' \quad &:= \quad f'''|f'' \wedge f'' \ ; \end{aligned}$$

i odgovarajuća pravila prezapisivanja su:

$$\begin{aligned} f_1 \wedge (f_2 \vee f_3) \quad &\Rightarrow \quad (f_1 \wedge f_2) \vee (f_1 \wedge f_3) \\ (f_2 \vee f_3) \wedge f_1 \quad &\Rightarrow \quad (f_2 \wedge f_1) \vee (f_3 \wedge f_1) \end{aligned}$$

Za pomeranje množenja ispod sabiranja, ulazna BNF klasa je:

$$t \quad := \quad var|int|t + t|\ int \cdot t \ ;$$

ona je razdvojena u dve izlazna klase:

[9] U radu [22], izlaz plana *stratify* je skup od više nivoa, od kojih svaki sadrži *tačno jedan* predikatski ili funkcijski simbol. Čini se da ovo ograničenje može da bude oslabljeno i onda bi, na primer, konvertovanje u preneks normalnu formu moglo da bude opisano ovim planom.

$$t' \quad := \quad t''|t' + t'$$
$$t'' \quad := \quad var|int|\ int \cdot t'' \ ;$$

i odgovarajuća pravila prezapisivanja su:

$$i \cdot (t_1 + t_2) \quad \Rightarrow \quad i \cdot t_1 + i \cdot t_2$$

Thin *Thin* je plan dokaza koji iscrpno primenjuje pravilo prezapisivanja koje eliminiše višestruka pojavljivanja nekog simbola. Ulazna BNF klasa je:

$$f \quad := \quad af|\neg f \ ;$$

izlazna klasa je:

$$f' \quad := \quad af|\neg af \ ;$$

i odgovarajuće pravilo prezapisivanja je:

$$\neg\neg f \quad \Rightarrow \quad f.$$

Reduce *reduce* je plan dokaza koji smanjuje (do najviše jedan) broj pojavljivanja nekog simbola (primetimo da u ovom planu dokaza slabimo uslov eliminisanja iz *remove* plana). I u ovom planu se odgovarajuće pravilo prezapisivanja primenjuje iscrpno. Mi ćemo ga koristiti za smanjivanje broja baznih atomičkih formula u formuli koja se dokazuje. Potrebna su nam sledeća pravila prezapisivanja:

$$f_1 \vee f \quad \Rightarrow \quad 0 \leq 0$$
$$f_1 \wedge f \quad \Rightarrow \quad f$$
$$f \vee f_1 \quad \Rightarrow \quad 0 \leq 0$$
$$f \wedge f_1 \quad \Rightarrow \quad f$$
$$f_2 \vee f \quad \Rightarrow \quad f$$
$$f_2 \wedge f \quad \Rightarrow \quad 1 \leq 0$$
$$f \vee f_2 \quad \Rightarrow \quad f$$
$$f \wedge f_2 \quad \Rightarrow \quad 1 \leq 0$$

gde f_1 ima jedan od sledećih oblika:

$0 \leq a \cdot 1$ gde je a nenegativan ceo broj;
$0 = b \cdot 1$ gde je b jednako 0;
$\delta|c \cdot 1$ gde je c ceo broj deljiv sa δ;
$\delta \not| c \cdot 1$ gde je c ceo broj koji nije deljiv sa δ;

a f_2 je jednog od sledećih oblika:

$0 \leq a \cdot 1$ gde je a negativan ceo broj;
$0 = b \cdot 1$ gde je b različito od 0;
$\delta|c \cdot 1$ gde je c ceo broj koji nije deljiv sa δ;
$\delta \not| c \cdot 1$ gde je c ceo broj deljiv sa δ.

Left-Association *left-assoc* je jedan od planova dokaza za reorganizovanje unutar jedne klase [22]. Ako neka klasa sadrži samo jedan funkcijski simbol i ako je taj funkcijski simbol binaran i asocijativan, onda članovi te klase mogu da budu transformisani u levo asocijativnu formu korišćenjem plana *left-assoc*. Korišćenje plana *left-assoc* omogućiće jednostavnije planove dokaza za rešavanje nejednakosti.

Za levu asocijativnost sabiranja, ulazna BNF klasa je

$t \quad := \quad t''|t + t$;

izlazna klasa je:

$t' \quad := \quad t''|t' + t''$;

a odgovarajuće pravilo prezapisivanje je:

$(t_1 + (t_2 + t_3)) \quad \Rightarrow \quad ((t_1 + t_2) + t_3)$

Za levu asocijativnost množenja, ulazna BNF klasa je oblika

$t \quad := \quad t''|t \cdot t$;

izlazna klasa je:

$t' \quad := \quad t''|t' \cdot t''$;

i odgovarajuće pravilo prezapisivanja je:

$(t_1 \cdot (t_2 \cdot t_3)) \quad \Rightarrow \quad ((t_1 \cdot t_2) \cdot t_3)$

Poly-form *poly-form* je plan dokaza koji ćemo koristiti za transformisanje u polinomijalnu normalnu formu formule koje pripadaju klasi:

$$\begin{aligned} t &:= summand|t + summand\ ; \\ summand &:= var|intprod|intprod \cdot var\ ; \\ intprod &:= int|intprod \cdot int\ ; \end{aligned}$$

Izlazna klasa je:

$$\begin{aligned} t &:= summand'|t + summand'\ ; \\ summand' &:= int \cdot var|int \cdot 1\ ; \end{aligned}$$

a odgovarajuće pravilo prezapisivanja:

$$\begin{aligned} (i_1 \cdot i_2) \cdot v &\Rightarrow i_3 \cdot v \\ (i_1 \cdot i_2) \cdot 1 &\Rightarrow i_3 \cdot 1 \\ i_1 \cdot i_2 &\Rightarrow i_3 \cdot 1 \\ i &\Rightarrow i \cdot 1 \end{aligned}$$

gde su i, i_1, i_2 celi brojevi i uslovi $i_1 \cdot i_2 = i_3$, $i \neq 1$ mora da budu zadovoljeni. (Četvrto pravilo treba primenjivati na termove t_1, t_2 u atomičkim formulama $t_1 \leq t_2$, $t_1 = t_2$, $\delta|t_1$, $\delta \not| t_1$ samo ako su oni celi brojevi.)

Reorder *reorder* je još jedan plan dokaza za reorganizovanje u okviru jedne klase. Ako neka klasa sadrži samo jedan funkcijski simbol i one je komutativan, onda ovaj plan dokaza može da se koristi za preuređivanje argumenata u okviru terma (za koji se pretpostavlja da je u levoj asocijativnoj formi). Mi ćemo da koristiti za preuređivanje argumenata u termu koji je u polinomijalnoj normalnoj formi.

Ulazna BNF klasa

$t \quad := \quad t'|t + t'$;

se ne menja ovim planom dokaza i on ne može biti opisan na čisto sintaksan način.

Odgovarajuća pravila prezapisivanja su:

$$\begin{aligned} i_1 \cdot v_1 + i_2 \cdot v_2 &\Rightarrow i_2 \cdot v_2 + i_1 \cdot v_1 \\ (t + i_1 \cdot v_1) + i_2 \cdot v_2 &\Rightarrow (t + i_2 \cdot v_2) + i_1 \cdot v_1 \end{aligned}$$

ali ona su primenjiva samo ako je uslov $v_2 \prec v_1$ zadovoljen, gde je $\prec$ neko uređenje nad skupom $var \cup \{1\}$. Dakle, za ovaj plan dokaza potrebne su određene meta funkcije koje izračunavanju uređenje $\prec$. Mi ćemo ga koristiti kako bi bio jednostavniji plan dokaza *collect* za neku promenljivu ili konstantu 1. Dakle,

za ovu konkretnu primenu, uređenje $\prec$ može da bude parcijalno uređenje skupa $var \cup \{1\}$, definisano za neki njegov element v na sledeći način:

$$u_1 \prec u_2 \quad \Leftrightarrow \quad u_1 \in (var \cup \{1\}) \setminus v \ \wedge \ u_2 = v \ .$$

Sa ovim ograničenjima, efekat plana *reorder* može biti reprezentovan sintaksno pojedinačno za svaki element skupa $var \cup \{1\}$. Na primer, ulazna BNF klasa za preuređivanje za promenljivu x ima oblik:

$$\begin{aligned} t &:= t'|t + summand \\ summand &:= int \cdot var|int \cdot 1 \\ var &:= x|y|z|\ldots \ ; \end{aligned}$$

a izlazna klasa je:

$$\begin{aligned} t &:= t'|t + int \cdot x \\ t' &:= summand|t' + summand \\ summand &:= int \cdot var'|int \cdot 1 \\ var' &:= y|z|\ldots \ ; \end{aligned}$$

Collect *collect* je plan dokaza koji se koristi za smanjivanje broja pojavljivanja neke promenljive u okviru terma.

BNF klasa se ne menja primenom ovog plana, ukoliko on nije proširen (slično kao plan *reorder*).

Odgovarajuća pravila prezapisivanja su:

$$\begin{aligned} i_1 \cdot v + i_2 \cdot v &\Rightarrow i_3 \cdot v \\ (t + i_1 \cdot v) + i_2 \cdot v &\Rightarrow t + i_3 \cdot v \end{aligned}$$

gde je $i_1 + i_2 = i_3$. Koristimo i dodatna četiri pravila:

$$\begin{aligned} i_1 \cdot v + i_2 \cdot v &\Rightarrow 0 \cdot 1 \\ (t + i_1 \cdot v) + i_2 \cdot v &\Rightarrow t \\ t + 0 \cdot v &\Rightarrow t \\ 0 \cdot v &\Rightarrow 0 \cdot 1 \end{aligned}$$

gde je $i_1 + i_2 = 0$. Efekat plana *collect* može biti reprezentovan sintaksno pojedinačno za svaki element skupa $var \cup \{1\}$. Na primer, ulazna BNF klasa za "sakupljanje" promenljivih x je oblika:

$$\begin{aligned} t &:= t'|t + int \cdot x \\ t' &:= summand|t' + summand \\ summand &:= int \cdot var|int \cdot 1 \\ var &:= y|z|\ldots \ ; \end{aligned}$$

a izlazna klasa je:

$$\begin{aligned} t &:= t'|t' + nonzeroint \cdot x \\ t' &:= summand|t' + summand \\ summand &:= int \cdot var|int \cdot 1 \\ var &:= y|z|\ldots \ ; \\ nonzeroint &:= posint| - posint \ ; \end{aligned}$$

Isolate *isolate* je plan dokaza koji koristimo za izolovanje neke promenljive u okviru atomičke formule. Odgovarajuća pravila prezapisivanje su:

$$
\begin{array}{lcl}
0 \leq t + i \cdot x & \Rightarrow & (-1) \cdot t \leq i \cdot x \\
0 \leq t + i_1 \cdot x & \Rightarrow & i_2 \cdot x \leq t \\
0 \leq i \cdot x & \Rightarrow & 0 \leq i \cdot x \\
0 \leq i_1 \cdot x & \Rightarrow & i_2 \cdot x \leq 0 \\
0 = t + i \cdot x & \Rightarrow & (-1) \cdot t = i \cdot x \\
0 = t + i_1 \cdot x & \Rightarrow & t = i_2 \cdot x \\
0 = i \cdot x & \Rightarrow & 0 = i \cdot x \\
0 = i_1 \cdot x & \Rightarrow & 0 = i_2 \cdot x
\end{array}
$$

gde je $i_2 = -i_1$. Efekat plana *isolate* može biti opisan sintaksno pojedinačno za svaki elemenat skupa $var \cup \{1\}$. Na primer, ulazna BNF klasa za izolovanje promenljive x je oblika:

$$
\begin{array}{rcl}
af & := & 0 = t | 0 \leq t | \ posint | t \ \mid \ posint \not| t \\
t & := & t' | t' + int \cdot x \\
t' & := & summand | t' + summand \\
summand & := & int \cdot var | int \cdot 1 \\
var & := & y | z | \ldots \ ;
\end{array}
$$

a izlazna klasa je:

$$
\begin{array}{rcl}
af & := & int \cdot t' = int \cdot x | int \cdot t' \leq int \cdot x | int \cdot x \leq int \cdot t' \\
 & & \mid \ posint | t \ \mid \ posint \not| t \\
t' & := & summand | t' + summand \\
summand & := & int \cdot var | int \cdot 1 \\
var & := & y | z | \ldots \ ;
\end{array}
$$

D.5.3 Hodesov algoritam za PIA zasnovan na primeni planova za normalizaciju

Kao što je već rečeno, Hodesov algoritam je procedura odlučivanja za PRA. On se može koristiti i kao saglasna, ali nepotpuna procedura za PIA. Podrazumevamo da je formula koja se dokazuje zatvorena (ako nije razmatramo njeno univerzalno zatvorenje). Takođe, podrazumevamo da su sve promenljive standardizovane, tj. nema promenljivih istog imena a sa različitim dosezima. Za datu PIA formulu f, Hodesov algoritam može, korišćenjem normalizacija, biti opisan na sledeći način:

1. upotrebi *remove* za eliminisanje simbola $\rightarrow$ i $\leftrightarrow$;

2. upotrebi *remove* za eliminisanje simbola $>$, $<$, $\neq$, $\geq$;

3. upotrebi *stratify* za pomeranje simbola $\vee$, $\wedge$ i $\neg$ "ispod" kvantifikatora (tj. transformiši formulu u preneks normalnu formu);

4. primeni sledeće korake (4a-4h) za tekući unutrašnji kvantifikator (ako postoji); ako je tekući unutrašnji kvantifikator univerzalan, onda zameni $(\forall x)f$ formulom $\neg(\exists x)\neg f$ (gde je x promenljiva koja odgovara tekućem unutrašnjem kvantifikatoru);

   4a. upotrebi *stratify* za pomeranje simbola $\neg$ ispod simbola $\wedge$ and $\vee$;

4b. upotrebi *thin* za eliminisanje višestrukih pojavljivanja simbola $\neg$;

4c. upotrebi *remove* za eliminisanje simbola $\neg$;

4d. Zameni sve atomičke formule $t_1 = t_2$ sa $0 = t_2 + (-1) \cdot t_2$ i $t_1 \leq t_2$ formulama $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkim formulama $0 = t$ i $0 \leq t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod $+$s, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za promenljivu x, upotrebi *collect* za promenljivu x i onda upotrebi *isolate* za promenljivu x. Nakon toga, svaka atomička formula ili ne sadrži x ili je jednog od sledećih oblika:

$\lambda_i x \leq \alpha_i$

$\beta_i \leq \mu_i x$

$\gamma_i = \nu_i x$

gde su λ_i, μ_i i ν_i pozitivni celi brojevi i α_i, β_i su γ_i termovi koji ne sadrže x.

4e. upotrebi *stratify* za pomeranje simbola $\wedge$ ispod simbola $\vee$, tj. konvertuj formulu u disjunktivnu normalnu formu (nakon čega $(\exists x) f(x)$ ima oblik: $(\exists x)(f_1(x) \vee f_2(x) \vee \ldots f_n(x))$;

4f. za svaki disjunkt f_i u $(\exists x)(f_1(x) \vee f_2(x) \vee \ldots f_n(x))$ primeni sledeće korake:

- ako postoji jednakost oblika $\gamma = \nu x$ u formuli f_i, prezapiši svaku atomičku formulu oblika $\lambda_i x \leq \alpha_i$ u $\lambda_i \gamma \leq \nu \alpha_i$; prezapiši svaku atomičku formulu oblika $\beta_i \leq \mu_i x$ u $\nu \beta_i \leq \mu_i \gamma$; prezapiši svaku atomičku formulu oblika $\gamma_i = \nu_i x$ u $\nu \gamma_i = \nu_i \gamma$ (i.e. prezapiši $\gamma = \nu \wedge \bigwedge_i \lambda_i x \leq \alpha_i \wedge \bigwedge_j \beta_j \leq \mu_j x \wedge \bigwedge_k \gamma_k = \nu_k x$ u

$$\bigwedge_i \lambda_i \gamma \leq \nu \alpha_i \wedge \bigwedge_j \nu \beta_j \leq \mu_j \gamma \wedge \bigwedge_k \nu \gamma_k = \nu_k \gamma \ ;)$$

- ako ne postoje jednakosti oblika $\gamma = \nu x$ u formuli f_i, prezapiši sve parove oblika $\lambda_i x \leq \alpha_i \wedge \beta_j \leq \mu_j x$ u $\lambda_i \beta_j \leq \mu_j \alpha_i$ (tj. prezapiši $\bigwedge_i \lambda_i x \leq \alpha_i \wedge \bigwedge_j \beta_j \leq \mu_j x$ u

$$\bigwedge_{i,j} \lambda_i \beta_j \leq \mu_j \alpha_i \ );$$

i zatim eliminiši tekući unutrašnji kvantifikator (tj. kvantifikator $\exists x$) jer je on redudantan;

4g. Prezapiši sve atomičke formule $t_1 = t_2$ u $0 = t_2 + (-1) \cdot t_2$ i formule $t_1 \leq t_2$ u $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkoj formuli $0 = t$ i $0 \leq t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za konstantu 1, upotrebi *collect* za konstantu 1 i onda upotrebi *isolate* za konstantu 1. Nakon toga, svaka atomička formula je jednog od sledećih oblika:

$0 \leq \sum_{j=1}^{k} \alpha_i x_i + a_i \cdot 1$

$0 \leq \sum_{j=1}^{k} \alpha_i x_i$

$0 \leq a_i \cdot 1$

$0 = \sum_{j=1}^{k} \beta_i x_i + b_i \cdot 1$

$0 = \sum_{j=1}^{k} \beta_i x_i$

$0 = b_i \cdot 1$

gde su a_i, α_i, b_i, β_i celi brojevi, (simboli x_i označavaju promenljive).

4h. upotrebi *reduce* za smanjivanje broja baznih atomičkih formula (na najviše jedan);

5. upotrebi *stratify* za pomeranje simbola $\neg$ ispod simbola $\wedge$ i $\vee$;

6. upotrebi *thin* za elminisanje višestrukih pojavljivanja simbola $\neg$;

7. upotrebi *remove* za eliminisanje simbola $\neg$;

8. pojednostavi sve atomičke formule sakupljanjem svih konstanti, tj. prezapiši sve atomičke formule $t_1 = t_2$ u $0 = t_2 + (-1) \cdot t_2$ i atomičke formule $t_1 \leq t_2$ u $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkoj formuli $0 = t$ and $0 \leq t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za konstantu 1, upotrebi *collect* za konstantu 1 i onda upotrebi *isolate* za konstantu 1. Nakon toga, svaka atomička formula ima jedan od sledećih oblika:

   $0 \leq a_i \cdot 1$

   $0 = b_i \cdot 1$

   gde su a_i, b_i celi brojebi.

9. Upotrebi *reduce* za smanjivanje broja baznih atomičkih formula (na najviše jedan).

10. Ako je ulazna formula prezapisana u $0 \leq 0$, onda je ona valjana (u PIA); inače (tj. ako je prezapisana u $1 \leq 0$), ona nije valjana.

Primetimo da koraci (4g) i (4h) mogu biti eliminisani, ali oni povećavaju efikasnost algoritma.

Kao što je pokazano, većina koraka Hodesovog algoritma može biti opisana na čisto sintaksan način — u terminima primena pravila prezapisivanja, tj. u terminima primena normalizacija. Preostali koraci ne mogu biti opisani tako, ali deo njeih može biti opisan primenom uslovnog prezapisivanja ili prezapisivanje višeg reda (što ovde neće biti razmatrano).

D.5.4 Cooperov algoritam za PIA zasnovan na primeni planova za normalizaciju

Kao i Hodesov algoritam, Cooperov algoritam je baziran na ideji uzastopne eliminacije kvantifikatora. Dobijena formula bez kvantifikatora može biti razrešena jednostavnim postupkom. Podrazumevamo da je formula koja se dokazuje zatvorena (ako nije razmatramo njeno univerzalno zatvorenje). Takođe, podrazumevamo da su sve promenljive standardizovane, tj. nema promenljivih istog imena a sa različitim dosezima. Za datu PIA formulu f, Cooperov algoritam može, korišćenjem normalizacija, biti opisan na sledeći način [10]:

1. upotrebi *remove* za eliminisanje simbola $\rightarrow$ i $\leftrightarrow$;
2. upotrebi *remove* za eliminisanje simbola $>$, $<$, $\neq$, $\geq$;
3. upotrebi *stratify* za pomeranje simbola $\vee$, $\wedge$ i $\neg$ ispod kvantifikatora (tj. transformiši formulu u preneks normalnu formu);
4. primeni naredne korake (4a0-4h) za tekući unutrašnji kvantifikator (ako postoji); ako je tekući unutrašnji kvantifikator univerzalan, onda zameni $(\forall x)f$ formulom $\neg(\exists x)\neg f$ (gde je x promenljiva koja odgovara tekućem unutrašnjem kvantifikatoru);

   4a0 [11] upotrebi *remove* za eliminisanje simbola $=$;

   4a. upotrebi *stratify* za pomeranje simbola $\neg$ ispod simbola $\wedge$ i $\vee$;

   4b. upotrebi *thin* za eliminisanje višestrukih simbola $\neg$;

   4c. upotrebi *remove* za eliminisanje simbola $\neg$;

   4d. Prezapiši sve atomičke formule $t_1 \leq t_2$ u $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkim formulama $0 \leq t$, $i|t$, $i \not| t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za promenljivu x, upotrebi *collect* za promenljivu x i onda (samo za atomičke formule $0 \leq t$) upotrebi *isolate* za promenljivu x. Nakon toga, svaka atomička formula ili ne sadrži x ili je jednog od sledećih oblika:

[10] Predstavljeni algoritam prilagođen je ideji normalizacija i razlikuje se od originalnog algoritma u nekoliko detalja.

[11] Primetimo da simbol $=$ može biti eliminisan u koraku 2 (zajedno sa simbolima $<$, $>$, $\geq$, $\neq$). Ovaj korak je primenjen na ovoj poziciji zbog sledećih razloga: *(i)* na ovaj način, Hodesova i Cooperova procedura imaju najvećim delom istu strukturu (i mogu deliti najveći deo kôda); *(ii)* na ovaj način, Hodesova i Cooperova procedura biti korišćene od strane nekog nad-algoritma koji bi eliminisao unutrašnji kvantifikator primenom koraka $4-{-}4j$ ili jedne ili druge procedure (u zavisnosti od formule koja se dokazuje); ako Cooperova procedura vrati formulu bez simbola $|$ i $\not|$, onda i Hodesova procedura može biti upotrebljena za eliminaciju narednog unutrašnjeg kvantifikatora; s druge strane, ako Hodesova procedura vrati formulu koja uključuje simbol $=$, onda on može biti eliminisan kako bi Cooperova procedura mogla da bude upotrebljena za eliminaciju narednog unutrašnjeg kvantifikatora. Ovaj pristup može biti koristan u praktičnim primenama, jer i za Cooperova i za Hodesov algoritam postoje skupovi formula za koje je on efikasniji.

$\lambda_i x \leq \alpha_i$
$\beta_i \leq \mu_i x$
$\delta_i | \gamma_i + \nu_i x$
$\eta_i \not| \varepsilon_i + \xi_i x$
gde su λ_i, μ_i, ν_i, δ_i, ξ_i i η_i pozitivni celi brojevi i α_i, β_i, γ_i i ε_i su termovi koji ne sadrže x.

4e. Neka je δ NZD svih vrednosti λ_i, μ_i, ν_i and ξ_i. Prezapiši sve atomičke formule $\lambda_i x \leq \alpha_i$ u $\delta x \leq \phi_i \cdot \alpha_i$ gde je $\phi_i = \delta/\lambda_i$ (primetimo da je ϕ_i (pozitivan) ceo broj). Prtezapiši sve atomičke formule $\beta_i \leq \mu_i x$ u $\phi_i \cdot \beta_i \leq \delta x$ gde je $\phi_i = \delta/\mu_i$ (ϕ_i je (pozitivan) ceo broj). Prezapiši sve atomičke formule $\delta_i | \gamma_i + \nu_i x$ u $\psi_i | \phi_i \cdot \gamma_i + \delta x$ gde je $\phi_i = \delta/\nu_i$ and $\psi_i = \delta_i \cdot \phi_i$ (ϕ_i je pozitivan ceo broj). Prezapiši sve atomičke formule $\eta_i \not| \varepsilon_i + \xi_i x$ u $\psi_i \not| \phi_i \cdot \varepsilon_i + \delta x$ gde je $\phi_i = \delta/\xi_i$ i $\psi_i = \eta_i \cdot \phi_i$ (ϕ_i je pozitivan ceo broj). Nakon toga, zameni $(\exists x) f(\delta x)$ formulom $(\exists x)(f(x) \wedge \delta | x)$, u kojoj su svi koeficijenti uz promenljive x jednaki 1. Dakle, u novoj formuli svaka atomička formula ili ne sadrži promenljivu x ili je jednog od sledećih oblika:

(i) $x \leq a_i$
(ii) $b_i \leq x$
(iii) $\delta_i | c_i + x$
(iv) $\eta_i \not| d_i + x$,

gde su a_i, b_i, c_i, d_i izrazi koji ne sadrže x i δ_i, η_i su pozitivni celi brojevi.

4f. Neka je δ NZD svih vrednosti δ_i i η_i i neka je formula $f_{-\infty}(x)$ dobijena od $f(x)$ zamenjivanjem svih formula tipa (i) formulama $0 \leq 0$ i zamenjivanjem svih formula tipa (ii) formulama $1 \leq 0$. Zameni $(\exists x) f(x)$ formulom[12]

$$\bigvee_{j=0}^{\delta-1} f_{-\infty}(j) \vee \bigvee_{j=0}^{\delta-1} \bigvee_{b_i} f(b_i + j)$$

4g. Prezapiši sve atomičke formule $t_1 \leq t_2$ u $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkim formulama $0 \leq t$, $i|t$, $i \not| t$ upotrebi *stratify* da pomeriš simbole $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za konstantu 1, upotrebi *collect* za konstantu 1 i (samo za atomičke formule $0 \leq t$) upotrebi *isolate* za konstantu 1. Nakon toga, svaka atomička formula je jednog od sledećih oblika:
$0 \leq \sum_{j=1}^{k} \alpha_i x_i + a_i \cdot 1$
$0 \leq \sum_{j=1}^{k} \alpha_i x_i$
$0 \leq a_i \cdot 1$

[12]Primetimo da je navedena disjunkcija $\bigvee_{j=0}^{\delta-1}$ u originalnom radu [29] $\bigvee_{j=1}^{\delta}$ jer je u njemu korišćena relacija $<$ umesto $\leq$.

$\delta_i | (\sum_{j=1}^{k} \gamma_i x_i + c_i \cdot 1)$
$\delta_i | \sum_{j=1}^{k} \gamma_i x_i$
$\delta_i | c_i \cdot 1$
$\eta_i \not| (\sum_{j=1}^{k} \varepsilon_i x_i + d_i \cdot 1)$,
$\eta_i \not| \sum_{j=1}^{k} \varepsilon_i x_i$,
$\eta_i \not| d_i \cdot 1$,
gde su a_i, α_i, c_i, γ_i, d_i, ε_i celi brojevi, δ_i, η_i su pozitivni celi brojevi i simboli x_i označavaju promenljive.

4h. upotrebi *reduce* za smanjivanje broja baznih atomičkih formula (na najviše jednu);

5. upotrebi *stratify* za pomeranje simbola $\neg$ ispod simbola $\wedge$ i $\vee$;

6. upotrebi *thin* za eliminisanje višestrukih pojavljivanja simbola $\neg$;

7. upotrebi *remove* za eliminisanje simbola $\neg$;

8. pojednostavi sve atomičke formule sakupljanjem svih konstanti, tj. prezapiši sve atomičke formule $t_1 \leq t_2$ u $0 \leq t_2 + (-1) \cdot t_2$; za sve termove t u atomičkim formulama $0 \leq t$, $i|t$, $i \not| t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za konstantu 1, upotrebi *collect* za konstantu 1 i onda (samo za atomičke formule oblika $0 \leq t$) upotrebi *isolate* za konstantu 1. Nakon toga, svaka atomička formula je jednog od sledećih oblika:

$0 \leq a_i \cdot 1$
$\delta_i | c_i \cdot 1$
$\eta_i \not| d_i \cdot 1$,
gde su a_i, c_i i d_i celi brojevi i δ_i, η_i su pozitivni celi brojevi.

9. upotrebi *reduce* za smanjivanja broja baznih atomičkih formula (na najviše jedan);

10. ako je ulazna formula prezapisana u $0 \leq 0$, onda je ona valjana (u PIA); inače (tj. ako je ulazna formula prezapisana u $1 \leq 0$), ona nije valjana

Primetimo da koraci (4g) i (4h) mogu da budu izostavljeni, ali oni povećavaju efikasnost algoritma.

Kao što je pokazano, većina koraka Cooperovog algoritma može biti opisana na čisto sintaksan način — u terminima primena pravila prezapisivanja, tj. u terminima primena normalizacija. Preostali koraci ne mogu biti opisani tako, ali deo njeih može biti opisan primenom uslovnog prezapisivanja ili prezapisivanje višeg reda (što ovde neće biti razmatrano).

D.5.5 Algoritam za PNA a la Cooper zasnovan na primeni planova za normalizaciju

Cooper je u svom radu [29] opisao proceduru odlučivanja za PIA, ali ona može biti prilagođena za PNA postavljanjem dodatnih ograničenja za svaku promenljivu: za datu formulu, možemo je najpre univerzalno zatvoriti; nakon toga, za svaki univerzalni kvantifikator, prezapisujemo $(\forall x)f$ u $(\forall x)(x \geq 0 \rightarrow f)$, i za svaki egzistencijalni kvantifikator prezapisujemo $(\exists x)f$ u $(\exists x)(x \geq 0 \wedge f)$. Na taj način dobijamo PIA formulu ekvivalentnu datoj PNA formuli.

Pored toga, Cooperov algoritam može biti redizajniran tako da predstavlja specijalizovanu proceduru odlučivanja za PNA. Takav algoritam je nešto složeniji, ali radi samo sa prirodnim brojevima (i generiše dokaz na objektnom jeziku teorije PNA, što je često veoma pogodno u dokazivačima teorema). U Cooperovom algoritmu (opisanom u poglavlju D.2); potrebno je zameniti korake (4d), (4e) i (4f) na sledeći način (pri čemu su svi brojevi koji se pojavljuju u opisu algoritma prirodni, a ne celi):

4d. Za sve termove t, t_1, t_2 u atomičkim formulama $t_1 \leq t_2$, $i|t$, $i \nmid t$ upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, upotrebi *left-assoc* za $\cdot$, upotrebi *left-assoc* za $+$, upotrebi *poly-form*, upotrebi *reorder* za promenljivu x, i onda upotrebi *collect* za pormenljivu x. Nakon toga, svaka atomička formula ili ne sadrži x ili je jednog od sledećih oblika:

$\alpha'_i + \lambda'_i x \leq \alpha''_i + \lambda''_i x$

$\delta_i | \gamma_i + \nu_i x$

$\eta_i \nmid \varepsilon_i + \xi_i x$

gde su λ'_i, λ''_i, ν_i, δ_i, ξ_i i η_i pozitivni prirodni brojevi a α'_i, α''_i, γ_i i ε_i su termovi koji ne sadrže x.

Ako je $\lambda'_i > \lambda''_i$, onda prezapiši $\alpha'_i + \lambda'_i x \leq \alpha''_i + \lambda''_i x$ u $\alpha'_i + \lambda_i x \leq \alpha''_i$, gde je $\lambda_i = \lambda'_i - \lambda''_i$. Ako je $\lambda'_i < \lambda''_i$, onda prezapiši $\alpha'_i + \lambda'_i x \leq \alpha''_i + \lambda''_i x$ u $\alpha'_i \leq \alpha''_i + \lambda_i x$, gde je $\lambda_i = \lambda''_i - \lambda'_i$. Ako je $\lambda'_i = \lambda''_i$, onda prezapiši $\alpha'_i + \lambda'_i x \leq \alpha''_i + \lambda''_i x$ u $\alpha'_i \leq \alpha''_i$. Nakon toga, svaka atomička formula ili ne sadrži x ili je jednog od sledećih oblika:

$\alpha'_i + \lambda_i x \leq \alpha''_i$

$\beta'_i \leq \beta''_i + \mu_i x$

$\delta_i | \gamma_i + \nu_i x$

$\eta_i \nmid \varepsilon_i + \xi_i x$

gde su λ_i, μ'_i, ν_i, δ_i, ξ_i i η_i pozitivni prirodni brojevi a α'_i, α''_i, β'_i, β''_i, γ_i i ε_i su termovi koji ne sadrže x.

4e. Neka je δ NZD svih vrednosti λ_i, μ_i, ν_i i ξ_i. Prezapiši sve atomičke formule $\alpha'_i + \lambda_i x \leq \alpha''_i$ u $\phi_i \alpha'_i + \delta x \leq \phi_i \alpha''_i$ gde je $\phi_i = \delta / \lambda_i$ (primetimo da je ϕ_i (pozitivan) prirodan broj). Prezapiši sve atomičke formule $\beta'_i \leq \beta''_i + \mu_i x$ u $\phi_i \beta'_i \leq \phi_i \beta''_i + \delta x$ gde je $\phi_i = \delta / \mu_i$ (ϕ_i je (pozitivan) prirodan

broj). Nakon toga (kao u koraku (4e) Cooperovog algoritma), prezapiši sve atomičke formule $\delta_i|\gamma_i+\nu_i x$ u $\psi_i|\phi_i\gamma_i+\delta x$ gde je $\phi_i = \delta/\nu_i$ i $\psi_i = \delta_i\phi_i$ (ϕ_i i ψ_i su pozitivni prirodni brojevi); prezapiši sve atomičke formule $\eta_i \not| \varepsilon_i + \xi_i x$ u $\psi_i \not| \phi_i\varepsilon_i + \delta x$ gde je $\phi_i = \delta/\xi_i$ i $\psi_i = \eta_i\phi_i$ (ϕ_i i ψ_i su pozitivni prirodni brojevi). Nakon toga, zameni formulu $(\exists x)f(\delta x)$ formulom $(\exists x)(f(x) \wedge \delta|x)$, u kojoj su svi koeficijenti uz promenljivu x jednaki 1, tj. u novoj formuli svaka atomička formula ili ne sadrži x ili je jednog od sledećih oblika:

$$\begin{array}{rl} (i) & a'_i + x \leq a''_i \\ (ii) & b'_i \leq b''_i + x \\ (iii) & \delta_i|c_i + x \\ (iv) & \eta_i \not| d_i + x, \end{array}$$

gde su a'_i, a''_i, b'_i, b''_i, c_i, d_i izrazi koji ne sadrže x a δ_i i η_i su pozitivni prirodni brojevi.

4f. neka je δ NZD svih vrednosti δ_i i η_i. Zameni formulu $(\exists x)f(x)$ formulom

$$\bigvee_{j=0}^{\delta-1} f(j) \vee \bigvee_{j=0}^{\delta-1} \left(\bigvee_{b'_i} f(b'_i - b''_i + j) \wedge (b''_i \leq b'_i) \right)$$

Korektnost ovog algoritma može biti dokazana slično kao i korektnost osnovnog Cooperovog algoritma za PIA [29].

Dodatno, novi algoritam možemo da proširimo tako da u koraku (4f) koristimo dodatno prezapisivanje kako bi bilo izbegnuto pojavljivanje negativnih celih brojeva. Na primer, ako postoji atomička formula oblika $a'_i + x \leq a''_i$ u formuli $f(x)$, onda će u $f(b'_i - b''_i + j)$ ova formula dobiti oblika $a'_i + b'_i + j \leq a''_i + b''_i$; ako postoji atomička formula oblika $\delta_i|c_i + x$ u formuli $f(x)$, onda će ona u $f(b'_i - b''_i + j)$ imati oblik $\delta_i|c_i + b'_i + (\delta_i - 1)b''_i + j$. Atomičke formule tipova (ii) i (iv) mogu biti obrađen analogno. Ukoliko je primenjeno ovo dodatno prezapisivanje, tekuća formula u algoritmu će biti PNA formula nakon svakog njegovog primenjenog koraka.

D.5.6 Zajednička struktura

Kao što smo videli, Hodesova i Cooperova procedura mogu da budu opisane tako da imaju veoma sličnu strukturu i konsekventno mogu da dele značajan deo kôda. Ova činjenica može biti iskorišćena i u kreiranjeu nad-algoritma koji bi mogao da koristi obe ove procedure (videti fusnotu D.5.4). Naredna tabela ilustruje zajednički deo strukture ove dve procedure.

Pokazali smo da Hodesova i Cooperova procedura mogu da budu implementirane primenom prezapisivanja. Većina koraka u tim implementacijama može biti opisana na čisto sintaksan način — u terminima normalizacija. Ostali koraci zahtevaju složenije forme prezapisivanja kao što je uslovno prezapisivanje ili prezapisivanje višeg reda. Pokazali smo takođe da ova dva algoritma imaju sličnu strukturu i da mogu da dele veliki deo kôda. Na bazi toga, moguće je

	Hodesov algoritam	Cooperov algoritam
1	upotrebi *remove* za eliminisanje simbola $\rightarrow$ i $\leftrightarrow$	
2	upotrebi *remove* za eliminisanje simbola $>, <, \neq, \geq$	
3	upotrebi *stratify* za pomeranje simbola $\vee$, $\wedge$ i $\neg$ "ispod" kvantifikatora	
4	primeni korake 4a0-4h za tekući unutrašnji kvantifikator (ako postoji); ako je potrebno, zameni $(\forall x)f$ formulom $\neg(\exists x)\neg f$	
4a0		upotrebi *remove* za eliminisanje $=$
4a	upotrebi *stratify* za pomeranje simbola $\neg$ "ispod" simbola $\wedge$ i $\vee$	
4b	upotrebi *thin* za eliminisanje višestrukih simbola $\neg$	
4c	upotrebi *remove* za eliminisanje simbola $\neg$	
4d	upotrebi *stratify* za pomeranje simbola $\cdot$ "ispod" simbola $+$, upotrebi *left-assoc* za $\cdot$ i $+$, *poly-form*, *reorder*, *collect* i *isolate* za promenljivu x	
4e	upotrebi *stratify* za pomeranje simbola $\wedge$ "ispod" simbola $\vee$	zameni $(\exists x)f(\delta x)$ fomrulom $(\exists x)(f(x) \wedge \delta \vert x)$
4f	svaku formulu f_i zameni formulom $\bigwedge_i \lambda_i\gamma \leq \nu\alpha_i \wedge \bigwedge_j \nu\beta_i \leq \mu_i\gamma \wedge \bigwedge_k \nu\gamma_i = \nu_i\gamma$ ili formulom $\bigwedge_{i,j} \lambda_i\beta_j \leq \mu_j\alpha_i$ i eliminiši unutrašnji kvantifikator	zameni $(\exists x)f(x)$ formulom $\bigvee_{j=0}^{\delta-1} f_{-\infty}(j) \vee \bigvee_{j=0}^{\delta-1} \bigvee_{b_i} f(b_i + j)$
4g	upotrebi *stratify* za pomeranje simbola $\cdot$ "ispod" simbola $+$, *left-assoc* za $\cdot$ i $+$, *poly-form*, *reorder collect*, *isolate* za konstantu 1.	
4h	upotrebi *reduce* za smanjivanje broja baznih atomičkih formula (na najviše jedan)	
5	upotrebi *stratify* za pomeranje simbola $\neg$ "ispod" simbola $\wedge$ i $\vee$	
6	upotrebi *thin* za eliminisanje višestrukih simbola $\neg$;	
7	upotrebi *remove* za eliminisanje simbola $\neg$	
8	upotrebi *stratify* za pomeranje simbola $\cdot$ ispod simbola $+$, *left-assoc* za $\cdot$ i $+$, *poly-form*, *reorder collect*, *isolate* za konstantu 1.	
9	upotrebi *reduce* za smanjivanje broja baznih atomičkih formula (na najviše jedan)	
10	Ako je ulazna formula prezapisana u $0 \leq 0$, onda je ona valjana (u PIA), a inače nije.	

Tabela D.6: Poređenje strukture Hodesove i Cooperove procedure

kreirati nad–algoritam koji kombinuje eliminaciju qvantifikatora zasnovanu na Hodesovom i Cooperovom algoritmu. Fleksibilna implementacija Hodesove i Cooperove procedure zasnovana na primeni normalizacija iskorištena je u EPM shemi i u opštem okviru za kombinovanje i integrisanje procedura odlučivanja.

D.6 Presburger arithmetic: Summary

In Presburger Natural Arithmetic (PNA), all variables are of the sort $\mathbf{N}$, the set of function symbols is $\Sigma = \{0, s, +\}$, $(0 : \mathbf{N},\ s : \mathbf{N} \rightarrow \mathbf{N},\ + : \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N})$ and the set of the predicate symbols is $\Pi = \{<, >, \leq, \geq\}$ (all the predicate symbols are of the sort $\mathbf{N} \times \mathbf{N}$). The axioms of PNA are those of Peano arithmetic without axioms concerning multiplication. Similarly we introduce Presburger arithmetic over integers — PIA and Presburger arithmetic over rationals — PRA. Theories PNA, PIA and PRA are decidable [97, 72].

Cooper's algorithm [29] is one of decision procedures for PIA and Hodes' algorithm [52] is one of decision procedures for PRA. Hodes' algorithm can also be used as sound (but incomplete) procedure for universally quantified fragment of PIA. It is part of the tradition of automated reasoning that the intractability of Cooper's decision procedure (with worst case complexity $2^{2^{2^n}}$) for Presburger integer arithmetic makes is too expensive for practical use. However, we performed a series of experimental comparisons between Cooper's and Hodes'

algorithms and obtained results suggesting that Cooper's algorithm is, at least, not worse than one due to Hodes' for practical applications.

We show that Hodes' and Cooper's procedures could be implemented using rewriting. Most of steps in them can be described in a pure syntactical way — in terms of normalizations, in the spirit of Bundy's idea of proof-plans for normalization [22]. We also show that these two algorithms have a rather similar structure and, therefore, can share a lot of the code.

Literatura

[1] A. Armando, L.Compagna, and S. Ranise. System Description: RDL Rewrite and Decision Procedure Laboratory. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, IJCAR 2001.*, volume 2083 of *LNCS*, pages 663–669. Springer-Verlag, 2001.

[2] A. Armando and S. Ranise. Constraint Contextual Rewriting. In *Proceedings of the International Workshop on First order Theorem Proving (FTP'98)*, pages 65–75, Vienna, Austria, November, 23-25 1998.

[3] A. Armando and S. Ranise. A Practical Extension Mechanism for Decision Procedures. In *Proceedings of Formal Methods Tools 2000 (FMT'2000)*, 2000.

[4] A. Armando and S. Ranise. Termination of Constraint Contextual Rewriting. In *Proceedings of 3rd International Workshop on Frontiers of Combining Systems (FroCoS'2000)*, Lecture Notes in Artificial Intelligence 1794, pages 47–61, Nancy, France, March 2000.

[5] L. Bachmair. *Proof Methods for Equational Theories.* PhD thesis, University of Illinois, 1987.

[6] L. Bachmair and H. Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In *Proceedings of the 10th Conference on Automated Deduction*, number 449 in Lecture Notes in Computer Science, pages 427—441. Springer, 1990.

[7] L. Bachmair and H. Ganzinger. Strict Basic Superposition. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in Lecture Notes in Artificial Intelligence. Springer, 1998.

[8] L. Bachmair and A. Tiwari. Abstract Congruence Closure and Specializations. In David A. MacAllester, editor, *Proceedings of the 17th Conference on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Artificial Intelligence. Springer, 2000.

[9] C. Barrett, D. Dill, and J. Levitt. Validity Checking for Combinations of Theories with Equality. In *International Conference on Formal Methods in Computer-Aided Design*, number 1166 in LNCS, pages 187–201. Springer, 1996.

[10] Clark W. Barrett, David L. Dill, and Aaron Stump. A Framework for Cooperating Decision Procedures. In David A. MacAllester, editor, *Proceedings of the 17th Conference on Automated Deduction (CADE–17)*, number 1831 in Lecture Notes in Artificial Intelligence. Springer, 2000.

[11] N. S. Bjørner. *Integrating decision procedures for temporal verification*. PhD thesis, Stanford University, 1998.

[12] E. Black, F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of DARPA Speech and Natural Language Workshop*, February 1992.

[13] W. W. Bledsoe. A new method for proving certain Presburger formulas. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, U.S.S.R., 1975.

[14] A. Bouhoula and M. Rusinowitch. Automated Case Analysis in Proof by Induction. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)*, San Mateo, CA, 1993. Morgan Kaufmann.

[15] R. S. Boyer and J S. Moore. *A Computational Logic*. Academic Press, 1979. ACM monograph series.

[16] R. S. Boyer and J S. Moore. *A Computational Logic Handbook*. Academic Press, 1988. Perspectives in Computing, Vol 23.

[17] R. S. Boyer and J S. Moore. Integrating Decision Procedures into Heuristic Theorem Provers: A Case Study of Linear Arithmetic. In J. E. Hayes, J. Richards, and D. Michie, editors, *Machine Intelligence 11*, pages 83–124, 1988.

[18] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, editor, *Proceedings of the 10th Conference on Automated Deduction*, number 449 in Lecture Notes in Artificial Intelligence, pages 647–648. Springer-Verlag, 1990. Also available from Edinburgh as DAI Research Paper 507.

[19] Alan Bundy. *The Computer Modelling of Mathematical Reasoning*. Academic Press, 1983.

[20] Alan Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[21] Alan Bundy. A science of reasoning. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–198. MIT Press, 1991. Also available from Edinburgh as DAI Research Paper 445.

[22] Alan Bundy. The Use of Proof Plans for Normalization. In R. S. Boyer, editor, *Essays in Honor of Woody Bledsoe*, pages 149–166. Kluwer, 1991. Also available from Edinburgh as DAI Research Paper No. 513.

[23] R. Busatto. *The use of proof planning in normalisation.* PhD thesis, University of Edinburgh, 1995.

[24] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–340. Morgan Kaufmann, 1991.

[25] Michael A. Colon and Tomas E. Uribe. Generating finite–state abstractions of reactive systems using decision procedures. In *International Conference on Computer–Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 293—304. Springer, 1998.

[26] R. L. Constable, S. F. Allen, H. M. Bromley, et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[27] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM Press, 1971.

[28] D. C. Cooper. Programs for Mechanical Program Verification. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, pages 43–59. Edinburgh University Press, 1971.

[29] D. C. Cooper. Theorem Proving in Arithmetic Without Multiplication. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, pages 91–99. Elsevier, New York, 1972.

[30] D. Craigen, S. Kromodimoeljo, I. Meisels, B. Pase, and M. Saaltink. EVES: An Overview. In *Proceedings of Formal Software Development Methods (VDM '91)*, number 552 in LNCS, pages 389–405. Springer, 1991.

[31] N. Cutland. *Computability: an introduction to recursive function theory.* Cambridge University Press, 1980.

[32] D. Cyrluk, P. Lincoln, and N. Shankar. On Shostak's Decision Procedure for Combinations of Theories. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of the 13th Conference on Automated Deduction*, number 1104 in Lecture Notes in Artificial Intelligence. Springer, 1996.

[33] Satyaki Das, David L. Dill, and Seungjoon Park. Experience with predicate abstraction. In *11th International Conference on Computer–Aided Verification*, pages 160—172. Springer, 1999.

[34] M. Dauchet. Termination of rewriting is undecidable in the one rule case. In *Proceedings of the 4th Int. Conf. on Rewriting Techniques and Applicationb*, volume 324 of *Lecture Notes in Computer Science*, pages 262–268. Springer–Verlag, 1988.

[35] M. Dauchet. Simulation of turing machines by a left-linear rewrite rule. *Theoretical Computer Science*, 103:409–120, 1992.

[36] M. Davis. A computer program for Presburger's algorithm. In A. Robinson, editor, *Proving Theorems, (as Done by Man, Logician, or Machine)*, pages 215–233. Cornell Unirvesity, Ithaca, New York, 1957.

[37] N. Dershowitz. Ordering for term–rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[38] N. Dershowitz. Applications of the Knuth-Bendix Completion Procedure. Technical Report ATR-83(8478)-2, Office of Laboratory Operations, The Aerospace Corporation, 1983.

[39] N. Dershowitz. Termination of rewriting. In J.-P. Jouannaud, editor, *Rewriting Techniques and Applications*, pages 69–116. Academic Press, 1987.

[40] N. Dershowitz and J.-P. Jouannaud. Rewriting systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, pages 243–320. Elsevier, Amsterdam, 1990.

[41] Dave Detlefs. An overview of the extended static checking system. In *Proceedings of the First Workshop on Formal Methods in Software Practice*, pages 1–9. ACM (SIGSOFT), 1996.

[42] David L. Detlefs, K. Rustan, M. Leino, Greg Nelson, and James B. Saxe. Extended static checking. Technical Report 159, Compaq SRC, 1998.

[43] Ehdm. User Guide for the EHDM Specification Language and Verification System, Version 6.1. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA., 1993.

[44] Ершов, Ю.Л., Лавров, И.А., Тайманов, А.Д., and Тайцлин, М.А. Елементарные теории. Успехи математических наук, XX(4(124)):37–108, 1965.

[45] M. Fisher and M Rabin. Super -exponential complexity of Presburger arithmetic. Technical report, Massachusetts Institute of Technology, May 1972. MAC Technical Memorandum 33.

[46] E. Friedgut. Sharp threshold for graph properties and the k-sat problem. *Journal of the American Mathematical Society*, 12:1017–1054, 1999.

[47] B. M. Gabbay, C. J. Hogger, and J. A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1. Oxford University Press, Oxford, 1993.

[48] Ian P. Gent and Toby Walsh. The SAT phase transition. In *Proceedings of ECAI-94*, pages 105–109, 1994.

[49] M. J. Gordon, A. J. Milner, and C. P. Wadsworth. *Edinburgh LCF - A mechanised logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer Verlag, 1979.

[50] E. Grädel. Subclasses of Presburger Arithmetic and the Polynomial-Time Hierarchy. *Theoretical Computer Science*, 56:289–301, 1988.

[51] D. Hilbert and P. Bernays. *Grundlagen der Mathematik (Zweite Auflage)*. Springer–Verlag, 1968. Izdanje na ruskom jeziku: Основания Математики, Москва, "Наука", 1979.

[52] Louis Hodes. Solving Problems by Formula Manipulation in Logic and Linear Inequalities. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, Imperial College, London, England, 1971.

[53] D. Hofbauer. Termination proofs by multiset path ordering imply primitive recursive derivation lengths. In *Proceeding sof the Second Conference on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 347–358. Springer, Berlin, 1990.

[54] G Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[55] G Huet. A complete proof of correctness of the Knuth–Bendix completion algorithm. *Journal of the Computer and System Sciences*, 23:11–21, 1981.

[56] G. Huet and D. C. Oppen. Equations and rewrite rules: A survey. In R. Book, editor, *Formal languages: Perspectives and open problems*. Academic Press, 1980.

[57] Predrag Janičić. GD-SAT model and crossover line. Journal of Experimental and Theoretical Artificial Intelligence, 13(3):181–198, 2001.

[58] Predrag Janičić and Alan Bundy. A General Setting for the Flexible Combining and Augmenting Decision Procedures. *Journal of Automated Reasoning*, 28(3):257–305, 2002.

[59] Predrag Janičić, Alan Bundy, and Ian Green. A framework for the flexible integration of a class of decision procedures into theorem provers. In Harald Ganzinger, editor, *Proceedings of the 16th Conference on Automated Deduction (CADE-16)*, number 1632 in Lecture Notes in Artificial Intelligence Series, pages 127–141. Springer, 1999.

[60] Predrag Janičić, Ian Green, and Alan Bundy. A comparison of decision procedures in Presburger arithmetic. In Ratko Tošić and Zoran Budimac, editors, *Proceedings of the VIII Conference on Logic and Computer Science (LIRA '97)*, pages 91–101, Novi Sad, Yugoslavia, September 1–4 1997. University of Novi Sad. Also available from Edinburgh as DAI Research Paper No. 872.

[61] Matthias Jantzen. Basics of term rewriting. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer–Verlag, 1997.

[62] F. Jelinek, Lafferty J. D., and R. L. Mercer. Basic methods of probabilistic context free grammars. Technical report, 1990. Technical Report RC 16374 (72684), IBM, Yorktown Heights, NY.

[63] A.K. Joshi and Y. Schabes. Tree-adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*. Elsevier Science, 1992.

[64] D. Kapur, P. Narendran, and F. Otto. On ground confluence of term rewriting systems. *Information and Computation*, 86:14–31, 1990.

[65] D. Kapur and M Subramaniam. Lemma discovery in automating induction. In M. A. McRobbie and J. K. Slaney, editors, *13th International Conference on Automated Deduction (CADE-13)*, pages 538–552. Springer, 1996.

[66] D. Kapur and H. Zhang. An overview of rewrite rule laboratory (RRL). *J. of Computer Mathematics with Applications*, 29(2):91–114, 1995.

[67] Deepak Kapur. Shostak's Congruence Closure as Completion. In *International Conference on Rewriting Techniques and Applications, RTA '97*, Barcelona, Spain, June 1997.

[68] Deepak Kapur and Xumin Nie. Reasoning about Numbers in Tecton. In *Proceedings of 8th International Symposium on Methodologies for Intelligent Systems, (ISMIS'94)*, pages 57–70, Charlotte, NC, October 1994.

[69] Deepak Kapur and M. Subramaniam. Using an induction prover for verifying arithmetic circuits. *Software Tools for Technology Transfer*, 3(1):32–65, 2000.

[70] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational problems in abstract algebra*, pages 263–297. Pergamon Press, 1970.

[71] Peter Krauss. Quantifier elimination. In G. et al. Muller, editor, *Logic Conference, Kiel 1974*. Springer–Verlag, Berlin, 1975.

[72] Georg Kreisel and Jean Louis Krivine. *Elements of Mathematical Logic: Model Theory*. North Holland, Amsterdam, 1967.

[73] Brigitte Krenn and Christer Samuelson. *The Linguist's Guide to Statistics*. 1997. on line: http://www.coli.uni-sb.de/~krenn/stat_nlp.ps.

[74] J.-L. Lassez and M.J. Maher. On Fourier's algorithm for linear arithmetic constraints. *Journal of Automated Reasoning*, 9:373–379, 1992.

[75] D. C. Luckham, S. M. German, F. W. Von Henke, R. A. Karp, P. W. Milne, D. C. Oppen, W. Polak, and W. L. Scherlis. Stanford Pascal Verifier user manual. Technical report, 1979. CSD Report STAN-CS-79-731, Stanford University, Stanford, CA.

[76] Z. Manna. *STeP: The Stanford Temporal Prover*. Technical report, 1994. STAN-CS-TR-94-1518, Computer Science Department, Stanford University, Stanford, CA.

[77] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proc. of 4th Int. Conf. on System Science*, pages 789–792, Honolulu, Hawaii, 1970.

[78] A. Manning, A. Ireland, and A. Bundy. Increasing the Versatility of Heuristic Based Theorem Provers. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 93, St. Petersburg*, number 698 in Lecture Notes in Artificial Intelligence, pages 194–204. Springer-Verlag, 1993.

[79] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Naples, 1984. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980.

[80] A. Middeldorp and B. Gramlich. Simple termination is difficult. *Applicable Algebra in Engineering, Communication and Computing*, 6:115–128, 1995.

[81] Žarko Mijajlović. Odlučive teorije. *Računarstvo*, 1(1):3–21, 1991.

[82] Žarko Mijajlović, Zoran Marković, and Kosta Došen. *Hilbertovi problemi i logika*. Zavod za udžbenike i nastavna sredstva, Beograd, 1986.

[83] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Phase transition and search cost in the $2 + p$-sat problem. In Proceedings of the Fourth Workshop on Physics and Computation, pages 229–232. Boston University, 1996.

[84] A. Mostowski. On direct products of theories. *Journal of Simbolic Logic*, 17:1–31, 1952.

[85] Bernhard Nebel. Artificial Intelligence: A Computational Perspective. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 237–266. CSLI Publications, 1996.

[86] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, October 1979.

[87] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, April 1980. Also: Stanford CS Report STAN-CS-77-646, 1977.

[88] Greg Nelson. Combining satisfiability procedures by equality–sharing. *Contemporary Mathematics*, 29:201–211, 1984.

[89] K. Nordström, B. Petersson and J. Smith. *Programming in Martin-Löf Type Theory*. Oxford University Press, 1990.

[90] Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

[91] Derek C. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12, 1980.

[92] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. K. Srivas. PVS: Combining Specification, Proof Checking, and Model Checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 1996 Conference on Computer-Aided Verification*, number 1102 in LNCS, pages 411–414, New Brunswick, NJ, 1996. Springer-Verlag.

[93] David Y.W. Park, Skakebaek U., Mats P.E. Heimdahl, Barbara J. Czerny, and David Dill. Checking properties of safety critical specifications using efficient decision procedures. In *Second Workshop on Formal Methods in Software Practice*, 1998.

[94] D. Plaisted. A recursivelt defined ordering for proving termination of term rewriting systems. Technical report, 1978. Technique Report R-78-943, University of Illinois at Urbana-Champaign, Urbana, IL.

[95] D. Plaisted and A. Sattler-Klein. Proof lengths for equational completion. *Information and Computation*, 125:154–170, 1996.

[96] David A. Plaisted. Equational reasoning and term rewriting systems. In B. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1. Oxford University Press, Oxford, 1993.

[97] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu metematyków slowiańskich, Warszawa 1929*, pages 92–101, 395. Warsaw, 1930. Annotated English version also available [112].

[98] O. Michael Rabin. Decidable theories. In Barwise Jon, editor, *Handbook of Mathematical Logic*, pages 595–629. North–Holland Publishing Company, 1977.

[99] J. A. Robinson. A machine oriented logic based on the resolution principle. *J Assoc. Comput. Mach.*, 12:23–41, 1965.

[100] Harald Rueß and Natarajan Shankar. Deconstructing Shostak. In *Proceedings of the Conference on Logic in Computer Science (LICS)*, 2001.

[101] H. Saidi and N. Shankar. Abstract and model check while you prove. In *11th International Conference on Computer–Aided Verification*. Springer, 1999.

[102] Y. Schabes and R.C. Waters. Stochastic lexicalized context-free grammar. Technical Report 93-12, Mitsubishi Electric Research Laboratories, Cambridge Research Center, 1993.

[103] Joseph R. Shoenfield. *Mathematical logic*. Addison–Wesley, Reading, MA, 1967.

[104] R. E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, January 1984. Also: *Proceedings of the 6th International Conference on Automated Deduction*, volume 138 of *Lecture Notes in Computer Science*, pp. 209–222. Springer-Verlag, June 1982.

[105] Robert E. Shostak. On the SUP-INF method for proving Presburger formulas. *JACM*, 24(4):529–543, October 1977.

[106] Robert E. Shostak. A practical decision procedure for arithmetic with function symbols. *JACM*, 26(2):351–360, April 1979.

[107] T. Skolem. Über einige Satzfunktionen in der Arithmetik. In J. E. Fenstad, editor, *Selected Works in Logic (by Th. Skolem)*. Universitets–forlaget, Oslo, 1970.

[108] W. Snyder. A fast algorithm for generating reduced ground rewrting system from a set of ground equations. *Journal of Symbolic Computation*, 15:415–450, 1993.

[109] R. Socher-Ambosius. Boolean algebra admits no convergent rewriting system. In *Proceedings of the 4th International Conference on rewriting techniques and applications*, volume 488 of *LNCS*. Springer, Berlin, 1991.

[110] Željko Sokolović. Odlučivost matematičkih teorija. magistarski rad, Matematički fakultet, Beograd, 1987.

[111] Irena Spasić and Predrag Janičić. *Teorija algoritama, jezika i automata – zbirka zadataka*. Matematički fakultet, Beograd, 1999.

[112] Ryan Stansifer. Presburger's Article on Integer Arithmetic: Remarks and Translation. Technical Report TR 84-639, Department of Computer Science, Cornell University, September 1984.

[113] Larry Stockmeyer. Classifying the computational complexity of problems. *The Journal of Symbolic Logic*, 52(1):1–44, March 1987.

[114] J. Su, D. Dill, and J. Shakkebaek. Formally verifying data and control with weak reachability invariants. In *Formal Methods in Computer-Aided Design*, 1998.

[115] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975.

[116] A. Tarski, A. Mostowski, and Robinson R. M. *Undecidable Theories*. North Holland, 1953.

[117] Axel Thue. Die Lösung eines Spezialfalls eines generellen logischen Problems. *Videnskabs-Selskabets Skrifter. I. Math.-Naturv. Klasse*, 8, 1910.

[118] Cesare Tinelli. Extending the CLP scheme to unions of constraint theories. Master's thesis, Department of Computer Science, University of Illionis at Urbana–Champaign, 1995.

[119] Cesare Tinelli. *Combining Satisfiability procedures for Automated Deduction and Constraint–Based Reasoning*. PhD thesis, Department of Computer Science, University of Illionis at Urbana–Champaign, 1999.

[120] Cesare Tinelli and Mehdi Harandi. A New Correctness Proof of the Nelson–Oppen Combination Procedure. In Franz Baader and Klaus U. Schultz, editors, *Frontiers of Combining Systems: Proceeding of the 1st International Workshop*, pages 103–120. Kluwer, 1996.

[121] Cesare Tinelli and Christophe Ringeissen. Non–Disjoint Unions of Theories and Combinations of Satisfiability Procedures. *Submitted for publication*, 1999.

[122] H. Zhang and D. Kapur. First-order theorem proving using conditional rewrite rules. In E. Lusk and R. Overbeek, editors, *Proceedings of 9th Conference on Automated Deduction*, number 310 in Lecture Notes in Computer Science, pages 1–20. Springer, 1985.

[123] H. Zhang and J. L. Rémy. Contextual rewriting. In J. P. Jouannaud, editor, *Proceedings of 1st International Conference on Rewriting Techniques and Applications*, number 202 in Lecture Notes in Artificial Intelligence Series, pages 1–20. Springer, 1985.

[124] Hantao Zhang. Contextual rewriting in automated reasoning. *Fundamenta Informaticae 24*, 24:107–123, 1995.