Institut za matematiku
Prirodno-matematicki fakultet
Univerzitet u Beogradu

Irena Pevac

DOKAZIVANJE TEOREMA PRIRODNIM IZVODJENJEM UZ POMOC RACUNARA

doktorska disertacija

Beograd, juni 1987.

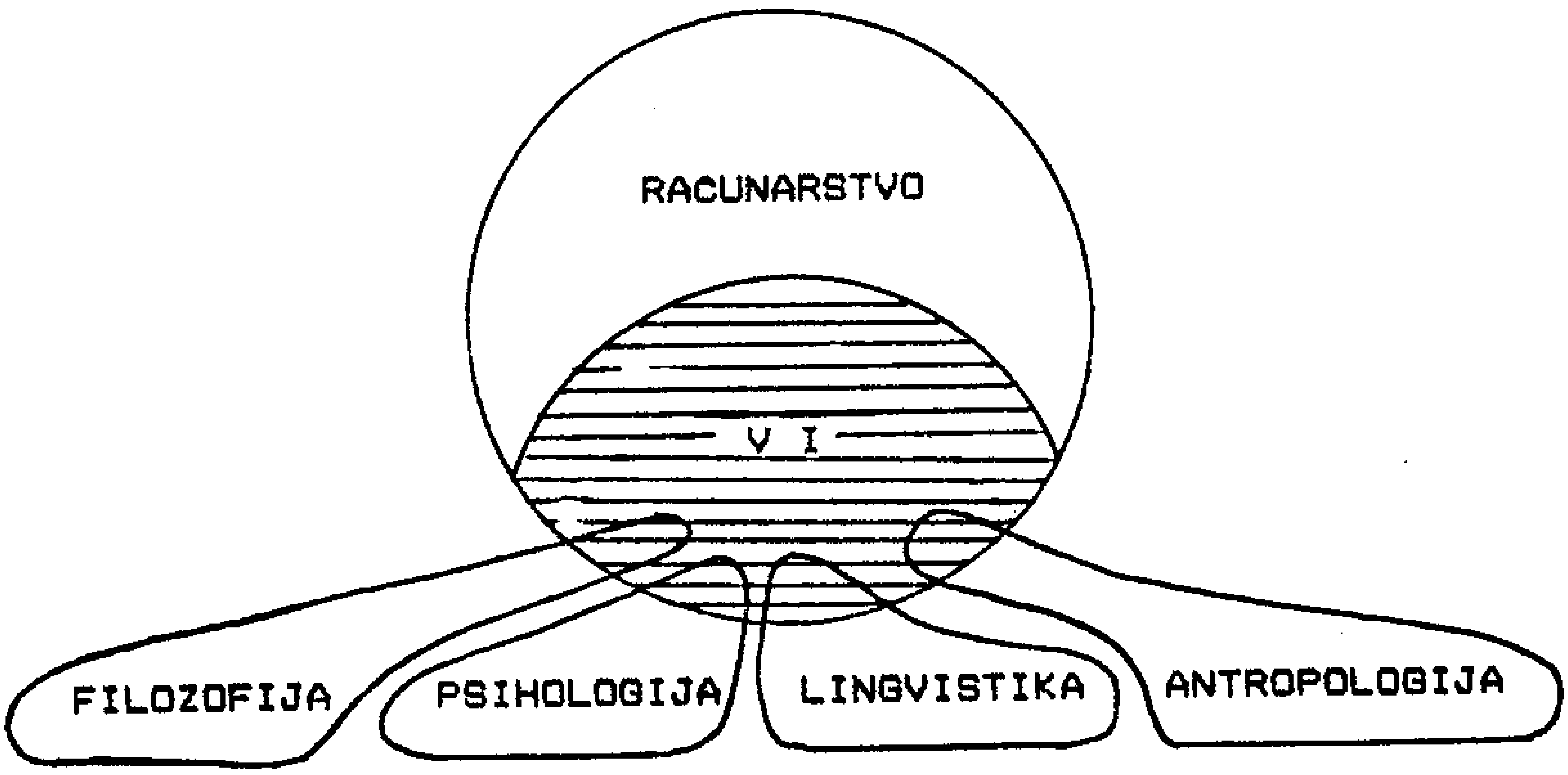
S A D R Ž A J

1. UVOD

Veštačka inteligencija (u daljem tekstu VI) istražuje simboličko rezonovanje, algoritamski nerešive probleme, predstavljanje znanja i izvodjenja deduktivnog i induktivnog tipa na računaru.

Cilj radova u VI je stvaranje takvih računara i programa za njih, kojima se izvode aktivnosti za koje je potrebna inteligencija čoveka.

Veštačka inteligencija nije autonomna disciplina već je deo računarstva, ali i druge discipline kao psihologija, filozofija, lingvistika i antropologija dele interesovanje za probleme koji u svom rešavanju podrazumevaju korišćenje inteligencije. Shematski položaj VI je negde na preseku kao što prikazuje slika 1.



Slika 1.

Osnovni stavovi od kojih se polazi su da se inteligencija može objasniti kao aktivnost manipulisanja simbola, da se to može realizovati na fizičkom sistemu, specijalno digitalnom računaru kao univerzalnoj napravi za manipulisanje simbolima, da se razni aspekti ljudske inteligencije mogu modelovati pomoću računara i najzad daleki cilj kojem težimo bilo bi otkriće teorije inteligencije koja bi bila dovoljno opšta da obuhvati fenomene kako ljudske tako i mašinske inteligencije.

VI je relativno mlada naučna disciplina. Počeci njenog razvoja datiraju od 1930. godine da bi svoj pravi procvat doživela tek pojavom računara četvrte i pete genaracije. U okviru VI razvile su se sledeće značajnije oblasti istraživanja: dokazivanje teorema, mašinsko učenje, prepoznavanje oblika i govora, rešavanja problema, specijalni programski jezici VI i predstavljanje znanja.

Srž VI je nalaženje tehnika u problemskim prostorima gde postoji kombinatorična eksplozija za generisanje podciljeva sa apriori većom verovatnoćom da vode ka uspešnom kraju, umesto da koristimo metod kompletne pretrage.

U centru istraživanja VI je priroda uma što je jedna od najvećih svetskih misterija. Sa raznih aspekata pokušava se otkriti koherentno činjenično stanje karaktera naše inteligencije i znanja.

U daljem razvoju nauka smatra se da će i same nauke doći do metapozicije u kojoj će bavljenje naukom (posmatranje, eksperiment, teorija, testiranje...) zahtevati razumevanje tih informacionih procesa. Time će se približiti i nestati granice izmedju nauke kao celine (prikupljanje i organizovanje znanja o svetu) i VI (razumevanje kako se znanje prikuplja i organizuje).

Kaže se da nema prave koherentnosti u VI, već je polje sastavljeno od raznih kolekcija ideja kako raditi sa aplikacijama nenumeričkog izračunavanja omogućavajući da mašine rade stvari koje zahtevaju nivo ljudskih kognitivnih sposobnosti i inteligencije.

U daljem tekstu citiraćemo mišljenja najistaknutijih naučnika u oblasti VI o položaju, značaju i koherentnosti VI prema <12>.

Margaret Boden:"VI pati od nekih problema kao i filozofija : ona dodiruje neodgovoreno ili čak odgovara na pitanja za koja se zna da ne postoji odgovor. Čim se nadje adekvatan način da se odgovori na neko pitanje on postaje problem za specijaliste podpolja. Razne nauke razvijale su se od renesanse, dok su se razne discipline izrasle iz VI razvijale svega 30 godina. Ipak imamo različita podpolja kao što su sistemi zasnovani na pravilima, prepoznavanje oblika, procesiranje slike".

Nils Nilsson:"Deo VI zasnovan na rezonovanju sa deklarativno predstavljenim znanjem (logičkim formulama) sa primesama semantike u specijalizovanim procedurama i strukturama podataka je koherentno polje. To će se razvijati kao glavna disciplina neophodna u inteligentnim sistemima. Postoji jezgro VI/kognitivna nauka što podrazumeva sistematsko studiranje aktualnih i mogućih inteligentnih sistema."

Pamela Mc Corduck:"VI će verovatno sadržati najveći deo nauke o računarstvu. Novo polje će biti čvrsto zasnovano na nauci o simbolima ili informaciji i biće jasno i uredjeno."

Alan Newell:"Jedna od najvećih svetskih misterija priroda uma je u centru VI. Njegovo otkriće praćeno otkrićem prirode života i porekla univerzuma bit će glavno poglavlje náučnog napredka čovečanstva. Kada se to dogodi postojaće koherentno činjenično stanje prirode inteligencije, znanja, namera, želja kao i odgovor na pitanje kako je moguće da se ti fenomeni pojave u našem fizičkom svetu."

Saul Amarel:"Smatram da će u sledećoj dekadi najveći napredak doživeti mašinsko učenje, problemi projektovanja i planiranja, rezonovanje vodjeno raznim planovima, kvalitativni i kvantitativni modeli i tehnologije za gradnju i usavršavanje ekspertnih sistema sa bazama znanja sa 10K pravila."

1.1 O HEURISTICKOM PRISTUPU U VEŠTACKOJ INTELIGENCIJI

Heuristički pristup je široko primenjivan u raznim oblastima VI od mašinskog ucenja i gradnje ekspertnih sistema, posebno onih zasnovanih na pravilima i sa domenskim prostorima koji zahtevaju verovatnosno rezonovanje, preko teorije igara, rešavanja problema pa sve do dokazivanja teorema.

Impresivan rad ekspertnih sistema zavisi od velikog korpusa znanja u kome postoji znatan broj neformalnih pravila (heuristika) koje sistem vode ka verovatnim putevima s jedne strane i izbegavaju puteve za koje postoji verovatnoca da nisu dobri s druge strane.

U novije vreme razvila se cela nova oblast **heuretika** koja se bavi izučavanjem polja heuristika: prirode heuristika, njihove medjuzavisnosti i organizacije korpusa heuristika, te najzad, generisanja novih heuristika.

Navedimo samo neke od hipoteza o heuretici prema <51>:

1) Heuretika je izvorno polje znanja i iziskuje dalja istraživanja u VI. Što se tiče metoda istraživanja objekata smatra se da će empirijska paradigma dominantna u istraživanjima u VI i tu doprineti novim saznanjima. To znači da treba testirati hipoteze o heuristikama konstruisanjem i studiranjem računarskih programa koji koriste heuristike i pokušavaju da generišu nove.

2) Heuristička pravila imaju tri osnovne namene: izbegavanje koraka, akcija odnosno alternativa koje nisu verovatne, ili nisu poželjne, ili nisu dobre, zatim nalaženje koraka, akcija odnosno alternativa koje su povoljne, verovatne ili vode ka uspehu i najzad služe kao podaci za indukovanje novih heurističkih pravila. U poslednjem slučaju, dakle, rezultat primene heuristika više nije novi domenski koncept već nova heuristika.

Ilustrujmo kako heuristika može voditi istraživanje bilo da ga vodi čovek ili računar, ka novim konceptima koji su interesantni odnosno značajni.

Posmatrajmo heuristiku: Za datu binarnu funkciju f(x,y) definiraj i študiraj funkciju g(x)=f(x,x).

Rezultat je predstavljen sledecom tabelom:

f(x,y)		g(x)	
množenje	x * y	kvadriranje	x^2
sabiranje	x + y	dupliranje	2x
unija	x ∪ y	identitet	x ≡ x
presek	x ∩ y	identitet	x ≡ x
oduzimanje	x − y	identično nuli	0
ubijanje	ub(x,y)	samoubistvo	ub(x,x)
dati gol	go(x,y)	autogol	go(x,x)
uslužiti	us(x,y)	samoposluživanje	us(x,x)

3) Heuristike crpu snagu na osnovu raznih vrsta regularnosti i kontinuiteta u svetu.

Ako je neka akcija A bila ranije ili bi bila korisna u situaciji S onda možemo očekivati da ce slična akcija kao što je A biti korisna u buducnosti u situacijama koje su slične sa S. Drugim rečima ako bismo nekako aktualno izračunali korisnost heuristike, onda bi ta funkcija:

prikladnost(akcija,situacija)

bila neprekidna po obe promenljive.

4) Heuretika proučava i prostor svih heuristika.

Lenat u svom članku <51> pokušava da u prostor svih heuristika iz raznih područja tj. domena (teorija skupova, teorija brojeva, programiranje u LISP-u, igranje igara, bitke u mornarici itd.) uvede uredjenje sa generalizacijom i specijalizacijom. Na vrhu stoje takozvani slabi metodi (generiši i testiraj, ujednačavanje itd.).

slabi metodi

Ako je f podskup od ...x A x...x B x... i R:A—>B i A ⊂ B
tada definišemo {(...,a,...,R(a),...)∈f}

A=B

R <— jednakost

Ako je f podskup
od ...xAx..xAx... i
R:A—>A tada definišemo
{(...,a,...,R(a),...)∈f}

Ako je f podskup od
...xAx...xBx... i A ⊂ B
tada definišemo
{(...a....a.....)∈f}

R<— jednakost

A=B

Ako je f podskup od ...xAx...xAx...
tada definišemo {(...a...a...)∈f}

..Ax...xAx...<— AxCxAxB

..Ax..xAx.. <— AxCxA

Ako f:AxCxA->B definiši
g(x,y)=f(x,y,x)

C singlton

Ako f:AxA->B
def. g(x)=f(x,x)

Ako F:AxC->A definiši
{(a,c)| f(a,c)=a}

C singlton

Ako f:A->A definišemo
{a| f(a)=a}

ka heuristikama specifičnim za dati domen

Slika 2

Na dnu su milioni vrlo specifičnih heuristika koje se odnose na pojmove specifične za dati domen. Izmedju njih su heuristike tipa onih na slici 2. kao na primer:

- ispitaj ekstremalne slučajeve,
- obrati pažnju na tačke nagomilavanja,
- pogledaj šta se dešava kada se proces ponovi,
- za funkciju f(x,y) ispitaj slučaj kada je x=y.

Strategije za traženje puta u kombinatorično velikim problemskim prostorima za rešavanjve problema primenjuju neki tip vodjenja na osnovu vrednosti funkcije ocene pozicije za dati prostor pretrage. Za funkciju ocene pozicije f domen predstavljaju čvorovi u prostoru a antidomen su realni brojevi koje im prideljuje. Vrednost f(N) za čvor N meri odnosno procenjuje koliko je korisno nastaviti pretragu iz tog čvora. Što je veća vrednost funkcije u čvoru to je povoljnija (odnosno više obećava) primena operatora na taj čvor. Heuristička strategija traženja vodjena funkcijom ocene pozicije uvek traži od čvora kome je vrednost funkcije najveća.

Traženje najpre u širinu (breadth first) i traženje najpre u dubinu (depth first) su trivijalni slučajevi pretrage na osnovu heurističke funkcije. Funkcija sa traženjem najpre u dubinu se konstruiše tako da vrednost čvora predstavlja njegovo rastojanje od početnog čvora. Funkcija sa traženjem najpre u širinu se dobija ako se uzme recipročna vrednost od rastojanja od početnog čvora.

Traženje najpre u dubinu primenjujemo kad dubina nije prevelika, a traženje u širinu kad broj alternativa izbora čvora nije veliki.

Traženje vodjeno heurističkom funkcijom interesantnosti ili dobrote je indikovano postojanjem prirodne mere rastojanja medju čvorovima i dobar put je verovatno medju izglednim parcijalnim putevima na svim nivoima.

Većina strategija pretraga koje se koriste za nalaženje poteza u šahu su primeri tehnika za rešavanje problema. Jedna od najviše primenjivanih ideja za nalaženje dobrog poteza je metod kojim se generišu legalni potezi i odgovori do neke dubine, a zatim izračuna vrednost heurističke funkcije za rezultujuću poziciju. Ovaj metod pretrage je uveo Shannon primetivši da je to oblik igre tipa pokušaja i greške koji ima tendenciju simuliranja ljudskog procesa mišljenja. U <79> on kaže da je proces mišljenja po nekim psiholozima u osnovi karakterisan sledećim koracima: razna moguća rešenja problema se probaju intuitivno ili simbolično bez aktuelnog fizičkog izvodjenja; rezonovanjem tj. nekom mentalnom evaluacijom se bira najbolja varijanta tih pokušaja i zatim za nju traži rešenje.

U skladu sa tim pristupom u šahu se posmatraju sve moguće kombinacije poteza do dubine utvrdjene praktičnim ograničenjima resursa tj. memorijskog prostora i raspoloživog vremena. Zatim se bira potez koji ima maksimalnu vrednost funkcije evaluacije uz pretpostavku da će protivnik takodjer pokušati da maksimizira funkciju evaluacije sa svog stanovišta na svakom koraku.

Prostor pretrage se izuzetno brzo uvećava. Da bi se broj alternativa koliko toliko zadržao unutar prihvatljivih ograničenja resursa, uvode se razne dodatne heuristike za redukovanje drveta pretrage i usmeravanje u povoljnom pravcu.

2. ISTORIJSKI PREGLED RAZVOJA DOKAZIVANJA TEOREMA

Cilj automatskog dokazivanja teorema jeste projektovanje i implementacija računarskih programa koji dokazuju ili pomažu pri dokazivanju teorema.

Prema učešću čoveka u procesu dokazivanja na računaru programe za automatsko dokazivanje teorema delimo na nezavisne od učešća čoveka ili čisto automatske gde program samostalno dokazuje teoremu kada i ako uspe i na interaktivne dokazivače gde programi pronalaze delove dokaza a zatim u interaktivnom izvodjenju dokaza korisnik te delove kombinuje sa sopstvenom intuicijom i eventualno po potrebi preusmerava program u cilju nalaženja dokaza. Čisto automatski tipovi dokazivača predstavljaju dominantnu oblast istraživanja u 50 tim i ranim 60-tim godinama, dok je u skorije vreme s obzirom na izuzetnu kompleksnost problema poraslo interesovanje za interaktivne dokazivače.

U početku je primena programa za dokazivanje teorema bila isključivo u oblasti matematike sve dok se nije uvidelo da se i drugi problemi mogu predstaviti kao moguće teoreme koje treba dokazati. Tako je automatsko dokazivanje teorema (u daljem tekstu ADT) našlo primene u oblastima kao što su korektnost programa, generisanje programa, upitni jezici nad relacionim bazama podataka, projektovanje elektronskih kola itd...

Što se tiče jezika tj. formalne reprezentacije u kojoj se dokazuje teorema ona može biti iskazni račun, predikatski račun prvog reda, kao i logike višeg reda. Teoreme u iskaznom računu su proste za današnje dokazivače, medjutim, iskazni račun nije dovoljno ekspresivan. Predstavljanje njime je suviše prosto da bi se izrazile iole bogatije teorije. Logike višeg reda su, naprotiv, vrlo ekspresivne ali sa njima postoji niz praktičnih problema. Zato je verovatno najviše korišćen predikatski račun prvog reda.

Naravno, u kreiranju dokazivača, posebno interaktivnih, ne polazimo uvek od nivoa kvantifikatorske formule. Kako bismo korisniku omogucili lakše komuniciranje sa sistemom poželjno je da se formule izraze na visokom nivou tj. na formalizovanom podskupu prirodnog jezika, a takodjer i komunikacija čoveka i računara mora biti prilagodjena čoveku. Sam program za dokazivanje teorema radi na nivou formula predikatskog računa. Da bi se to postiglo, u toku rada se formule često transformišu iz jednog oblika u drugi.

Sa gledišta tehnike automatskog dokazivanja teorema najvise izučavanja je posvećeno rezolucijskim pravilima izvodjenja. Primena rezolucije pretpostavlja da se formula iz predikatskog računa prvog reda prevede na oblik sastavaka, čime se znatno gubi na prirodnom obliku formule koji je blizak čoveku.

Drugo pravilo izvodjenja, paramodulacija, je po tehnici slično rezoluciji a obezbedjuje generalisanu jednakosnu substituciju.

Kako je relacija jednakosti vrlo značajna u mnogim matematičkim teorijama, uvodjenje jednakosti kao dela pravila izvodjenja je značajno proširenje i poboljšanje rezolucijskih procedura. Slagle i Norton su čak otišli dalje sa ugradjivnjem i drugih značajnih relacija kao što je parcijalno uredjenje.

Pravila prirodne dedukcije Gentzenovog tipa ne zahtevaju transformisanje formule na oblik sastavaka, niti se zaključak negira. Takva pravila su vrlo bliska onima koje čovek koristi u dokazivanju teorema, pa su naročito pogodna za dokazivače interaktivnog tipa.

Pomenimo još i sisteme sa prirodnim izvodjenjem u kojima ima mnogo pravila, te je najveći problem u kreiranju programa koji kontroliše redosled njihove primene. Izvodjenja su definisana korišćenjem empirijskih metoda koje su rezultat posmatranja i imitacije rada matematičara. Opšta odlika tih sistema je zadržavanje centralne logičke implikacije u formuli za razliku od rezolucije i prirodne dedukcije, kao i razbijanje tekućeg cilja na dva ili više jednostavnijih. U takvim sistemima se koristi niz heuristika kako onih opšte prirode tako i specifičnih za dati domen.

Programi za ADT dokazuju teoreme u raznim formalizovanim teorijama u oblasti matematike kao što su teorija skupova, topologija, geometrija, a primenjuju se i u oblasti korektnosti programa i primenama relacionih baza podataka. Dele se na programe opšte namene koji mogu lako dokazivati teoreme u više različitih oblasti i na one koji su specijalne namene tj. efektivni su samo u specijalnoj oblasti primene.

Prema prilazu programi za ADT se dele na one koji baziraju na studiranju i simuliranju ljudskog mišljenja dok se oni drugi zasnivaju na čisto logičkim osnovama.

Predstavnici prve škole su Newell, Shaw, Simon, Hao Wang, Gelernter, Pastre, Brown i Bledsoe. U drugoj struji pomenimo Herbranda, Robinsona, Wosa, Henchena, Kovalskog, Slaglea, Overbeeka, Carsona, Lovelanda i Luckhama.

Oba pristupa su istraživana još u periodu od 1960 do 1970 godine ali logički pristup je u tom periodu razmatran mnogo više i sa više uspeha nego onaj sa simuliranjem ljudskog mišljenja.

Sa aspekta teorije Herbrand je doveo do velikog preokreta kada je pokazao da za dati predikatski račun prvog reda postoji algoritam kojim se za datu teoremu u konačno mnogo koraka potvrdjuje da ona jeste teorema. Problem je semiodlučiv jer u slučaju kad je na ulazu formula koja nije teorema postupak se nastavlja u beskonačnost.

Ideja je da se dokaz teoreme u predikatskom računu prvog reda izvede kontradikcijom. U tom cilju se negirana formula prevodi u konjunktivnu normalnu formu. Egzistencijalne promenljive se zamene Skolemovim funkcijama i zatim primeni neka tehnika razmišljanja u traženju kontradikcije. Herbrandova tehnika se sastoji u generisanju stalno povećavajućeg skupa formula sa slobodnim promenljivim i zatim se testira svaki skup da li je nezadovoljiv tj. neistinit za svako prideljivanje istinosnih vrednosti Bulovskim promenljivim.

Kompjuterski programi koji su primenjivali tu tehniku su uglavnom razočarali. Otežavajuća okolnost u praksi je bio nagli rast skupova sa slobodnim promenljivim. To je sa velikim brojem primera sa kojima treba testirati nezadovoljivost onemogućavalo generisanje dokaza u ograničenom vremenu i memorijskom prostoru.

Zatim J. Robinson uvodi pojam unifikacije u <76> i primenjuje ga na rezoluciju. Nadalje su razmatrani razni aspekti rezolucije i neke restrikcije njene primene. Rezolucija sa kompletnom pretragom i bez ograničenja se takodjer pokazala neefikasnom jer većina komplikovanijih problema ima za posledicu da se pojavljuje preveliki broj sastavaka u toku izvodjenja dokaza što u praksi znači prekoračenje raspoloživih prostorno vremenskih resursa.

Poboljšanja su bila vezana za istraživanja strategija za način primene pravila izvodjenja i odnosila su se na redosled primene sastavaka i izbegavanje nekih parova sastavaka. Takodjer se pokušalo sa pronalaženjem pravila izvodjenja koja su moćnija napr. nekoliko rezolucijskih koraka u jedan. Najzad se pokušalo u pravcu pronalaženja novih boljih predstavljanja problema i javljaju se sistemi sa prirodnim izvodjenjem.

Prostorno i vremensko ograničenje nametnulo je razmatranje sledećih pitanja:

1) Gde treba sledeće da traži program za ADT,
2) Koji deo prostora pretrage možemo ignorisati,
3) Da li je moguća kanonizacija nove informacije,
4) Kako informaciju osloboditi suvišnog.

U odgovoru na prvo pitanje Wos i ostali <84> su dali ideju kako da menjanjem redosleda mogućih izvodjenja ubrzamo put do dokaza. Program je koncipiran tako da se probaju sve rezolucije u kojima je jedan od sastavaka jedinični pre probanja rezolucije na parovima bez jediničnih sastavaka. Takodjer je preporučeno da se pre koriste kraći od dužih sastavaka. Kovalski i Slagle su sugerisali manje efikasne strategije izbora sastavaka koje se zasnivaju na broju literala, a Overbeek predlaže tehniku za izračunavanje značaja simbola i na osnovu toga biranje sastavka sa kojim će se raditi.

U cilju rešenja drugog pitanja Wos, Robinson i Carson u <85> predlažu strategiju potpornog skupa koja koristi prednosti pretpostavljene neprotivrečnosti aksioma. Ideja je da kontradikcija nije izvodiva iz samog skupa aksioma. Loveland i Luckham uvode linearnu rezoluciju koja onemogućava da se generisani sastavci koriste zajedno kao roditelji.

Rešavanjem trećeg pitanja Wos i ostali su uveli pojam demodulacije i automatizaciju kanonizacije. Pojednostavljenja u <86> se vrše na nivou terma.

Najzad, G. Robinson je razmatrao četvrto pitanje i metodom uključivanja odbacuje većinu sastavaka koji su logički slabiji od onih već prisutnih.

Pregledom rezultata u oblasti ADT mora se priznati da cilj još nije postignut ni izdaleka jer se pokazalo da je oblast izuzetno kompleksna. Treba istaći da se u skorije vreme ipak pojavilo nekoliko primera gde su programi doprineli dokazivanju novih teorema koje su osrednje teške.

Glavni problem je što programi nemaju dovoljno razvijenu inteligentnu strategiju za vodjenje programa kojom bi omogućili generisanje što više pravih izvodjenja, a što manje pogrešnih.

Ambiciozan cilj, pored dokazivanja teorema, za programe u budućnosti uključuje i generisanje novih teorema, izvodjenje novih pojmova i definicija ili čak generisanje novih teorija.

U ranijem periodu većina istraživanja je bila vezana za nalaženje restrikcija za primenu rezolucije i dokazivanje kompletnosti te restrikcije. Drugim rečima trebalo je pokazati da program koristeći te restrikcije još uvek može, u principu, da dokaže bilo koju teoremu u zadatom predikatskom računu prvog reda. U praksi efektivnost znači da program koristeći te restrikcije u raspoloživom vremenu i prostoru može da izvede razuman broj dokaza. Sve dok se zahteva kompletnost u opštem slučaju se dozvoljava više dedukcija no što raspoloživi resursi u praksi omogućavaju. Stoga počev od 70-tih godina istraživači polako usmeravaju istraživanja od kompletnosti ka efektivnosti, ka programima specijalne namene i primenama van matematike. U novije vreme sve se više istražuju verovatnosni i sistemi sa nepotpunim znanjem za izvodjenje zaključaka koji slede iz pretpostavki sa odredjenom verovatnoćom i pod odredjenim pretpostavkama o svetu.

Glavni problem svih pristupa leži u kontrolisanju prostora pretrage odnosno u vodjenju programa ka pravim dedukcijama. Ljudi u toku dokazivanja obradjuju vrlo mali deo prostora pretrage, a koraci koje oni poduzimaju su najčešće relevantni. Intuitivno čovek oseća u čemu je problem i koje su dedukcije one prave. Prenošenjem tih ljudskih mogućnosti na računar znatno se unapredjuju dokazivači. U narednom periodu treba automatizovati razne heuristike koje ljudi koriste. To uključuje istraživanja u: korišćenju primera i kontraprimera, odredjivanju i upotrebi specijalnih slučajeva, rezonovanju po analogiji i upotrebi lema u toku dokazivanja. Sve to je već razmatrano do odredjene granice, ali predstoji dalje studiranje i razvijanje.

Ideja o mehaničkoj napravi koja rezonuje na nivou ljudskih kognitivnih mogućnosti stara je koliko i san o mašini. Stoga su se prvi računarski programi koji dokazuju teoreme simuliranjem ljudskog rezonovanja pojavili već 1950. godine istovremeno sa pojavom prvih računara.

Jedan od značajnijih iz tog perioda je računarski program sa nazivom Logic Theory Machine <63> koji su realizovali Newell, Shaw i Simon. Program radi u domenu iskaznog računa. Idejno je koncipiran tako da koristi tehnike kojima ljudi rešavaju probleme <81>. Organizacija pretrage u dokazu se razlikuje od standardnih algoritamskih metoda kao što su istinosne tablice. Značajna novina koju su oni uveli u ADT je tehnika rada unazad tj. od teoreme koju dokazujemo ka eventualno prostijim podproblemima, što je ubrzo široko prihvaćeno. Izvodjenje dokaza je vrlo blisko ljudskom načinu rada ali na skromnom domenu iskaznog računa.

Gelernter sa saradnicima u programu The Geometry Theorem Machine daje sličnu organizaciju pretrage dokaza kao što je u Logic Theory Machine. Kao novinu uvodi upotrebu problemskih heuristika. To su pravila koja su najčešće vrlo korisna i bitno skraćuju dokaz, ali nisu uvek primenljiva. Dokazivač se koristi dijagramom koji se obično studentima prezentira uz problem kako bi se suzile alternative u traganju za dokazom. I ovde se dokaz generiše unazad počev od cilja. Takvu pretragu nazivamo metod redukovanja problema.

Norton je 1960. razvio dokazivač za teoriju grupa koji takodjer koristi metodu redukovanja problema. U organizaciji pretrage dokaza uključen je veći broj heuristika posebno za nejednakosti.

Guard i ostali istovremeno razvijaju računarski program za ADT <40> gde čovek vodi kontrolu vodjenja dokaza. Interaktivna kontrola se i u sadašnjem trenutku smatra najefikasnijim načinom kontrole vodjenja dokaza bar za blisku budućnost. U interaktivnom radu manifestuje se idealno sprega ljudske intuicije i nepogrešivosti računara.

Brown razvija dokazivače za aritmetiku i teoriju skupova <13> i <14> koristeći teoremu u obliku liste a dokaz izvodi nizom transformacija koje ne menjaju istinosnu vrednost tvrdjenja.

Bundy pravi dokazivač za aritmetiku <16>, Pastre za teoriju skupova <65>, a Ballantyne i Benett za oblast topologije <2> koji su zasnovani na principu da se hipoteza teoreme predstavi dijagramom na osnovu kojeg se utvrdjuju razne posledice hipoteze sve dok se ne potvrdi zaključak procesom grananja unapred pomoću dijagrama.

Bledsoe i grupa saradnika razvili su UT dokazivač teorema <7> i <9> na interaktivnom principu za predikatski račun prvog reda sa prirodnim izvodjenjem i heurističkim pristupom uz razbijanje na podciljeve, pojednostavljenja koja zavise od konteksta, algebarska pojednostavljenja, kontrolisano grananje, upotrebu lema itd. Oblast primene su teorija skupova, topologija, teoreme u dokazivanju korektnosti programa i najzad oblast matematičke analize. Ovo je trenutno nalkompletniji dokazivač koji primenjuje heuristički pristup testiran na širokom domenu. Detaljniji prikaz će biti dat u odeljku 2.1.

Pomenimo još Brown-ovu ideju iz <15> kombinovanja deduktivnih mogućnosti dokazivača sa sistemom sa induktivnim rezonovanjem koji generiše nove rečenice nad datim jezikom koje bi se upućivale dokazivaču.

Ideja je da takav programski matematički sistem za rezonovanje uz pomoć sistema za uvodjenje novih matematičkih definicija, sistema za analizu povoljnijih oblika prezentacije, proceduralnog sistema za kodiranje i sistema za otklanjanje grešaka može da kreira odnosno konstruiše matematičke teorije.

Na području mašinskog učenja, induktivnog rezonovanja i formiranja teorija najznačajniji doprinos je dao Lenat (opisano u <52> - <54>) koji je programima AM i EURISKO pokazao da heurističkom pretragom možemo modelirati brojne aspekte kreativnog istraživanja kao i to da programi mogu generisati nove heuristike.

2.1 DOKAZIVACI SA PRIRODNIM IZVODJENJEM I HEURISTICKIM PRISTUPOM RAZVIJENI U SVETU

Kao sto se vidi iz prethodnog odeljka dva slogana: **SIMULIRAJ LJUDE** i **KORISTI MATEMATICKU LOGIKU** su odredila osnovne pravce u budućim istraživanjima u oblasti ADT. Obe struje su značajne i u obe su kreirani korisni sistemi automatskog izvodjenja dokaza.

U ovom odeljku ćemo opisati neke detalje i koncepciju rada dokazivača razvijenih u svetu koji su značajnije doprineli pravcu istraživanja kojim se simulira dokazivanje teorema na način kako to ljudi rade.

2.1.1 Dokazivač koji su razvili Newel, Shaw i Simon

Newel, Shaw i Simon su razvili dokazivač pod imenom "Logic Theory Machine" (u daljem tekstu LT) koji dokazuje teoreme u iskaznom računu koristeći pored standardnih izvodjenja od aksioma ka zadatoj teoremi i postupak rada unazad koji nije kompletan, naime za datu teoremu nemamo garanciju da ćemo koristeći data pravila izvodjenja završiti na aksiomama ili poznatim teoremama. Naravno valjanost je ispunjena. Primenjeni metodi garantuju da je svaki generisani podproblem deo niza formula koje završavaju u zadatoj teoremi i da je svaki generisani podproblem izveden po pravilima izvodjenja iz predhodnih.

Mada ovaj heuristički metod pri neograničenim resursima teorijski ne može da dokaže svaku teoremu u praksi se pokazao kao prilično efikasan i značajno je uticao na dalji razvoj ovog pristupa ADT. U njihovom radu <63> je prvi put ukazano na značaj podciljeva i traženje supstitucije pri ujednačavanju.

Predjimo sad na opis koncepcije dokazivača.

Podjimo od aksioma iskaznog računa i sledećih pravila izvodjenja:

1) **Zamena promenljive izrazom.** U bilo kojoj teoremi sva pojavljivanja neke promenljive možemo zameniti novim izrazom tj. formulom iskaznog računa. Na primer u aksiomi $p \Rightarrow q \vee p$ možemo sva pojavljivanja promenljive p zameniti sa $q \wedge p$ čime se dobija $q \wedge p \Rightarrow q \vee (q \wedge p)$.

2) **Zamena logičkog veznika svojom definicijom.** Na primer veznik $p \Rightarrow q$ se definiše da znači isto što i $\neg p \vee q$. Pritom je dozvoljena i zamena definiensa definiendumom i obrnuto. Ako u prethodnoj aksiomi zamenimo implikaciju po definiciji dobija se $\neg p \vee (q \vee p)$.

3) **Pravilo modus ponens.** Ako su A i $A \Rightarrow B$ teoreme tada je i B teorema.

Podjimo od aksioma i dotle dokazanih teorema. Primenom navedenih pravila izvodjenja nekontrolisano, brzo bi došlo do prekoračenja resursa. Stoga se u LT uvode heuristike nazvane metodi. One predstavljaju niz operacija koje čine glavni i stalni doprinos ka nalaženju dokaza.

Heuristike su sledeće:

1) **Metod supstitucije.** Njime se traži dokaz zadate formule nalaženjem aksiome ili prethodno dokazane teoreme koja se može nizom primena zamena promenljive i zamenom logičkih veznika po definiciji transformisati tako da se kao rezultat dobije zadata formula.

2) **Metod modus ponensa.** Pokušava se zamena zadate formule novom formulom primenom pravila modus ponens, ali tako da dokaz nove formule obezbedjuje dokaz zadate formule. Na primer ako se traži dokaz za B ovaj metod će , ukoliko nadje aksiomu ili teoremu oblika $A \Rightarrow B$ zameniti dati podcilj sa A kao novim podproblemom. Ako se A dokaže budući da je $A \Rightarrow B$ već teorema i B će biti dokazano.

3) **Metod grananja.** Ovaj metod koristi tranzitivnost relacije implikacije da bi generisao novi podproblem za koji važi da njegov dokaz obezbedjuje dokaz zadate teoreme. Tako će izraz A => C metodom grananja unapred tražiti aksiomu ili teoremu oblika A => B i zatim dati izraz zameniti novim podciljem oblika B => C.

Slično korišćenjem ove metode grananjem unazad, ako se pronadje aksioma ili teorema oblika B => C, data formula se zamenjuje novim podciljem A => B, koji, ako ga dokažemo, obezbedjuje dokaz date formule A => C.

Ta tri metoda su nezavisna i koriste se tako što se jedan primeni na podprobleme generisane drugim metodom. Njihova primena obezbedjuje rad unazad od date teoreme koju dokazujemo ka aksiomama i poznatim teoremama generisanjem podproblema. Ovi metodi ne garantuju kompletnost, pa čak ni to da generisani podproblem mora da bude istinit. Netačnost potproblema, naravno, ne obezbedjuje dokaz polaznog tvrdjenja. Tek kad dokažemo istinitost potproblema, potvrdili smo i polazni cilj.

Opšti metod traženja dokaza je organizovan na sledeći način:

1) Za zadatu formulu koju treba dokazati najpre se proba metod supstitucije koristeći sve aksiome i teoreme koje su u sistemu smeštene u odgovarajućim datotekama.

2) Ako 1) ne uspe proba se metod modus ponensa i za svaki novo generisani problem sa primenom modus ponensa pokušava se dokaz tog podproblema metodom supstitucije. Ako supstitucija ne uspe podproblem se dodaje na listu podproblema.

3) Ako modus ponens ne uspe za sve teoreme na listi teorema, isti ciklus se ponavlja sa grananjem unazad i to najpre probamo da generišemo podproblem , a zatim da ga dokažemo metodom supstitucije. Ako nije uspelo novi podproblem se dodaje listi podproblema.

Ako metod supstitucije uspe sa nekim podproblemom zadata teorema je dokazana.

4) Ako su svi metodi pokušani za zadatu formulu i nije generisan dokaz rutina bira sledeći neisprobani podproblem sa liste podproblema i sa njim ponavlja postupak.

Proces se nastavlja sve dok se ne ispuni jedan od sledećih uslova:

1) nadjen je dokaz,
2) programirano vreme predvidjeno za dokazivanje je isteklo,
3) u računaru više nema raspoloživog prostora,
4) iscrpili smo sve podprobleme sa liste.

Za efikasnu pretragu vrlo je značajna primena heuristike koja s jedne strane usmerava ka cilju, a sa druge odbacuje varijante za koje nema šanse da se do cilja stigne.

Primena rutine testa sličnosti prethodi primeni rutine za ujednačavanje i ova druga se primenjuje samo ako je testom sličnosti konstatovano da su formule slične, čime se značajno skraćuje pretraga.

Test sličnosti sastoji se u sledećem:

Dve formule su slične ako su im leve i desne strane jednake u odnosu na:

1) maksimalan broj nivoa od glavnog logičkog veznika do bilo kojeg slova,
2) broj različitih slova,
3) broj mesta tj. broj pojavljivanja slova.

2.1.2 Dokazivač koji su razvili Gelernter i saradnici

Gelernter i njegovi saradnici razvili su dokazivač nazvan GEOMETRY MACHINE (u daljem tekstu GM). Slično kao i u LT i on se bazira na dobro poznatom analitičkom metodu tj. radu unazad od zadate teoreme koju treba dokazati ka aksiomama. Glavni doprinos je u uvodjenju heuristika kojima se otkrivaju lažne generisane rečenice korišćenjem dijagrama. Ovo je opisano u <39>. Odbacivanjem lažnih generisanih rečenica koje nemaju model na dijagramu, znatno se redukuje broj grana u generisanom drvetu što je posebno značajno za drveta sa većom dubinom, čime se znatno povećavaju šanse za nalaženje dokaza.

Heuristički pristup zastupljen je i u prepoznavanju sintaksne simetrije odredjene klase stringova, vidi <40>.

Ovo je heuristika kojom se imitira rad matematičara koji za A i B koji su sintaksno simetrični dokaže A, a za B kaže dokaz je analogan. Na računaru mašina ispituje sintaksnu simetričnost dva izraza. Ukoliko je potvrdi i za jedan od izraza postoji dokaz, mašina će i drugi proglasiti tačnim uz primedbu da je sintaksički konjugat prvog izraza. Kako su dokazi sintaksno simetričnih izraza identični sintaksički konjugati se isključuju iz grafa dokaza.

Radeći unazad sistem GM generiše graf rešenja problema definisan na sledeći način: neka je G_0 tvrdjenje koje treba dokazati. To je cilj problema. Ako je G_n formalno tvrdjenje sa osobinom da G_{n-1} direktno sledi iz G_n tada kažemo da je G_n podcilj reda n za dati problem. Svi G_j za $j<n$ su viši podciljevi od G_i, dok je G_0 podcilj reda 0. Svaka grana predstavlja transformaciju iz G_n u G_{n-1}. Problem je dokazan kad neko G_n može direktno da se izvede iz premisa i aksioma.

Kad je podcilj konjunkcija tvrdjenja graf se razbija na tom mestu i svaki paralelni podcilj moramo posebno dokazati.

Opšta koncepcija dokazivača GM je sledeća:

1) Pravi se dijagram. Konstruišu se tri liste: za segmente, uglove i trouglove. Svakom elementu liste pridel jena je podlista koja ga opisuje.

2) Pravi se početna konfiguracija sistema. Premise se smeštaju na listu dokazanih formula, a zaključak na graf rešenja kao nulti podcilj.

3) Definicije neprimitivnih predikata u premisama se dodaju listi dokazanih formula.

4) Iz grafa rešenja se bira tekuci podcilj koji treba dokazati. Ukoliko na grafu nema više podciljeva dokaz nije uspeo i prelazimo na 10).

5) Biraju se odgovarajuce aksiome i teoreme iz memorije i radeci unazad geneiršemo skup nižih podciljeva tako da ako je bar jedan od njih dokazan, odgovarajuci podcilj se može dokazati modus ponensom i odgovarajucom aksiomom ili teoremom.

6) Odbacuju se svi podciljevi iz tog skupa nižih podciljeva koji nisu valjani na dijagramu kao i oni koji su viši podci-ljevi na grafu ili su sintaksno simetrični nekom višem cilju.

7) Ako je neki od podciljeva iz skupa nižih podciljeva valjan na osnovu svog primera na listi dokazanih formula, ili se može pretpostaviti sa dijagrama tekuci podcilj je dokazan.

Ako smo potvrdili neki niži podcilj prelazimo na 9), inače na 8).

8) Prihvatljivi niži ciljevi koji nisu suvišni dodaju se grafu i bira se novi tekuci podcilj.

Ako nema prihvatljivih nižih podciljeva a moguća je konstrukcija na tom mestu, tekuci podcilj se obeležava kao privremeno bez potomaka. Ako konstrukcija nije moguća, ili ako je računar pokušao i nije našao nijednig potomka, tekuci cilj se proglašava bez potomaka.

Prelazimo na 4).

9) Ako je tekuci podcilj dokazan dodaje se listi dokazanih

formula zajedno sa svim višim posledicama koje su odredjene grafom. Ako nema paralelnih podciljeva koji su ostali da se dokažu računar rekonstruiše dokaz sa grafa i štampa ga. U protivnom ide na 4)

10) Ako u nekom momentu svaki slobodni podcilj na grafu postane bez potomaka, mašina ne uspeva, naravno pod uslovom da prethodno nije potrošen memorijski prostor ili strpljenje korisnika.

Glavne heuristike se nalaze unutar blokova 4), 5) i 6). One služe za odbacivanje biranje i razvijanje podciljeva.

Na primer heuristika za izabiranje podcilja koji će najverovatnije uspeti se zasniva na uporedjivanju skupa premisa i podcilja prema predikatima i logičkim veznicima. Bira se onaj podcilj čiji se skup predikata i logičkih veznika po broju najviše poklapa sa nekom premisom.

2.1.3 Dokazivač koji je razvio Pastre

Program Pastre-a dokazuje teoreme u oblasti teorije skupova korišćenjem metoda koje su analogne ljudskom načinu dokazivanja. I ovde se vrši razbijanje problema na više jednostavnijih koji su prostiji za dokaz. Heuristika koja se primenjuje na stavove koji više ne mogu dalje da se razbiju, zasniva se na specifičnom predstavljanju. Opšti metod dokazivanja je da se hipoteze prihvate kao tačne, a program treba da istraži sve posledice koje slede iz njih sve dok se ne potvrdi zaključak. Ukoliko to ne uspe formalni iskaz teoreme se modifikuje i isto ponovi za novo tvrdjenje. Program nije kompletan tj. ne može da dokaže svaku teoremu. Detaljan opis programa je dat u <65>, a ovde navodimo neke značajnije heuristike.

Specifičnost prezentacije je u predstavljanju binarnih relacija. Na primer relacija $R(x,y)$ se predstavlja na grafu tako što se uvodi grana od x ka y sa obeležjem R.

$$x \xrightarrow{R} y$$

Pripadanje skupu koje se često predstavlja kao unarni predikat ovde se prikazuje kao binarna relacija, na primer $x \in A$ na grafu daje

$$x \xrightarrow{\in} A$$

Analogno i funkcije $f(x)=z$ se na grafu predstavljaju sa

$$x \xrightarrow{f} z$$

Sva tvrdjenja sa kojima program radi se zamenjuju u jedno ili više tvrdjenja sledećeg oblika:

$$H_1 \wedge H_2 \wedge \ldots \wedge H_n \Rightarrow C_1 \vee C_2 \vee \ldots \vee C_m$$

sa značenjem : ako su sve hipoteze H_i $i=1,2,..n$ tačne onda je bar jedan C_j tačan za $j \in \{1,2,\ldots,m\}$.

Svaka hipoteza je oblika:

-predikat $P(x,y,\ldots)$ gde su argumenti promenljive ili konstante,

-binarna relacija Rxy koja se procesira na poseban način,

-univerzalna hipoteza oblika $(\forall x)(\forall y)\ldots(A(x,y,\ldots) \Rightarrow B(x,y,\ldots))$, gde je tvrdjenje u zagradi ponovo u standardnoj formi,

-egzistencijalna hipoteza $(\exists x)\ Q(x,\ldots)$ gde je $Q(\ldots)$ jedinična hipoteza ili relacija ili konjunkcija takvih hipoteza,

-neka od konstanti "tačno" ili "lažno",

-disjunkcija hipoteza gore navedenih oblika.

U opstem slučaju m=1 i postoji jedan disjunkt u zaključku koji je
- jedinični,
- relacija Rxy,
- egzistencijalni,
- konstanta "lažno".

Hipoteze koje su relacije se izdvajaju sa liste hipoteza i predstavljaju na grafu. Ova reprezentacija redukuje memorijski prostor i vreme izvršenja.

Opšta koncepcija dokazivanja koristi grananje uz zadržavanje centralne logičke implikacije. Teorema oblika $H_1 \wedge H_2 \wedge \ldots \wedge H_n \Rightarrow C$ se dokazuje tako što program traži sve posledice koje slede iz hipoteza H_i pa zatim one postaju nove hipoteze. Primeniće se tvrdjenje $A_1 \wedge \ldots \wedge A_p \Rightarrow B$ ako postoji supstitucija koja ujednačava sve A_i-ove sa nekim H_j. Zatim se B izmenjen datom supstitucijom dodaje listi hipoteza, sem u slučaju kad već postoji na listi kao hipoteza.

Teorema je dokazana:
1) Ako je zaključak C sadržan u hipotezama ili je generisan kao nova hipoteza.
2) Ako je izvedena kontradikcija tj. ako je B konstanta "lažno".
3) Ako je zaključak C oblika $(\exists x)P(x)$ a u grafu postoji elemenat koji zadovoljava P.

U programu su, imitiranjem metoda koje matematičari koriste u dokazivanju teorema, izgradjene heuristike koje vode dokaz.

Ako se uporedi ovaj program sa Brownovim <14>, koji takodjer dokazuje teoreme u oblasti teorije skupova može se videti da su im mogućnosti komplementarne. Zbog različitih koncepcija vodjenja dokaza i nalaženja supstitucija pri utvrdjivanju unifikacije u nekim primerima je uspešniji jedan, a u drugim drugi.

2.1.4. Dokazivači koje je razvio Bledsoe sa saradnicima

Kao što se vidi iz dosadašnjeg pregleda ADT rani radovi su bili posvećeni heurističkom pristupu, dok su istraživanja 60-tih godina bila više posvećena usavršavanju rezolucijski zasnovanih dokazivača. Nakon 70-tih godina ponovo se sve češće koriste heurističke metode, upotrebljava se domenski orjentisano znanje i koristi interaktivan rad.

Uloga rezolucije se često svodi na minimum tako što joj prethodi procedura prirodnog izvodjenja koja razgrana dokaz i eventualno ga kompletira u nekim jednostavnijim slučajevima <5>. Ovaj pristup je sledjen i u dokazivaču u sistemu GRAPH koji će biti opisan u narednim poglavljima.

Bledsoe je u početku radio na unapredjenju rezolucije a zatim je postupno unapredjivao sistem prirodnog izvodjenja koji je blizak prirodnoj dedukciji Gentzenovog tipa i potpuno odbacio rezoluciju kao pravilo izvodjenja.

Svojim radom on je preneo na računar niz metoda koje ljudi koriste u dokazivanju teorema i značajno unapredio i poveo istraživanja u tom pravcu.

U <5> se opisuje procedura razbijanja na podciljeve nazvana SPLIT kojom se tekući cilj oblika

$$A \wedge B$$

razbija na dva nova podcilja A i B koji se dalje svaki posebno dokazuju. Slično

$$A \Leftrightarrow B$$

se razbija na $A \Rightarrow B$ i $B \Rightarrow A$ i analogno za slučajeve koji se primenom ekvivalentnih transformacija mogu dovesti na oblik gde je konjunkcija vrhovna logička operacija. Taj metod je početak približavanju ka sistemu prirodnog izvodjenja, mada se u <5> primenjuje pre korišćenja rezolucije.

Kasnije su i mnogi drugi istraživači preuzeli razne varijante razbijanja na podciljeve kao što se vidi u <13>, <16>, <58>, <65> itd.

U <5> se takodjer uvodi i lista REDUCE kao skup domenski orjentisanih heuristika za teoriju skupova kojima se izvode neke transformacije koje i matematičari koriste u dokazima.

Izmedju ostalog tu su:

$S \in A \cap B$ zamenjujemo sa $S \in A \wedge S \in B$,
$S \in A \cup B$ zamenjujemo sa $S \in A \vee S \in B$,
$S \in \neg A$ zamenjujemo sa $\neg(S \in A) \wedge S \in U$,
$S \in$ SB A zamenjujemo sa $S \subset A \wedge S \in U$,
$S \in$ SNG A zamenjujemo sa $S=A \wedge S \in U$...itd.

gde U označava univerzalni skup, SB partitivni skup, a SNG singlton.

U <7> je prikazan dokazivač koncipiran interaktivno gde se učešće čoveka koristi da bi se povećala moć i brzina dokazivača. Ovaj sistem bazira na sistemu izvodjenja prirodnog tipa.

U cilju lakse komunikacije čoveka i računara, teoreme se predstavljaju u prirodnom obliku ili u obliku drveta čime se korisniku prikazuje njihova logička struktura. Da bi se korisnik lakše snalazio pri dužim formulama data je i mogucnost prikaza formule u kojoj su sva pojavljivanja nekog izraza zamenjena sa F ukoliko je ranije izvršena zamena F-a tim izrazom.

Intervencije korisnika svedene su na minimum, naime inicijativa se prepušta čoveku tek ukoliko računar sam ne uspe da kompletira dokaz unutar datog prostora i vremena.

Nema neprirodnih konverzija kao što je zamena $A \Rightarrow B$ u $\neg A \vee B$ kod primene rezolucije.

Čovek može da bira neku od komandi koje cemo kasnije navesti i objasniti njihovo dejstvo. Vecina komandi je tako koncipirana da ukoliko se njihovom primenom podcilj izmeni program traži odobrenje od korisnika. Ukoliko se covek saglasi sa predloženom izmenom, ona će se i definitivno sprovesti.

Sledeća tabela daje pregled raspoloživih komandi:

KOMANDA	KORISNIK OTKUCA	AKCIJA SISTEMA
PUT	PUT X = ( )	Sistem zamenjuje svako pojavljivanje X u tekućem podcilju sa datim izrazom ( ).
DEFN	D A	Zamenjuje sva pojavljivanja A sa datom definicijom.
USE	USE N	Iz memorije uzima teoremu sa brojem N i dodaje je kao konjunkt hipoteze (levoj strani centralne implikacije) tekućeg podcilja.
	USE ( )	Dodaje ( ) kao konjunkt hipoteze.
LEMMA	LEMMA ( )	Najpre dokazuje lemu ( ), a zatim poziva komandu USE ( ).
PROCEED	GO	Nastavlja sa dokazom bez izmena od strane korisnika.
DETAIL	DETAIL	Generiše podcilj tekućeg cilja i predstavlja ga korisniku i čeka dalju akciju.
COUNT	CNT N	Povecava vremensko ograničenje za tekući

		podcilj N puta.
FAIL	F	Utvrdjuje da je tekuci podcilj lažan.
ASSUME	A	Utvrdjuje da je tekuci podcilj tačan.
BACKUP	REJECT	Vraća se u dokazu na prethodni cilj.
REORDER	(N M)	Preuredjuje hipotezu oblika n-arne konjunkcije i zaključak oblika m-arne disjunkcije tako da na prvo mesto u hipotezi ide N-ti konjunkt a u zaključku M-ti disjunkt.
DELETE	DELETE N,M...	Izostavlja hipoteze broj N,M...
PRETTY PRINT	T P	Tekuci podcilj štampa kao drvo kako bi se lakše videla struktura logičkih operacija u formuli.
	T P F	Ako je ranije upotrebljena komanda PUT F ( ) podcilj se prikazuje u obliku gde je svako pojavljivanje ( ) zamenjeno simbolom F.

	TP C F	Isto kao predhodno ali se odnosi samo na zaključak formule.
	TP H F	Analogno za hipotezu.
HISTORY	RUN HISTORY	Sistem daje ceo dokaz uz eliminisanje neproduktivnih koraka.
ADD REDUCE	ADD-REDUCE ( )	( ) se dodaje spisku pravila za redukovanje.
ADD PAIRS	ADD-PAIRS ( )	( ) se dodaje spisku u tabeli parova.
ADD DEFN	ADD-DEFN (A( ))	( ) se dodaje spisku definicija kao definiens a definiendum je atom A.

Predjimo sada na opis rada dokazivača.

Sistem prirodnog izvodjenja nazvan IMPLY se sastoji od sledecih pravila za transformisanje podciljeva:

$H \Rightarrow (B \Rightarrow C) \longrightarrow H \wedge B \Rightarrow C$

$H \Rightarrow (B \Leftrightarrow C) \longrightarrow (H \wedge B \Rightarrow C) \wedge (H \wedge C \Rightarrow B)$.

Glavne funkcije sistema su razbijanje na podciljeve, na primer cilj $H \Rightarrow A \wedge B$ razbija na dva nova podcilja $H \Rightarrow A$ i $H \Rightarrow B$, zatim funkcija grananja unazad i unapred i supstituisanje jednakih.

U osnovi programa IMPLY je program za unifikaciju. Ako je σ najopštiji unifikator od A i B tada se podcilj $A \Rightarrow B$ može smatrati tačnim, a σ se primenjuje na sledeće podciljeve.

Program IMPLY radi na sledeći način:

Za dati podcilj oblika H => C najpre poziva program REDUCE koji primenjuje pravila za prevodjenje podformula datog podcilja u drugi oblik korišćenjem zadatih domenski orjentisanih heuristika,

na primer u teoriji skupova:

$t \in A \cap B \longrightarrow t \in A \wedge t \in B$,

$t \in A \cup B \longrightarrow t \in A \vee t \in B$.

$0 \in A \longrightarrow$ "tačno".

itd. kao što je gore navedeno ili prikazano u <5>.

Zatim program razbija na podciljeve ako može. Posle razbijanja pokušava se metodom pretrage najpre u širinu sledecih sedam koraka tim redosledom. Ako neki korak ne uspe prelazi se na sledeci. Uspeh koraka obično rezultuje novim pozivom funkcije IMPLY.

1) Ujednačavanje,
2) Ujednačavanje i grananje unazad,
3) Korišćenje rutine PAIRS,
4) Pravilo PEEK.
5) Definiši C,
6) Definiši "čudne" pojmove,
7) Definiši bilo koji pojam.

Ovo bi bio globalni pregled rada dokazivača.

Pomenimo još da je Bledsoe u svojim radovima primetio da primenu definicija treba strogo kontrolisati i dao neka gruba uputstva za pravac daljeg istraživanja, kao što je primena definicija samo u desnoj strani centralne implikacije, definisanje čudnih pojmova pre onih standardnih itd.

U <31> je dat novi pristup u tom pravcu, a u poglavlju 6.1 će biti detaljnije opisana heuristika za izbor relevantne definicije ili leme kojom se transformiše tekući podcilj u dokazivaču u sistemu GRAPH.

3. ARITMETICKA TEORIJA GRAFOVA, FORMALIZACIJA I ORGANIZACIJA ZNANJA

Na Elektrotehničkom fakultetu Univerziteta u Beogradu, pod rukovodstvom akademika prof. D. Cvetkovica razvijan je od 1980. - 1985. ekspertni sistem za klasifikaciju i unapredjenje znanja iz oblasti teorije grafova sa namenom da pomogne u naučnim istraživanjima u toj oblasti. Sistem GRAPH sadrži sledeća tri glavna podsistema: BIBLI deo za obradu bibliografskih podataka, ALGOR deo za obradu grafova i najzad THEOR koji sadrži dokazivač teorema.

U daljem izlaganju bice dat prikaz dela THEOR i to dokazivača zasnovanog na prirodnom izvodjenju sa heurističkim pristupom. Formule u drvetu dokaza zadržavaju svoj prirodan oblik (kako ih matematičari koriste i zapisuju). Ne uvode se neprirodne transformacije kao što je izbacivanje implikacije i ekvivalencije kod primene rezolucije ili prirodne dedukcije. Slede se postupci koje graf teoretičari koriste kad dokazuju teoreme iz te oblasti kako bi se olakšalo praćenje dokaza i eventualna intervencija čoveka.

Dokazivač koji koristi rezoluciju i indukciju kao pravila izvodjenja, koji je takodjer sastavni deo dela THEOR i koji se koristi kao pomoćni moduo u sklopu ovog dokazivača za dokazivanje uprošćenih tvrdjenja, ovde neće biti razmatran, a opisan je u <33>, <44> i <45>.

Kako je sistem interaktivan kao jedan od prvih problema pojavljuje se problem komunikacije čoveka i računara. U ovom delu sistema koristi se jezik "pravi GTCL" (što je skraćenica od reči Graph Theoretical Computer Language) koji predstavlja mali podskup uprošćenog i formalizovanog dela engleskog jezika. Sintaksa jezika je opisana u <82>.

U radu sa sistemom, rečenice iz oblasti teorije grafova koje korisnik saopšti računaru, prevode se na nivo formula kvantifikatorskog računa prvog reda i, naravno, interno na sistem kodova. U cilju efikasnije komunikacije sa korisnikom u toku rada rečenice se sa nivoa formula kvantifikatorskog računa mogu ponovo, po potrebi, vratiti na nivo prirodnog jezika t.j. na nivo "pravi GTCL" buduci da u sistemu postoji i inverzni prevodilac.

Programski moduo za dokazivanje teorema iz oblasti teorije grafova mogao bi sa malim modifikacijama da se koristi i za dokazivanje teorema u drugim oblastima matematike pod uslovom da se poštuje sintaksa jezika "pravi GTCL" i da se radi o teoriji prvog reda.

Teorija grafova je formalizovana pomoću AGT (aritmeticke teorije grafova) koja predstavlja proširenje formalne aritmetike, a obuhvata sve delove teorije grafova (usmereni i neusmereni grafovi, multigrafovi itd). Formalizaciju je moguće uvesti i na razne druge načine, a ovu koja će biti predstavljena u ovom odeljku uveo je D. Cvetkovic. Sledi pregled AGT u obimu potrebnom za ilustrovanje rada dokazivača, a detaljniji opis je dat u <27>.

U AGT postoje tri tipa promenljivih koje razlikujemo prema tipu slova. Obeležavamo ih nizom od najviše pet simbola od kojih je prvi obavezno slovo, a ostali simboli, ako ih uopšte ima, su brojevi i imaju ulogu indeksa. Prema početnom slovu odredjujemo tip promenljive na sledeci način:

X,Y,Z promenljive koje označavaju čvorove,
U,V,W promenljive koje označavaju grane,
K,L,M,N promenljive koje označavaju prirodne brojeve.

Na sličan način nizom gore opisanog oblika čije je početno slovo neko od sledecih slova označavacemo sa:

G,H imena grafova,
D konstante,
F funkcijska slova,
A operacije nad grafovima,
P,Q,R,S,T redom predikatska slova arnosti n gde je n=0,1,2,3,4 respektivno.

Jedini izuzetak od gornje konvencije su binarne relacije oblika $X \alpha Y$ gde $\alpha \in \{=, \neq, >, <, \geq, \leq\}$ koje se ne zapišuju u obliku R(X,Y) poput ostalih dvomesnih predikata. Njihov zapis u kvantifikatorskoj formuli je u obliku XαY, kao što je uobičajeno.

U AGT postoji konvencija o prioritetu logičkih operacija uvedena tako da su ∀, ∃ i ¬ najveceg prioriteta, a zatim prioritet opada sleva udesno za operacije ∨, ∧, => i <=>. Za dve operacije istog prioriteta najpre se izvršava prva u zapisu tj. po konvenciji prioritet opada sleva udesno.

Takodjer, svaki kvantor vezuje različito označene vezane promenljive, a skup vezanih promenljivih i skup slobodnih promenljivih su disjunktni.

Postoji još i konvencija po kojoj se različito kodiraju zagrade kojima ogradjujemo argumente predikata i funkcija od onih zagrada kojima menjamo prioritet logičkih operacija, kao i operacija na nivou terma. Ova poslednja konvencija nema značaja za zapis na nivou formule kvantifikatorskog računa jer tada nema razlika u predstavljanju zagrada. Razlika se manifestuje tek na nivou kodova u internom zapisu u računaru. Kvantifikatorske formule se predstavljaju sa minimalnim brojem zagrada s obzirom na uvedenu konvenciju o prioritetu logičkih operacija.

Nadalje, pomenimo da su sve konstante, relacije, operacije i aksiome prenete iz formalne aritmetike sa istim značenjem. Pored toga u AGT postoje još i osnovni predikati S1(X,Y,U) sa značenjem "čvorovi X i Y su povezani granom U", Q1(N) i Q2(M) sa značenjem "graf ima N čvorova" i "graf ima M grana" respektivno. Osnovni predikati zadovoljavaju sledeće aksiome:

1) ¬S1(X,X,U),
2) S1(X,Y,U) <=> S1(Y,X,U),
3) S1(X,Y,U) ∧ S1(X,Y,V) => U=V,
4) (∃N)(N>=1 ∧ Q1(N)) ∧ (Q1(N1) ∧ Q1(N2) => N1=N2),
5) (∃M)(M>=1 ∧ Q2(M)) ∧ (Q2(M1) ∧ Q2(M2) => M1=M2),
6) S1(X,Y,U) ∧ S1(X1,Y1,U) => (X=X1 ∧ Y=Y1) ∨ (X=Y1 ∧ Y=X1)

7) $Q3(X) \wedge Q3(Y) \wedge S1(X,Y,U) \Rightarrow Q4(U)$,

8) $Q4(U) \Rightarrow (\exists X)(\exists Y)(Q3(X) \wedge Q3(Y) \wedge S1(X,Y,U))$.

Predikatska slova Q3 i Q4 se definišu kao što ce biti dato u pregledu definicija. Ovime se definišu konačni neusmereni grafovi bez petlji i višestrukih gra-na. Postoji i mogućnost variranja aksioma kako bi se dobili digrafovi, grafovi sa petljama itd.

Graf nije promenljiva buduci da se radi o teoriji prvog reda. Ukoliko želimo da pomenemo graf dodajemo ime grafa kao sufiks naziva predikatskog slova. Tako na pr. S1G(X,Y,U) ima značenje "čvorovi X i Y su spojeni granom U u grafu G". Isti mehanizam dodavanja sufiksa važi za konstante i funkcije. Na ovaj način ne možemo specificirati neki konkretan graf već G označava uvek graf uopšte. Pored toga u sufiksu naziva predikatskog slova mogu da se pojave još i sledeći nastavci:

1) B, 2) BE, 3) E, 4) BCE , 5) BC,

gde na mesto slova B i C dolaze imena grafova, a na mesto slova E naziv unarne ili binarne operacije nad grafovima.

Ovi sufiksi redom označavaju da se predikat koji nosi u sufiksu naziva predikatskog slova odgovarajući nastavak odnosi na:

1) graf sa imenom B,

2) graf koji je dobijen primenom unarne operacije E iz grafa sa imenom B,

3) graf koji je dobijen unarnom operacijom E iz datog grafa,

4) graf koji je dobijen binarnom operacijom E iz grafova sa imenima B i C,

5) grafove B i C.

U toku rada sistem koristi deo znanja iz oblasti teorije grafova koji je smešten u datoteke definicija, aksioma, lema i logički valjanih formula. Datoteke se mogu menjati tako što ih korisnik modifikuje pre početka dokazivanja ili na primer dodavanjem neke nove leme čak i u toku dokaza. Navedimo sada neke osnovne pojmove sa odgovarajućom definicijom na engleskom jeziku (t.j. na jeziku pravi GTCL) onako kako ih korisnik smešta u datoteku definicija. Za svaki pojam biće dat i prevod na nivo AGT kao što je to sistem preveo i interno predstavio u obliku kvantifikatorske formule.

D1. TO BE A POINT: "X is a point iff X is greater or equal to 1 and for all N if the graph has N points then X is less or equal to N".

$$Q4(X) \iff 1 \le X \land (\forall N)(Q1(N) \Rightarrow X \le N).$$

D2. TO BE A LINE: "U is a line iff U is greater or equal to 0 and for all M if the graph has M lines then U is less or equal to M".

$$Q5(U) \iff 0 \le U \land (\forall M)(Q2(M) \Rightarrow U \le M).$$

D3. ADJACENT POINTS: "Points X and Y are adjacent iff there exists a line U such that X and Y are joined by a line U".

$$R1(X,Y) \iff (\exists U) S1(X,Y,U).$$

D4. ISOLATED POINT: "Point X is isolated iff for all Y, the points X and Y are not adjacent".

$$Q3(X) \iff (\forall Y) \neg R1(X,Y).$$

D5. POINTS ADJACENT IN THE COMPLEMENT OF A GRAPH: "Points X and Y are adjacent in the complement of a graph iff the points X and Y are not adjacent in the graph and the point X is not equal to the point Y".

$R1A1(X,Y) \Leftrightarrow X \neq Y \wedge \neg R1(X,Y).$

D6. <u>POINTS JOINED BY A WALK OF LENGTH</u> K: "Points X and Y are joined by a walk of length K iff (K is equal to O and X=Y) or (K is greather then O and there exists Z such that (X and Z are joined by a walk of length K-1 and the points Z and Y are adjacent))".

$S2(X,Y,K) \Leftrightarrow (K=0 \wedge X=Y) \vee (K>0 \wedge (\exists Z)(S2(X,Z,K-1) \wedge$
$\wedge R1(Z,Y))).$

D7. <u>JOINED BY A WALK</u>: "Points X and Y are joined by a walk iff there exists K such that X and Y are joined by a walk of length K."

$R2(X,Y) \Leftrightarrow (\exists K)S2(X,Y,K).$

D8. <u>COMPLETE GRAPH</u>: "The graph is complete iff for all X and Y if X is not equal to Y then X and Y are adjacent".

$P1 \Leftrightarrow (\forall X)(\forall Y)(X \neq Y \Rightarrow R1(X,Y)).$

D9. <u>CONNECTED GRAPH</u>: "The graph is connected iff for all X and Y points X and Y are joined by a walk".

$P2 \Leftrightarrow (\forall X)(\forall Y)R2(X,Y).$

D10. <u>INCIDENT POINT AND LINE</u>: "Point X and line U are incident iff there exists a point Y such that points X and Y are joined by line U".

$R3(X,U) \Leftrightarrow (\exists Y)S1(X,Y,U).$

D11. <u>ADJACENT LINES</u>: "Lines U and V are adjacent iff there exists point X such that point X and line U are incident and point X and line V are incident".

$R4(U,V) \Leftrightarrow (\exists X)(R3(X,U) \wedge R3(X,V)).$

D12. <u>TOTALY DISCONNECTED GRAPH</u>: "The graph is totaly disconnected iff for all X and Y, X and Y are not adjacent".

$P3 \Leftrightarrow (\forall X)(\forall Y)\neg R1(X,Y).$

D13. TRIVIAL GRAPH: "The graph is trivial iff the number of points is equal to 1".

P4 <=> Q1(1).

D14. GRAPH HAS TRIANGLE: "The graph has a triangle iff there exists X,Y and Z such that (X and Y are adjacent and Y and Z are adjacent and Z and X are adjacent".

$$P5 \Leftrightarrow (\exists X)(\exists Y)(\exists Z)(R1(X,Y) \wedge R1(Y,Z) \wedge R1(Z,X)).$$

D15. DISCONNECTED GRAPH: "The graph is disconnected iff graph is not connected".

$$P6 \Leftrightarrow \neg P2.$$

D16. DISTANCE: "Points X and Y are at distance K iff X and Y are joined by a walk of length K and for all L if L is less then K then X and Y are not joined by a walk of length L".

$$S3(X,Y,K) \Leftrightarrow S2(X,Y,K) \wedge (\forall L)(L<K \Rightarrow \neg S2(X,Y,L))$$

D17. DIAMETER: "The graph is of diameter K iff graph is connected and there exists X and Y such that X and Y are at distance K and for all X1, Y1 and L (if X1 and Y1 are at distance L then L is less or equal to K)".

$$Q6(K) \Leftrightarrow P2 \wedge (\exists X)(\exists Y)S3(X,Y,K) \wedge (\forall X1)(\forall Y1)(\forall L)(S3(X1,Y1,L) \Rightarrow L<=K).$$

D18. EXCENTRICITY: "Point X is of excentricity K iff there exsists Y such that X and Y are at distance L and for all Z and L (if X and Z are at distance L then L is less or equal to K)".

$$R5(X,K) \Leftrightarrow (\exists Y)S3(X,Y,K) \wedge (\forall Z)(\forall L)(S3(X,Z,L) \Rightarrow L<=K).$$

D19. RADIUS: "The graph is of radius K iff graph is connected and there exists X such that point X is of excentricity K and for all X and L (if Y is of excentricity L then K is less or equal L)".

$Q7(K) \Leftrightarrow P2 \wedge (\exists X)R5(X,K) \wedge (\forall Y)(\forall L)(R5(Y,L) \Rightarrow K \le L)$.

D20. CENTRAL POINT: "Point X is a central point iff for all K (if X is of excentricity K then graph is of radius K)".

$Q8(X) \Leftrightarrow (\forall K)(R5(X,K) \Rightarrow Q7(K))$.

D21. K-REGULAR: "The graph is K regular iff for all X, X is of degree K".

$Q9(K) \Leftrightarrow (\forall X)R6(X,K)$.

D22. REGULAR: "The graph is regular iff there exists K such that graph is K regular".

$P7 \Leftrightarrow (\exists K)Q9(K)$.

D23. CIRCUIT: "The graph is a circuit iff graph is connected and graph is 2 regular".

$P8 \Leftrightarrow P2 \wedge Q9(2)$.

Ovo bi bio deo definicija koje cemo koristiti u primerima u apendiksu koji ilustruju rad dokazivaca. Detalji o daljoj formalizaciji teorije grafova u okviru AGT dati su u <27>.

Može se primetiti da ime grafa u sufiksu predikatskog imena ima ulogu skrivene slobodne promenljive. Medjutim, u AGT ne postoji mogucnost specifikovanja grafovske konstante u sufiksu predikatskog imena čime je eliminisana mogućnost greške da se po analogiji sa teoremom koja se odnosi na poseban graf generiše nova teorema. U toku rada na sistemu dozvoljeno je korišcenje principa analogije za teoreme sa grafovskim nastavcima na sledeći način. Kao što je navedeno u pregledu definicija pojam izolovanog čvora u obliku kvantifikatorske formule ima definiciju:

$Q3(X) \Leftrightarrow (\forall Y)\neg R1(X,Y)$.

Ukoliko se u nekoj rečenici u obliku kvantifikatorske formule pojavi na primer predikat Q3GA1(Y) sistem ce u slučaju zamene tog predikata po definiciji (na pr. smene definienduma definiensom) najpre potražiti definiciju izolovanog čvora. Bez menjanja te definicije u fajlu definicija interno ce generisati oblik:

Q3GA1(X) <=> (∀Y)⌉R1GA1(X,Y)

dodavanjem odgovarajućeg grafovskog nastavka i zatim standardnim postupkom preimenovanjem vezanih i supstitucijom slobodnih promenljivih vršimo zamenu predikata Q3GA1(Y) sa (∀Y1)⌉R1GA1(Y,Y1).

Na sličan način sistem zamenjuje po analogiji i u slučaju koriscenja lema, kao što ce biti detaljnije opisano u odeljku 4.2

4. OPIS ALGORITAMA ZA TRANSFORMISANJE KVANTIFIKATORSKIH FORMULA

Razna tvrdjenja iz oblasti teorije grafova sa kojima sistem GRAPH operiše u dokazivaču predstavljaju se u sistemu u više nivoa: engleski tekst, kodirani engleski tekst, kvantifikatorska formula, kodirana kvantifikatorska formula itd. Kako je rad sistema interaktivan čovek komunicira sa sistemom u toku kreiranja dokaza. U cilju uspešnije saradnje čoveka i mašine postoje ugradjene rutine za konverziju rečenica sa jednog nivoa na drugi. Tako korisnik umesto nivoa kvantifikatorske formule može zahtevati prevod na nivo engleskog jezika ukoliko mu to više odgovara. Na nivou kvantifikatorske formule rečenice se predstavljaju sa minimalnim skupom zagrada u odnosu na usvojeni prioritet logičkih operacija i u infiksnoj notaciji. Važe takodjer konvencije o zapisivanju binarnih predikata na dva načina kao i konvencija o različito označenim vezanim i slobodnim promenljivim kao što je navedeno u 3. glavi.

Kvantifikatorska formula može da se predstavi obeleženim korenskim drvetom. Čvorovi na lišću se obeležavaju atomarnim formulama, a ostali čvorovi se obeležavaju kvantifikatorima sa odgovarajućom vezanom promenljivom ili logičkim operacijama $\neg, \wedge, \vee, =>, <=>$. Koren stabla predstavlja vrhovnu logičku operaciju kojoj je domen dejstva cela formula.

Pomenimo i pojam redukovanog drveta. **Redukovano drvo** se dobija kada u polaznom drvetu uklonimo sve čvorove na lišću.

U cilju internog predstavljanja strukture drveta date formule potrebno je izvršiti prethodnu analizu formule koju ćemo izložiti u odeljku 4.1 ove glave.

4.1 ANALIZA KVANTIFIKATORSKE FORMULE

Da bi se omogucilo vršenje bilo kakve transformacije kvantifikatorske formule potrebno je najpre izvršiti njenu analizu što podrazumeva utvrdjivanje domena dejstva logičkih operacija , utvrdjivanje početka i kraja predikatskog naziva (predikatsko slovo + sufiks eventualno), utvrdjivanje početka i kraja predikata, pronalaženje nepotrebnih zagrada, pronalaženje vezanih promenljivih i njihovih lokacija i utvrdjivanje slobodnih promenljivih.

U narednom izlaganju dacemo kratak pregled tih algoritama.

1) **Nalaženje domena dejstva logičkih operacija** sastoji se u sekvencijalnoj pretrazi formule i nalaženju lokacija na kojima se pojavljuju znaci logičkih operacija i kvantifikatori. Zatim se za svaku operaciju ponaosob traži njen domen dejstva i to levi i desni u slučaju binarne logičke operacije i desni u slučaju unarne i kvantifikatora. Radi lakšeg opisa algoritma koristićemo brojač NZ koji daje broj svih otvorenih zagrada i brojač NMZ koji daje broj otvorenih zagrada u tekućem trenutku. U početku sekvencijalne pretrage sa datog mesta u formuli gde je locirana logička operacija za koju tražimo domen, brojače NZ i NMZ postavljamo na nulu. Ako se krećemo u desno svaki simbol otvorene zagrade povećava NZ i NMZ za jedan, dok simbol zatvorene zagrade smanjuje NMZ za jedan. Ako tražimo u pravcu leve strane simboli otvorene i zatvorene zagrade menjaju uloge. Za nalaženje početka levog domena dejstva za binarne logičke operacije $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ počinjemo sekvencijalnu pretragu od pozicije gde je smeštena ta operacija i krećemo se ulevo sve dok ne bude ispunjen jedan od sledeća tri uslova:

a) NMZ=1,

b) NMZ=0 i naišli smo na operaciju nižeg prioriteta,

c) stigli smo do početka formule.

Desni domen dejstva nalazi se analogno.

Za negaciju i kvantifikatore kraj domena dejstva nalazimo tako što najpre uočimo sledeći simbol.

a) Ako je to otvorena zagrada koja ne pripada kvantifikatoru traži se odgovarajuća zatvorena zagrada. Ukoliko unutar zagrada postoji bar jedno predikatsko slovo zatvorena zagrada predstavlja kraj domena dejstva. Ako u zagradi nema predikatskog slova zagrada potiče od terma i postupa se kao pod c).

b) Ako je to ponovo unarna operacija (znak negacije ili otvorena zagrada koja ogradjuje kvantifikator) onda se kraj domena dejstva polazne unarne operacije poklapa sa krajem dejstva ove unarne operacije.

c) Ako je to predikatsko slovo treba naći kraj predikata. Ukoliko se radi o relacijama $=$, $\neq$, $<$, $\leq$, $>$, $\geq$ pri sekvencijalnoj pretrazi nailazimo najpre na promenljivu, otvorenu zagradu koja nije diskutovana pod a) ili b), broj ili konstantu ili funkcijsko slovo. I u tom slučaju tražimo kraj predikata.

2. Nalaženje predikatskog naziva je u slučaju predikata koji počinju slovima P,Q,R,S,T jednostavno jer ta slova ukazuju istovremeno i na početak predikatskog naziva. Naziv se sastoji od tog slova i eventualno sufiksa kao što je ranije opisano. Ako se krećemo udesno po formuli počev od tog slova doći ćemo do kraja predikatskog naziva ako je ispunjen bilo koji od sledećih uslova:

-naišli smo na binarnu logičku operaciju,
-naišli smo na otvorenu zagradu koja uokviruje argumente

predikata,
-stigli smo do kraja formule.

3. **Nalaženje početka i kraja elementarne formule.** Za elementarne formule sa predikatskim nazivima tipa Q,R,S,T posle kraja predikatskog naziva nastavljamo kretanje udesno dodajuci deo formule izmedju zagrada za argumente. Predikati sa početnim slovom P su unarni pa se kraj elementarne formule poklapa sa krajem predikatskog naziva. Za aritmeticke predikate kojima je predikatsko slovo neka od relacija $=$, $\neq$, $<$, $\leq$, $>$, $\geq$ postupa se slično kao za nalaženje domena binarne logičke operacije. Da bi se dobio početak ide se ulevo od posmatranog relacijskog slova dok se ne ispuni neki od sledecih uslova:

a) NMZ=-1,
b) naišli smo na logičku operaciju,
c) stigli smo do početka formule,
d) NMZ=1 i naišli smo na zatvorenu zagradu koja potiče od kvantifikatora.

Ukoliko tražimo kraj domena dejstva postupa se analogno s tim što se slučaj pod d) ne pojavljuje.

4. **Notiranje nepotrebnih zagrada.** Ovde izostavljamo posmatranje zagrada koje uokviruju kvantifikatore, zagrada koje uokviruju argumente predikata kao i zagrade na nivou terma. Par zagrada je nepotreban u datoj formuli ako formula dobijena izostavljanjem tog para zagrada ima istu prezentaciju drveta kao polazna formula.

Nepotrebne zagrade mogu biti sledecih tipova:

a) spoljnje zagrade koje uokviruju celu formulu,

b) višestruke zagrade za koje podformula sadržana u tim zagradama ima spoljnje zagrade,

c) zagrade nad atomarnim formulama tj. u slučaju kad podformula u zagradama ne sadrži nijednu logičku operaciju,

d) zagrade koje su nepotrebne prema uvedenoj konvenciji o prioritetu logičkih operacija.

Spoljnje i višestruke zagrade se lako uočavaju i otklanjaju korišćenjem parametara NZ i NMZ. Pretpostavimo da su one otklonjene i predjimo na opis algoritama za uklanjanje nepotrebnih zagrada tipa c) i d).

Za svaki par zagrada koji je preostao neka IL označava simbol neposredno pre otvorene zagrade. Ukoliko smo na početku formule IL=0. Analogno pretpostavimo da ID ozanačava prvi simbol posle odgovarajuće zatvorene zagrade. Ponovo, ako smo na kraju formule, ID=0. Najzad, označimo sa IS vrhovnu logičku operaciju podformule sadržane unutar posmatranog para zagrada. Ako nijedna operacija nije sadržana unutar zagrada tada je IS=0.Pritom nastupa neka od sledećih mogućnosti:

a) Ako je IS neki od simbola 0, ∀, ∃, ¬ zagrade su suvišne.

b) Ako je IS neki od simbola ∨, ∧, =>, <=> i IL je ¬ ili zatvorena zagrada tada su zagrade potrebne.

c) Ako tačno jedna od IL i IR (na primer IL) predstavlja neki od simbola binarnih logičkih operacija tada uporedjujemo IR sa IS i odlučujemo da li su zagrade suvišne imajući u vidu konvenciju o prioritetu.

d) Ako su IL i IR oba binarne logičke operacije uporedjujemo IS sa IL i IR i na osnovu konvencije o prioritetu zaključujemo da li su zagrade potrebne.

Ovo ukazuje da je osobina zagrada da su suvišne ili da su potrebne lokalnog karaktera, u smislu da zavisi od podformule i to njene vrhovne logičke operacije i dva susedna simbola IL i IR. Takodjer prisustvo drugih nepotrebnih zagrada ne utiče na potrebnost pojedinog para zagrada.

4.2 TRANSFORMACIJA FORMULE KORIŠCENJEM LEMA OBLIKA EKVIVALENCIJE

Pretpostavimo da je data lema $\mathcal{L}$ oblika A<=>B gde su A i B kvantifikatorske formule. Neka je nadalje data formula $\mathcal{F}$ koju ćemo transformisati pomoću leme $\mathcal{L}$. Ukoliko se neka podformula F formule $\mathcal{F}$ poklapa ili se supstitucijom σ može dovesti, do na preimenovanje vezanih promenljivih, do poklapanja sa jednom stranom leme (na primer A) tada možemo podformulu F formule $\mathcal{F}$ zameniti drugom stranom leme (tj. sa B) u kojoj su slobodne promenljive zamenjene u skladu sa supstitucijom σ. (Pod supstitucijom podrazumevamo zamenu svakog pojavljivanja slobodne promenljive sa termom koji je njoj odgovarajući u zamenskoj komponenti date supstitucije. Definicija pojmova će biti data u 4.4.1).

U cilju realizacije ove zamene predstavimo formule F i A pomoću drveta, a zatim od tih drveta generišimo nova drveta T i D kojima se čvorovi na lišću obeležavaju predikatskim slovima iza kojih sledi simbol grafovske operacije ukoliko se on pojavio u sufiksu predikatskog slova odgovarajućeg predikata. Potreban uslov za izvršenje ove zamene jeste poklapanje novo generisanih drveta T i D do na preimenovanje vezanih promenljivih.

Bez obzira što se u suštini radi o poklapanju struktura koje su drveta ovo se može ustanoviti sekvencijalnom pretragom ogoljenih formi dveju formula u kojima su jedino vezane promenljive preimenovane kako bi se poklopile. Ta ogoljena forma dobija se izbacivanjem nepotrebnih zagrada, redukovanjem elementarne formule na predikatski naziv, a iz sufiksa izbacujemo deo sufiksa sa imenom grafa jer on ima ulogu slobodne promenljive. Kvantifikatori se redukuju samo na simbole $\forall$ i $\exists$, a odbacujemo zagrade koje ih uokviruju i vezane promenljive uz njih. Ukoliko se ogoljene forme formula F i A poklapaju ispunjen je prvi potreban uslov za izvršenje transformacije. Nadalje je potrebno utvrditi postojanje supstitucije slobodnih promenljivih formule A sa termima na nivou argumenata odgovarajućih predikata u posmatranoj formuli F.

Neka su $x_1, x_2, \ldots, x_n$ slobodne promenljive formule A. Pretpostavićemo da se svaka od njih javlja sama kao neki argument u bar jednom predikatu u formuli A. . Zatim odredjujemo odgovarajuće terme t_1, $t_2, \ldots, t_n$ u podformuli F tako da im odgovaraju po tipu i poziciji predikata i poziciji argumenata u predikatu. Takodjer izvršimo preimenovanje vezanih promenljivih ukoliko ih ima u formulama F i A kako bi bile odgovarajuće označene istim simbolima. Naravno, pri preimenovanju vodimo računa da koristimo dotle neupotrebljene simbole sa što nižim brojem u indeksu kao i da ne promenimo tip promenljivih koje koristimo (na primer promenljive za čvorove i dalje ostaju istog tipa ali im se eventualno menja indeks).

Budući da su dozvoljeni grafovski nazivi u sufiksima predikata treba pronaći supstituciju σ_1 za grafovska imena jer ona imaju ulogu slobodnih promenljivih.

Najzad primenjujemo supstitucije σ i σ_1 na formulu A. Ukoliko je dobijena formula identična (do na preimenovanje vezanih promenljivih) sa podformulom F, unifikacija je uspela i ispunjen je i dovoljan uslov za transformaciju podformule F pomoću B. Pre no što ćemo to uraditi treba primeniti substitucije σ i σ_1 na formulu B. Ukoliko skup slobodnih promenljivih u B nije podskup skupa promenljivih u A promenljive koje su iz B, a nisu iz A, mogu se izabrati proizvoljno koristeći nove simbole. Najzad podformulu F iz formule $\mathcal{F}$ zamenjujemo sa tako transformisanom formulom B.

Na analogan način može se vršiti i zamena pomoću iste leme zdesna ulevo pri čemu u prethodnom izlaganju A i B menjaju mesto.

Sledi primer i objašnjenje.

Primer 1.

Neka je lema $\mathcal{L}$ oblika ekvivalencije data sledećom kvantifikatorskom formulom:

$\mathcal{L} \equiv$ S2G(X,Y,K+1) <=> $(\exists Z)$(S2G(X,Z,K) $\wedge$ R1G(Z,Y)).

Neka je nadalje data rečenica koju ćemo pomoću te leme transformisati i koja u obliku kvantifikatorske formule izgleda ovako:

$\mathcal{F} \equiv$ R1HA1(Y,X1) $\wedge$ $(\exists X)$(S2HA1(X1,X,1) $\wedge$ R1HA1(X,X2)) => $(\exists L)$S2HA1(Y,X2,L).

Da bismo primenili zamenu po lemi $\mathcal{L}$ zdesna ulevo, najpre se pravi ogoljeni oblik desne strane leme. B ogoljeno je oblika:

$\exists$(S2 $\wedge$ R1).

U kvantifikatorskoj formuli rečenice se redom izdvajaju sve njene podformule. Sa svakom podformulom se proba da li je moguća zamena. Obeležićemo sa F podformulu:

$(\exists X)$(S2HA1(X1,X,1) $\wedge$ R1HA1(X,X2)).

I njen ogoljeni oblik je takodjer

$\exists$(S2 $\wedge$ R1),

što je prvi potreban uslov za mogućnost zamene. Nadalje se utvrdjuje supstitucija σ, a zatim σ_1.

$\sigma \equiv$ { X—>X1, Y—>X2, K—>1}, $\sigma_1 \equiv$ { G—>HA1 }.

Primenom supstitucija σ i σ_1 na B utvrdjujemo podudarnost od $B\sigma\circ\sigma_1$ sa podformulom F do na preimenovanje vezanih promenljivih. Zatim se supstitucije σ i σ_1 primenjuju na suprotnu stranu leme $\mathcal{L}$ tj. na A. Tako dobijamo

S2HA1(X1,X2,1+1),

a zatim možemo formulu F u rečenici zameniti sa tako modifikovanom drugom stranom leme. Da je u $A\sigma\sigma_1$ bilo vezanih promenljivih izvršilo bi se prethodno njihovo preimenovanje kako bi sve bilo u skladu sa konvencijom da su skupovi vezanih i slobodnih promenljivih disjunktni, kao i da su sve vezane uz različite kvantifikatore različite.

Rezultujuća formula je oblika:

R1HA1(Y,X1) $\wedge$ S2HA1(X1,x2,1+1) => $(\exists L)$S2HA1(Y,X2,L).

4.3 TRANSFORMACIJA FORMULE KORIŠĆENJEM VALJANE FORMULE OBLIKA EKVIVALENCIJE I IZVEDENE IZ TAUTOLOGIJE

Pretpostavimo da je $\mathcal{T}$ tautologija oblika A<=>B gde su sada A i B formule iskaznog računa. Pokušavamo da uspostavimo korespodenciju formule A i neke podformule F formule $\mathcal{T}$. Potrebno je da se drvo **unutrašnjih čvorova** (svi čvorovi drveta sem onih na lišću drveta) u drvetu kojim predstavljamo formulu A poklapa sa poddrvetom unutrašnjih čvorova u drvetu kojim predstavljamo formulu F. Pritom čvorovima na lišću drveta formule A dodeljujemo formule kojima je vrh drveta ispod listova u izdvojenom poddrvetu u drvetu formule F. Ukoliko jednakim iskaznim slovima u tom pridruživanju odgovaraju jednake formule (do na preimenovanje vezanih promenljivih) može se izvršiti transformisanje pomoću B.

Pre detaljnijeg opisa transformacije navedimo algoritam za utvrdjivanje da li je drvo T poddrvo drveta D.

1) Pronadjimo sve čvorove drveta D obeležene obeležjem vrhovnog čvora drveta T. Ukoliko takvih čvorova nema T nije poddrvo drveta D.

2) Za svaki od tih čvorova vršimo uporedjivanje poddrveta od D kome je to vrhovni čvor.

3) Ukoliko se vrhovni čvor izdvojenog poddrveta poklapa sa vrhovnim čvorom iz T isprobavamo da li im se poklapaju obeležja kod sinova. Ukoliko to ne uspe T nije poddrvo od D.

4) Metodom sintaksne analiza drveta ("backtracking metoda") to ponavljamo za sve čvorove drveta T. Ukoliko nismo iskočili sa negativnim odgovorom i utvrdili smo pokapanje obeležja svih odgovarajućih čvorova odgovor je pozitivan tj. T je poddrvo drveta D. Ukoliko smo pre kraja iscrpili sve čvorove izdvojenog poddrveta u D postupak se završava sa negativnim

odgovorom za to izdvojeno poddrvo i pokušavamo sa sledećim izdvojenim poddrvetom.

Vratimo se sad objašnjenju primene tautologije $\mathcal{T}$ na formulu $\mathcal{F}$. Obeležimo sa T drvo kojim predstavljamo formulu A. Neka T′ označava redukovano drvo koje se iz drveta T dobija odbacivanjem završnih čvorova. Najzad, obeležimo sa D drvo formule $\mathcal{F}$.

Opisanim algoritmom zatim utvrdjujemo da li je T′ poddrvo drveta D. Ukoliko je odgovor negativan transformacija nije moguća. Ako smo utvrdili da je T′ poddrvo drveta D uočimo podformulu F u $\mathcal{F}$ čije je poddrvo podudarno sa T′ u utapanju T′ u D. Zatim izvršimo pridruživanje $\Delta = (F_1/p_1, F_2/p_2, \ldots, F_n/p_n)$ gde su p_i -ovi redom sva iskazna slova koja se pojavljuju na lišću drveta T, a F_i -ovi su formule u drvetu D kojima vrh drveta u utapanju T′ u D odgovara čvoru p_i.

Dovoljan uslov za izvršenje zamene biće ispunjen ukoliko svakoj klasi jednakih p-ova u pridruživanju Δ odgovaraju formule koje su medjusobno podudarne do na preimenovanje vezanih promenljivih i izbacivanje nepotrebnih zagrada.

U slučaju da ovo nije potvrdjeno transformacija korišćenjem valjane formule nije dozvoljena. Ukoliko postoji podudarnost formula nastavljamo tako što u formuli B zamenimo sva iskazna slova formulama koje im odgovaraju prema datom pridruživanju Δ . Najzad, zamenjujemo podformulu F iz $\mathcal{F}$ sa tako izmenjenom formulom B. Zbog konvencije o različito označenim vezanim promenljivim, ukoliko neko slovo u formuli B ima više pojavljivanja nego u A za te slučajeve treba, pre uvrštavanja odgovarajuće formule F_i , izvršiti preimenovanje vezanih promenljivih u njoj ukoliko postoje.

Na analogan način se vrši zamena pomoću valjane formule oblika ekvivalencije u smeru zdesna ulevo, pri čemu A i B menjaju mesta u gornjem izlaganju.

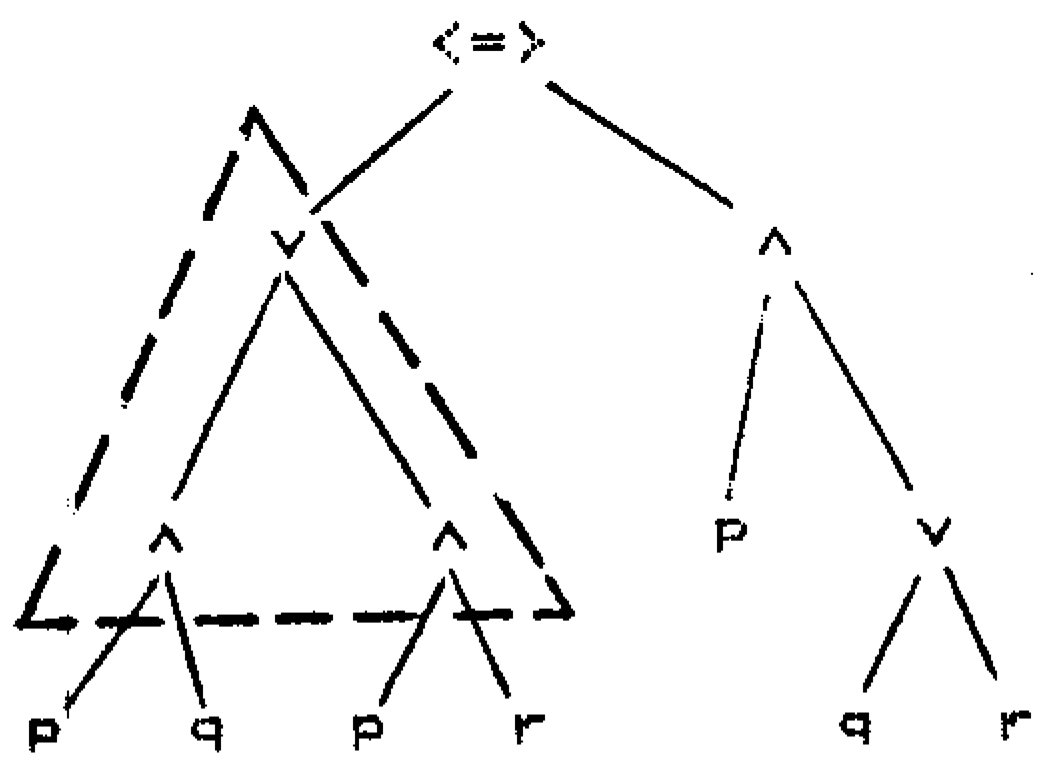
Navedimo primer zamene po valjanoj formuli koji ilustruje navedeno izlaganje.

Primer 2.

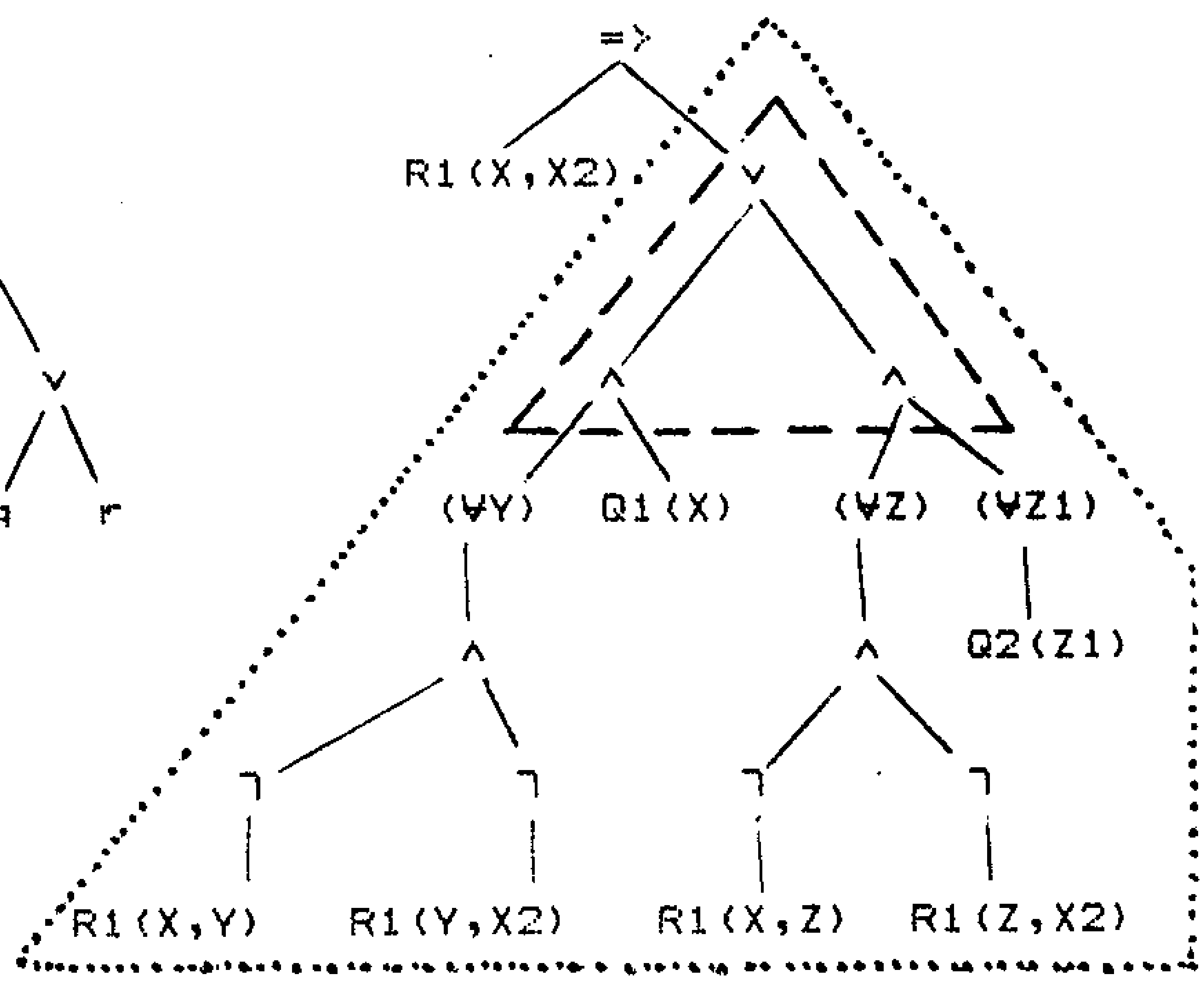
Neka je $\mathcal{T}$ valjana formula oblika $(p \wedge q) \vee (p \wedge r) \Leftrightarrow p \wedge (q \vee r)$ i neka je formula $\mathcal{F}$ data sa:

$R1(X,X2) \Rightarrow ((\forall Y)(\neg R1(X,Y) \wedge \neg R1(Y,X2)) \wedge Q1(X)) \vee (\neg(\forall Z)(\neg R1(X,Z) \wedge \neg R1(Z,X2)) \wedge (\forall Z1)Q2(Z1))$.

Slika 3. prikazuje drvo valjane formule $\mathcal{T}$, a slika 4. predstavlja drvo formule $\mathcal{F}$, koju cemo transformisati koriscenjem valjane formule $\mathcal{T}$ zamenom sleva udesno. Iscrtano je uokvireno poddrvo čiju podudarnost utvrdjujemo, a tačkasto je uokvirena podformula formule $\mathcal{F}$ koju cemo transformisati koristeci desnu stranu valjane formule $\mathcal{T}$.



Slika 3.



Slika 4.

Pridruživanje Δ je dato sa $\{$ $(\forall Y)(\neg R1(X,Y) \wedge \neg R1(Y,X2)/p$,

Q1(X)/q, $(\forall Z)(\neg R1(X,Z) \wedge \neg R1(Z,X2))/p$, $(\forall Z1)Q2(Z1)/r$ }.
Buduci da su crtasto uokvirena poddrveta na slikama 3. i 4. podudarna i kako su jos i formule D_1 i D_2 prideljene slovu p u pridruzivanju Δ podudarne do na preimenovanje vezanih promenljivih, moguce je izvrsiti transformaciju. Po izvrsenoj zameni rezultat je sledeca formula:

$$R1(X,X2) \Rightarrow (\forall Z)(\neg R1(X,Z) \wedge \neg R1(Z,X2)) \wedge (Q1(X) \vee (\forall Z1)Q2(Z1)).$$

4.4 TRANSFORMISANJE KVANTIFIKATORSKE FORMULE U PRENEKS OBLIK

Da bismo mogli da primenimo pravilo rezolucije tekuću formulu predikatskog računa prvog reda transformišemo u skup sastavaka primenom sledećeg algoritma. U implementaciji tog algoritma u sistemu GRAPH korak 1) je preskočen zbog usvojene konvencije o različito označenim promenljivim koje vezuju različiti kvantifikatori.

Algoritam:

1) Vrši se preoznačavanje promenljivih tako da različita pojavljivanja kvantifikatora vazuju različite promenljive.

2) Eliminišemo logičke operacije ekvivalencije i implikacije uzastopnom primenom zamena korišćenjem valjanih formula:

$A \Leftrightarrow B \longrightarrow (A \Rightarrow B) \wedge (B \Rightarrow A)$,

$A \Rightarrow B \longrightarrow \neg A \vee B$.

3) Negacija se pomera ka predikatskim slovima primenom sledećih valjanih formula za zamenu:

$\neg\neg A \longrightarrow A$,

$\neg(A \vee B) \longrightarrow \neg A \wedge \neg B$,

$\neg(A \wedge B) \longrightarrow \neg A \vee \neg B$,

$\neg(\forall x)A \longrightarrow (\exists x)\neg A$,

$\neg(\exists x)A \longrightarrow (\forall x)\neg A$.

4) Formula se prevodi u preneksnu formu tako da svi kvantifikatori prethode matrici formule gde matrica ne sadrži kvantifikatore. To se postiže primenom sledećih zamena (u kojima simboli K i Q označavaju znak bilo univerzalnog bilo egzistencijalnog kvantifikatora):

$(Kx)\ A(x) \wedge B \longrightarrow (Kx)(A(x) \wedge B)$,

$(Kx)\ A(x) \vee B \longrightarrow (Kx)(A(x) \vee B)$,

(U navedenim formulama x ne ulazi u B.)

$$(Kx)A(x) \wedge (Qy)B(y) \longrightarrow (Kx)(Qy)(A(x) \wedge B(y)),$$

$$(Kx)A(x) \vee (Qy)B(y) \longrightarrow (Kx)(Qy)(A(x) \vee B(y)).$$

5) Vrši se skolemizacija preneksne forme.
a) Ako su svi kvantifikatori univerzalni tj. ako je formula oblika $(\forall x_1)\ldots(\forall x_n)\ A$, gde A ne sadrži kvantifikatore onda je skolemizirana formula oblika A. (Pritom x_1, $x_2,\ldots,x_n$ označavaju promenljive kojima sufiks ima ulogu indeksa.)
b) Ako je formula oblika $(\forall x_1)(\forall x_2)\ldots(\forall x_n)(\exists y)\ B(y)$ za $n>0$ tada uvodimo nov n-arni funkcijski simbol f (specjalno, za $n=0$, f je nova konstanta) i rezultujuca formula je $(\forall x1)(\forall x2)\ldots(\forall xn)B(f(x_1,x_2,\ldots,x_n))$ sa jednim egzistencijalnim kvantifikatorom manje. Postupak ponavljamo sve dok sve egzistencijalne kvantifikatore ne zamenimo Skolemovim funkcijama, a zatim primenimo pravilo navedeno pod a).

6) Skolemiziranu formulu dovodimo na konjunktivnu normalnu formu primenom transformacija:

$$A \vee (B \wedge C) \longrightarrow (A \vee B) \wedge (A \vee C),$$

$$(A \wedge B) \vee C \longrightarrow (A \vee C) \wedge (B \vee C).$$

7) Najzad eliminišemo simbole konjunkcije zamenom formule oblika $A \wedge B$ sa dve formule A, B.

Dakle primenom koraka 1) do 7) u gornjem algoritmu proizvoljnu formulu prevodimo u skup sastavaka i pritom je svaki sastavak disjunkcija konačnog broja literala tj. predikat ili negacija predikata.

4.4.1 PRAVILO REZOLUCIJE

Definišimo najpre pravilo rezolucije za fundamentalne sastavke tj. sastavke čiji literali ne sadrže promenljive.

Definicija 1. Za sastavke C i D gde je C oblika

$A \vee C_1 \vee C_2 \vee \ldots \vee C_n$, a D je oblika

$\neg A \vee D_1 \vee D_2 \vee \ldots \vee D_m$ definišemo **rezolventu** R

$$R \equiv C_1 \vee C_2 \vee \ldots \vee C_n \vee D_1 \vee D_2 \vee \ldots \vee D_m.$$

Pritom, naravno, literali A i $\neg A$ mogu da se nadju na proizvoljnom mestu u n-arnoj odnosno m-arnoj disjunkciji.

Na primer, za sastavke Q(a) v R(b,a) i $\neg$Q(a) v R1(f(a,b),b) rezolventa je R(b,a) v R1(f(A,b),b).

Da bismo definisali rezolventu za proizvoljne sastavke uvešćemo pojam zamene ili supstitucije i unifikacije.

Definicija 2. Zapis oblika $x \to t$ ili t/x nazivamo **zamenskom komponentom.** Pritom t označava term koji ne sadrži x a x je promenljiva.

Definicija 3. **Zamena ili substitucija** Θ je konačan skup zamenskih komponenti $\Theta \equiv \{ x_1 \to t_1, x_2 \to t_2, \ldots x_n \to t_n \}$ ili drugačije zapisano $\Theta \equiv \{ t_1/x_1, t_2/x_2, \ldots, t_n/x_n \}$ gde je $x_i \neq x_j$ za $i \neq j$.

Definicija 4. **Θ primer** formule F u oznaci $F\Theta$ dobijamo zamenom svakog pojavljivanja x_i u formuli F termom t_i gde je $t_i/x_i \in \Theta$.

Definicija 5. **Kompozicija substitucija** Θ i σ u oznaci $\Theta \circ \sigma$ ili naprosto $\Theta\sigma$ za $\Theta \equiv \{ t_1/x_1, \ldots, t_n/x_n \}$ je skup $\Theta_1 \cup \sigma_1$ gde je $\Theta_1 = \{ t_1/x_1, \ldots, t_n/x_n \} - \{ x_1/x_1, \ldots, x_n/x_n \}$ i $\sigma_1 = \sigma - \{ u/x \mid t/x \in \Theta \}$.

Definicija 6. Zamena Θ je **unifikator** skupa C gde je C skup terma ili skup literala ako je $C\theta$ jednočlan skup.

Definicija 7. Unifikator θ je **najopstiji unifikator** skupa literala A ako za proizvoljan unifikator μ skupa A postoji zamena σ takva da je $\mu = \theta \circ \sigma$.

Definišimo najzad pravilo rezolucije za proizvoljne sastavke.

Definicija 8. Za sastavke C i D koji ne sadrže zajedničke promenljive i $A \subseteq C$, a $B \subseteq D$ i pritom važi:

1) postoji najopstiji unifikator θ za skup $A \cup B$,
2) jednočlani skupovi $A\theta$ i $B\theta$ obrazuju komplementaran par.

Rezolventu satavaka C i D definišemo kao sastavak

$$(C - A)\theta \;\cup\; (D - B)\theta.$$

Definicija 9. Neka su C i D sastavci i R njihova rezolventa, tada pravilo izvodjenja

$$\frac{C\ ,\ D}{R}$$

nazivamo **pravilom rezolucije.**

Definicija 10. **Opovrgavanje** datog skupa sastavaka S je konačan niz sastavaka $B_1, B_2, \ldots, B_n$ takav da za svaki član B_i, $1 \le i \le n$ važi:

1) $B_i \in S$ ili je B_i rezolventa neka dva predhodna člana niza,
2) B_n je prazan sastavak.

Kao što se vidi opovrgavanje datog skupa sastavaka S je izvodjenje praznog sastavka iz skupa S, koriscenjem pravila rezolucije kao jedinog pravila izvodjenja.

Da bismo dokazali da je formula F teorema procesom opovrgavanja, skup S formiramo od sastavaka dobijenih transformisanjem u preneks oblik aksioma date teorije, eventualno nekih definicija i lema kao i negacije zatvorene formule F.

Rezolucija je revolucionarno pravilo izvodjenja koje je kompletno. To znači da teorijski uz pretpostavku neograničenih prostorno vremenskih resursa možemo izvesti dokaz svake formule koja jeste teorema.

Cena jednostavnosti pravila u praktičnoj realizaciji se plaća time što na svakom koraku imamo više sastavaka no što je potrebno za dokaz, te vrlo brzo, i pri osrednje teškim teoremama dolazi do prekoračenja resursa. Dobar deo istraživanja koja u dokazivanju koriste rezoluciju i razne njene restrikcije odnose se na uvodjenje dodatnih heuristika odnosno neke inteligentne strategije u organizaciji pretrage kao što je opisano u odeljku 2.

U realizaciji modula koji dokazuje teoreme primenom rezolucije kao pravila izvodjenja u sistemu GRAPH ja sam implementirala algoritam za prevodjenje formule na oblik sastavaka, dok je kolega Hotomski implementirao proceduru opovrgavanja.

U ovom dokazivaču korisniku se takodjer daje mogućnost da koristi i indukciju kao pravilo izvodjenja o čemu se odlučuje pre početka rada.

U cilju povećanja šansi za nalaženje dokaza korisnik može, iz skupa definicija i lema koje mu sistem prikazuje jednu po jednu kao eventualno potrebne za izvodjenje dokaza, da odabere neke, koje će po njegovoj proceni ubrzati dokaz. Ovo u praksi znači da korisnik najčešće treba da izvede dokaz da bi to uradio na zadovoljavajući način, što je prilično nepovoljno za čoveka, s obzirom, na njemu neprirodan oblik formula sa kojima se dokaz izvodi.

Redosled sastavaka se takodjer može menjati u interaktivnom radu ukoliko dokaz ne uspe pod datim prostorno vremenskim ograničenjima, ali se i tu pojavljuje problem komunikacije korisnika sa sastavcima koji su daleko od formula koje čovek koristi u dokazivanju.

Stoga je interaktivni dokazivač u sistemu GRAPH, čiji je jedan deo i pomenuti moduo, koncipiran tako da prirodnim izvodjenjem koje sledi način kako ljudi dokazuju teoreme razvije drvo dokaza tako da dokaz svih formula na lišću drveta obezbedjuje dokaz polazne teoreme koja je vrh drveta dokaza. Data formula se razbija na jednostavnije podciljeve primenom raznih izvodjenja koje ćemo navesti i objasniti u narednom odeljku uz zadržavanje prirodnog oblika formule u celom drvetu dokaza. S obzirom na veliki broj mogućnosti za izvodjenje sinova za svaki podcilj, u slučaju kada korisnik vodi akciju, on bira prema raspoloživim komandama i sopstvenoj intuiciji, a u automatskom radu koristimo ugradjene heuristike i ugradjenu inteligentnu strategiju vodjenja dokaza. U odredjenom momentu, u slučaju da ugradjenim modulom nije potvrdjena tačnost nekog podcilja na lišću drveta dokaza, poziva se i dokazivač zasnovan na primeni rezolucije koji sada ima veće šanse da uspe jer se primenjuje na podcilj koji je jednostavniji.

5. INTERAKTIVNI DOKAZIVAČ TEOREMA SA HEURISTIČKIM PRISTUPOM I PRIRODNIM IZVODJENJEM U SISTEMU GRAPH

Ideja korišćenja ljudske intuicije u sprezi sa moćnim programskim sistemom, u cilju usmeravanja izvodjenja u dokazivanju teorema, javila se ubrzo nakon što se uvidelo da kompleksnost dokazivanja teorema u predikatskom računu prvog reda prevazilazi moć dosad razvijenih dokazivača, bilo da su oni rezolucijskog ili nerezolucijskog tipa. Slično kao i sistemi opisani u <1>, <7>, <41> i ovaj dokazivač teorema, koji je sastavni deo sistema GRAPH, je koncipiran tako da u interaktivnom radu korisnika i sistema generiše dokaz. U njemu postoje tri mogućnosti za izbor nivoa interaktivnog rada i o tome će biti više rečeno u daljem izlaganju.

Dokazivač je radjen u duhu radova Newell-a, Shaw-a i Simona <63>, Gelerntera <38>, <39>, Bledsoe-a <5>, <7>, <8>, <9>, Brown-a <13> i Pastre-a <65>. Sledjena je ideja o izvodjenju koje će biti blisko i prirodno za čoveka, kako bi on u interaktivnom radu lakše učestvovao u izvodjenju dokaza s jedne strane, i kako bi se na bazi eksperimenata sa dokazivačem na konkretnim teoremama iz oblasti teorije grafova lako formulisale dodatne heuristike u usmeravanju koraka u izvodjenju dokaza u cilju povećanja efektivnosti i moći dokazivača.

Nakon što je korisnik saopštio sistemu GRAPH rečenicu na engleskom jeziku (odnosno jeziku pravi GTCL), na primer, pod imenom P, a sistem je preveo na nivo predikatske formule, možemo započeti interaktivni dokaz koristeći komandu:

CREATE [TREE]<tree name> PROOF [OF[THE[SENTENCE]]]<sentence name> .

Sistem zatim generiše drvo dokaza ulazne rečenice sa zadatim imenom pod imenom datim u komandi . Dokazivanje se izvodi na nivou kvantifikatorske formule tako što sistem polaznu rečenicu P prevede na oblik kvantifikatorske formule i ona postaje polazni cilj u drvetu dokaza. Sistem zatim taj cilj zamenjuje jednostavnijim koristeći ekvivalentne transformacije za generisanje podciljeva ili ga razbija na dva nova podcilja, a zatim nastavlja sa sinovima isti postupak kreiranja podciljeva, sve dok podciljevi na lišću drveta dokaza ne postanu dovoljno prosti da se mogu dokazati ugradjenim rutinama. Razbijanje na podciljeve se vrši tako da konjunkcija podciljeva sinova u drvetu dokaza implicira podcilj koji im je otac u drvetu dokaza. Naravno, dokaz je završen kad sistem dokaže sve podciljeve na lišću drveta dokaza.

Drvo dokaza je korensko drvo, pri čemu tekući podcilj predstavlja koren drveta. Korisnik može da pomera koren drveta dokaza na željeni podcilj, a može i da uništi deo kreiranog drveta dokaza ukoliko konstatuje da neki od učinjenih koraka nije bio dobro izabran.

U izvodjenju nema neprirodnih transformacija kao što je eliminacija implikacije i ekvivalencije u postupku dokazivanja koji koristi rezoluciju kao pravilo izvodjenja. Rečenica se zadržava u svom prirodnom obliku kako bi korisnik u svakom momentu mogao lako da prati dokaz i lako da usmeri proces u željenom pravcu ukoliko zatreba.

Radi se unazad, od teoreme koju treba dokazati, ka podciljevima za koje sistem može da potvrdi tačnost korišćenjem ugradjenih modula. Ideja je da se ne ide sve do aksioma, već da sistem koristeći ugradjeno znanje smešteno u datotekama definicija, lema i valjanih formula, pomoću ugradjenih heuristika kojima se usmerava i vodi akcija, kompletira dokaz na višem nivou.

Podcilj se može po potrebi, na inicijativu korisnika, ili samog sistema, poslati na dokazivač zasnovan na rezoluciji, koji, sem eventualno izbora relevantnih aksioma, definicija i lema, radi bez učešća čoveka. Naravno u tom slučaju prethodi priprema formule: prevodjenje u preneks oblik i skolemizacija kao što je opisano u odeljku 4.4.

Koncepcija rada dokazivača koji koristi rezoluciju opisana je u <33>, <44> i <45>, dok je predmet ovog izlaganja koncepcija dokazivača koji sledi simuliranje čovekovog načina i pristupa u dokazivanju teorema.

Postoje tri režima rada ovog dokazivača kojima korisnik bira stepen interaktivnosti. Komanda glasi:

$$\text{SET MODE OF PROVING TO} \begin{Bmatrix} 0 \\ 1 \\ 2 \end{Bmatrix}.$$

Ukoliko je korisnik izabrao obeležje 0 sistem prelazi na režim rada u kome svaki korak bira čovek. Na raspolaganju su mu pojedine komande čiji opis i delovanje slede u narednom odeljku. To je oblik interaktivnog rada gde računar više služi kao sistem za proveru dokaza neke teoreme, jer je čovek taj koji smišlja i vodi akciju, korak po korak, po svom nahodjenju i raspoloživim komandama. Naravno prednost ovog načina rada saradnje čoveka i računara jeste u tome što računar ne pravi omaške u radu, kao što se to ljudima često dešava, a mana je, naravno, što čovek mora mnogo da se angažuje da bi kompletirao dokaz teoreme.

Ukoliko korisnik odabere obeležje 1 dokazivač prelazi na interaktivni rad sa smanjenim učešćem čoveka u vodjenju dokaza. To je oblik rada gde sistem koristi ugradjene heuristike da bi usmerio i samostalno kreirao pojedine delove dokaza, a korisnik, po potrebi, eventualno preusmerava dokaz. Ukoliko ništa nije specificirano sistem podrazumeva da je režim rada sa obeležjem 1.

Predjimo sada na opis koncepcije rada dokazivača u opštim crtama pod pretpostavkom da je režim rada zadat obeležjem 1. Rad sistema odvija se na sledeci način:

1) Sekvencijalno se izvršavaju takozvane "trivijalne transformacije" i generišu na taj način novi podciljevi sve dok bar jedna od transformacija sa spiska može da se primeni na ma koji od podciljeva na lišću drveta dokaza. (Pojam i spisak trivijalnih transformacija će biti naveden na kraju

ove glave u odeljku 5.0.1)

2) Za svaki od podciljeva na lišću drveta dokaza ispituje se, korišćenjem rutina za utvrdjivanje tačnosti podcilja, da li je on tačan. Rutine za utvrdjivanje tačnosti koriste razne logički valjane formule, a biće opisane na kraju ove glave u odeljku 5.0.2 detaljnije.

3) Utvrdjuje se kompleksnost kvantifikatorske formule tekućeg podcilja. Kompleksnost predstavlja brojevnu vrednost heurističke funkcije, koja je uvedena da bi se usmerio dokaz u pravcu povoljnih transformacija podcilja. Detalji heuristike i način evaluacije funkcije će biti detaljno opisani u odeljku 8.1.

Ukoliko primenom koraka 1) i 2) za dati podcilj nije više moguće generisati nove podciljeve i nije potvrdjena tačnost tekućeg podcilja ugradjenim rutinama, a kompleksnost njegove kvantifikatorske formule je različita od 0, tada sistem pokušava izvršenje jedne "netrivijalne transformacije". Pod netrivijalnom transformacijom podrazumevamo generisanje novog podcilja, koji dobijamo korišćenjem zamene jedne podformule date formule tekućeg podcilja, pomoću neke definicije ili leme. Budući da tu postoji veliki izbor mogućnosti, kako sa aspekta brojnosti datoteka definicija i lema, tako i izbora podformula date formule, ugradjena je heuristika kojom biramo relevantnu definiciju odnosno lemu koju ćemo upotrebiti za zamenu. Sistem proba razne varijante do odredjene dubine, pridељujući svakoj kvantitativna obeležja na osnovu kompleksnosti podciljeva na lišću, generisanih posle primene 1) i 2) na novi podcilj. Bira se naravno najpovoljnija varijanta sa gledišta ocene kompleksnosti.

Po generisanju novog podcilja korišćenjem netrivijalnih transformacija vraćamo se na korak 1).

Kada postignemo da kompleksnost formule u tekućem podcilju bude 0, a koracima 1) i 2) ne dobijamo nove podciljeve, niti je potvrdjena tačnost tekućeg podcilja, sistem aktivira dokazivač zasnovan na rezoluciji za taj podcilj. Ukoliko dokaz ni tada ne uspe dalja inicijativa se prepušta korisniku.

Dokaz je kompletiran ukoliko sistem dokaže sve podciljeve na lišću drveta dokaza.

Kada se korisnik opredeli za treći oblik interaktivnog rada dokazivača, sa obeležjem **2**, tada je vodjenje dokaza još više automatizovano. Naime, sada se nastavlja sa izvodjenjem netrivijalnih transformacija u dubinu, sve dok ima podciljeva na lišću za koje to možemo da uradimo.

Kada postignemo da je kompleksnost formule 0, a koracima 1) i 2) ne dobijamo nove podciljeve, niti je potvrdjena tačnost tekućeg podcilja sistem aktivira za taj podcilj dokazivač zasnovan na rezoluciji.

Ukoliko dokaz ne možemo kompletirati, režim rada se automatski smanjuje na niži nivo tj. vrednost mu postaje 1, ukoliko je prethodno bila 2. Time se dalja inicijativa prepušta korisniku.

Predjimo sada na opise trivijalnih transformacija kao i bloka za utvrdjivanje tačnosti podcilja.

5.0.1 Trivijalne transformacije

Trivjalne transformacije su one kojima se generiše bilo jedan ili dva nova podcilja, ali tako da su oni jednostavniji za dalji dokaz. Dakle, tu su one transformacije koje je uvek korisno primeniti u ciju pojednostavljenja kreiranja daljeg dokaza. Primenjujemo ih, prema sledećem redosledu, na svaki novo generisani podcilj sve dok bar jedna od navedenih transformacija može da se primeni.

1. **Razbijanje na podciljeve.** Podcilj se razbija na dva nova jednostavnija podcilja ukoliko mu je vrhovna logička

operacija konjunkcija ili ekvivalencija, ili ako je oblika implikacije, a sleve strane implikacije je vrhovna logička operacija disjunkcija ili implikacija, ili je sa desne strane implikacije vrhovna logička operacija konjunkcija. Razbijanje je ilustrovano sledecom tabelom:

Tekući podcilj	Novi podciljevi	
$A \wedge B$	A,	B,
$A \Leftrightarrow B$	$A \Rightarrow B$,	$B \Rightarrow A$,
$A \vee B \Rightarrow C$	$A \Rightarrow C$,	$B \Rightarrow C$,
$A \Rightarrow B \wedge C$	$A \Rightarrow B$,	$A \Rightarrow C$,
$(A \Rightarrow B) \Rightarrow C$	$\neg A \Rightarrow C$,	$B \Rightarrow C$.

Ovde su uglavnom sledjene ideje i heuristike sličnih dokazivača razvijenih u svetu, vidi Bledsoe <5>, Brown <14>, Pastre <65>.

2. **Eliminisanje dvojne negacije.** Primenjuje se sekvencijalno transformisanje datog podcilja koriscenjem valjane formule $\neg\neg A \Leftrightarrow A$ u smeru sleva u desno.

3. **Eliminisanje nepotrebnih kvantifikatora.** Ukoliko neka podformula, kojoj je bilo egzistencijalni ili univerzalni kvantifikator vrhovna logička operacija u podformuli, ne sadrži slobodna pojavljivanja promenljive vezane tim kvantifikatorom, on se izbacuje zajedno sa promenljivom uz njega.

4. **Pomeranje negacije iza kvantifikatora.** Negacija se pomera udesno kroz kvantifikatore koriscenjem transformacija

$$\neg(\exists x)F(x) \Leftrightarrow (\forall x)\neg F(x),$$
$$\neg(\forall x)F(x) \Leftrightarrow (\exists x)\neg F(x).$$

Ove transformacije takodjer primenjujemo sve dok može.

5. **Izbacivanje nekih kvantifikatora.** Ukoliko je u formuli vrhovni univerzalni kvantifikator, on se izbacuje. Za podcilj kome je implikacija vrhovna logička operacija izbacuje se univerzalni kvantifikator koji je vrhovna logička operacija na levoj strani implikacije kao i egzistencijalni kvantifikator koji je vrhovni na levoj strani implikacije. I ove transformacije primenjujemo sve dok može.

6. **Pojednostavljenja za aritmetičke relacije.** Ovde se literali oblika $\neg\, t_1 \neq t_2$ zamenjuju sa $t_1 = t_2$ (gde su t, t_1, t_2 proizvoljni termi) i analogno

$\neg\, t1 < t2 \longrightarrow t1 \geq t_2$,

$\neg\, t1 > t_2 \longrightarrow t_1 \leq t_2$,

$\neg\, t_1 \leq t_2 \longrightarrow t_1 > t_2$,

$\neg\, t_1 \geq t_2 \longrightarrow t_1 < t_2$.

Takodjer, imajući u vidu da su promenljive definisane nad skupom nenegativnih brojeva vrše se i sledeće zamene:

$t<1 \longrightarrow t=0$,

$t\leq 0 \longrightarrow t=0$,

$t\neq 0 \longrightarrow t>0$.

7. **Jednakosna supstitucija.** Ako je neki od konjunkta u levoj strani implikacije podcilja oblika x=y, tada se podcilj zamenjuje novim u kome se taj konjunkt izbacuje, a ostali konjunkti u levoj strani implikacije, kao i desna strana implikacije podcilja, se transformišu tako što se sva pojavljivanja x zamene sa y. (Pretpostavljamo da je x promenljiva, a y može biti term, promenljiva ili konstanta.)

8. **Pojednostavljenja na nivou terma.** Ova pojednostavljenja se odnose na nivo terma koji su argumenti predikata. Ako podterm terma ne sadrži promenljive već se sastoji samo od prirodnih brojeva i operacija + i *, izračunava se broje-

vna vrednost tog podterma i zatim podterm zameni njome.

Slično kao kod demodulacije opisane u <86> vrše se i sledece zamene: $x+0 \to x$, $x*0 \to 0$ itd.

9. **Utvrdjivanje istinitosti aritmetickih relacija za brojeve.** Za podformule koje se svode na predikat koji je aritmetička relacija iz skupa $\{=, \neq, <, >, \leq, \geq\}$, a argumenti su brojevi, utvrdjuje se istinosna vrednost tj. predikat se zamenjuje konstantom $\perp$ ili $\top$. Na podcilj se zatim primene pojednostavljenja koriscenjem sledecih tautologija sleva u desno u cilju izbacivanja konstante $\perp$, odnosno $\top$.

$\top \wedge F \to F$, $\qquad \perp \wedge F \to \perp$,

$\top \vee F \to \top$, $\qquad \perp \vee F \to F$,

$\top \Rightarrow F \to F$, $\qquad \perp \Rightarrow F \to \top$,

$F \Rightarrow \top \to \top$, $\qquad F \Rightarrow \perp \to \neg F$.

10. **Pomeranje negacije kroz $\wedge$ i $\vee$ ka podformulama.** Primenom zamene po valjanim formulama izvode se sledece zamene sleva u desno:

$\neg(A \wedge B) \longrightarrow \neg A \vee \neg B$,

$\neg(A \vee B) \longrightarrow \neg A \wedge \neg B$.

11. **Redukovanje negacija.** Koriscenjem valjanih formula transformišemo formulu pomocu:

$\neg A \Rightarrow \neg B \longrightarrow B \Rightarrow A$.

5.0.2 Utvrdjivanje tačnosti podcilja

Utvrdjivanje tačnosti podcilja ispituje se u bloku sa sledecim opcijama:

Pretpostavimo da je dati podcilj za koji ispitujemo tačnost oblika $A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow B_1 \vee \ldots \vee B_m$. U specijalnom slučaju kad su indeksi n, m = 1 radi se o proizvoljnom podcilju oblika implikacije. Kako su konjunkcija i disjunkcija u sistemu uvedene kao binarne operacije sistem najpre izdvaja maksimalnu n-arnu vrhovnu konjunkciju u pretpostavci i maksimalnu m-arnu vrhovnu disjunkciju u zaključku. Zatim drvo transformiše tako da formula sadrži minimalan broj zagrada s obzirom na prioritet logičkih operacija koristeći komutativnost i asocijativnost operacija konjunkcije i disjunkcije. Time se postiže lako izdvajanje po jednog konjunkta iz pretpostavke i jednog disjunkta u zaključku. Naravno, u specijalnom slučaju kad su n,m=1 izdvajamo celu pretpostavku i zaključak.

Za svaki par tako izdvojenih podformula A_i i B_j ispitujemo da li je formula $A_i \Rightarrow B_j$ nekog od sledećih oblika:

1) $F \Rightarrow F$
2) $(\forall x)F(x) \Rightarrow (\exists y)F(y)$
3) $(\forall x)F(x) \Rightarrow F(t)$
4) $(\forall x)F(x) \Rightarrow F(t) \wedge F(h)$
5) $(\forall x)F(x) \Rightarrow F(O)$
6) $(\forall x)F(x) \Rightarrow F(O) \wedge F(D)$
7) $F(t) \Rightarrow (\exists x)F(x)$
8) $F(t) \vee F(h) \Rightarrow (\exists x)F(x)$
9) $F(O) \Rightarrow (\exists x)F(x)$
10) $F(O) \vee F(D) \Rightarrow (\exists x)F(x)$

Pritom F predstavlja proizvoljnu formulu, t i h označavaju proizvoljne terme, a O i D označavaju osnovne terme (termi bez promenljivih).

Takodjer, ukoliko smo blokom trivijalnih transformacija pomoću 9. podcilj redukovali na T on se proglašava tačnim.

U slučaju da je sistem utvrdio tačnost pozivanjem ovog bloka uz podcilj se štampa komentar "Obviously true".

Tačnost nekog podcilja može se utvrditi i korišćenjem lema smeštenih u datoteku lema, pa čak i dokazanim podciljevima u stablu dokaza. Čak i kreirani podciljevi u drvetu dokaza mogu služiti za dokazivanje nekog podcilja, ali naravno, u tom slučaju ostaje da se kompletira dokaz podcilja na koji se pozivamo.

Korisniku je takodjer data mogućnost da neki podcilj proglasi tačnim. U tom slučaju sistem postavlja pitanje zašto je tako i očekuje kratak komentar. Pri ispisivanju drveta dokaza štampa se korisnikov komentar uz podcilj na koji se komentar odnosi.

Kao što je ranije navedeno, u slučaju da ovim blokom nije utvrdjena tačnost podcilja, sistem može pokušati dokaz blokom koji koristi rezoluciju kao pravilo izvodjenja. U slučaju pozitivnog ishoda tj. kompletiranog dokaza štampa se komentar "Proved by resolution".

5.1 OPIS KOMANDI U INTERAKTIVNOM DOKAZIVANJU TEOREMA U SISTEMU GRAPH

Predjimo sada na opis komandi koje su na raspoloženju korisniku u interaktivnom radu sa dokazivačem.

Sintaksom pisanja komandi predvidjeno je da se svaka komanda završava sa pauzom i tačkom tj. sa " .".

1. **Razbijanje na podciljeve.** Ovaj blok razbija tekući podcilj u drvetu dokaza oblika $A \wedge B$ na dva podcilja A i B. To se postiže komandom

FIND SUBGOALS [OF[THE[ROOT]]][OF]<tree name> .

Sem slučaja kad je konjunkcija vrhovna logička operacija ovde su uključeni i slučajevi kada ekvivalentnim logičkim transformacijama dati podcilj možemo transformisati u oblik gde konjunkcija postaje vrhovna logička operacija. Na sličan način kao što je opisano u literaturi na primer u <7>, <65> to uključuje sledeće:

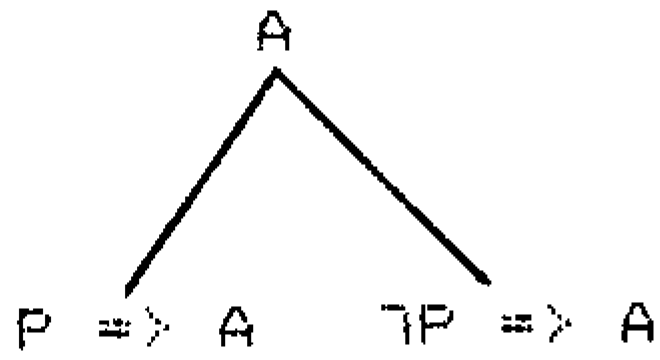
$A \Leftrightarrow B$	$A \Rightarrow B \wedge C$	$A \vee B \Rightarrow C$
$A \Rightarrow B$ $B \Rightarrow A$	$A \Rightarrow B$ $A \Rightarrow C$	$A \Rightarrow C$ $B \Rightarrow C$

Ukoliko je podcilj već dokazan ili ima sinove u stablu dokaza, ovo razbijanje se neće izvršiti.

2. **Analiza slučajeva.** Komandom

MODIFY [THE[ROOT]][OF]<tree name>BY CASE<sentence name> .

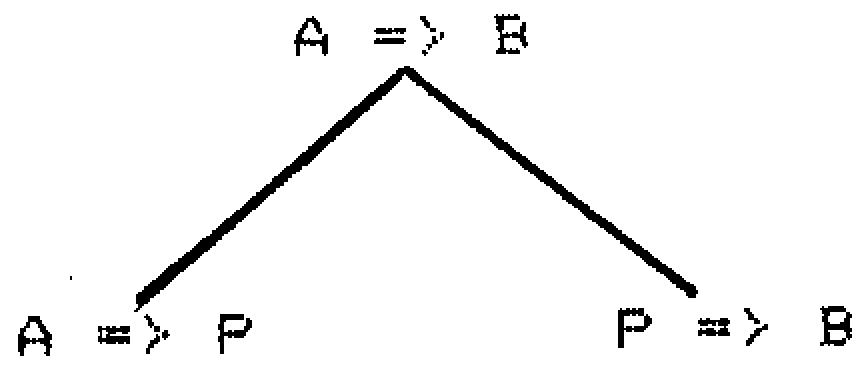
dati podcilj A razbija se na dva nova podcilja uz pomoć rečenice koju zadaje korisnik. Razbijanje analizom slučajeva ilustruje sledeća shema:



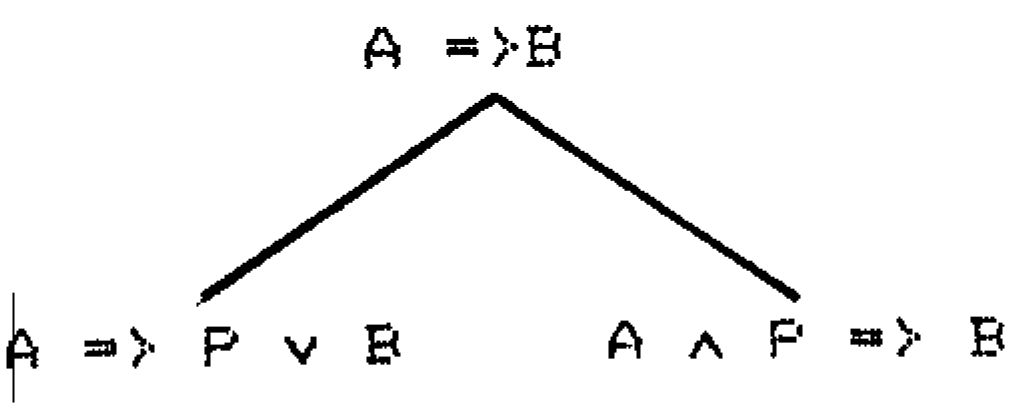
3. **Razbijanje pomoću medjucilja.** Tekući podcilj oblika implikacije, recimo A => B razbija se na dva nova komandom:

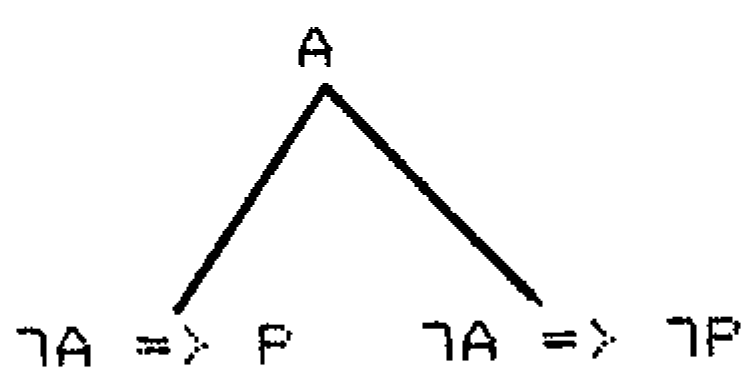
MODIFY[THE[ROOT]][OF]<tree name>BY TRANSITION [GOAL]<sentence name> .

Rezultat su novi podciljevi A => P i P => B gde rečenicu P zadaje korisnik kao što ilustruje shema:



4. **Reductio ad absurdum.** Za podciljeve oblika implikacije A => B vrši se razbijanje na A => P ∨ B i A ∧ P => B gde rečenicu P zadaje korisnik. Ukoliko polazna rečenica nije oblika implikacije već je prosto A tada se razbijanjem na podciljeve uz pomoć P dobijaju podciljevi ¬A => P i ¬A => ¬P kao što ilustruje prikaz:





5. **Pojednostavljenje podcilja.** Tekuci podcilj oblika A => B se pojednostavljuje izostavljanjem pretpostavke A a dalje se radi sa zaključkom B. Formula A se smešta u privremenu datoteku i može se dodati kao konjunkt hipotezi nekog od podciljeva smeštenih u drvetu dokaza u poddrevetu kome je podcilj A => B vrh.

Ovo se postiže komandom:

SIMPLIFY [THE[ROOT]][OF]<tree name> .

6. **Proširenje podcilja.** Suprotno od opisanog pod 5. ovde se formula iz privremene datoteke dodaje kao konjunkt pretpostavci tekuceg podcilja pod gore navedenim uslovima.

7. **Modifikacija podcilja pomocu definicija.**
Komandom

MODIFY [THE[ROOT]][OF]<tree name> BY DEFINITION .

se tekući podcilj transformiše u novi njemu ekvivalentan tako što se jedno pojavljivanje nekog predikata zameni svojom definicijom koja je smeštena u datoteci definicija. Po učitavanju ove komande sistem se obraća korisniku pitanjem koji predikat treba zameniti po definiciji. Odgovor korisnika treba da bude redni broj željenog predikata u datom podcilju i nakon toga sledi zamena odnosno generisanje novog podcilja.

Ovde je uključeno i korišćenje analogije gde za datu definiciju sistem sam po potrebi generiše analognu definiciju u kojoj svi predikati nose odredjeni grafovski nastavak, ukoliko u rečenici predikat koji želimo da zamenimo po definiciji, nosi grafovski nastavak.

8. **Modifikacija podcilja pomocu leme.**
Komandom

MODIFY [THE[ROOT]][OF]<tree name> BY LEMMAS .

postiže se generisanje novog podcilja modifikacijom u kojoj se koriste grafovske leme. Koristeći referentne brojeve koji ukazuju na predikate koji se pojavljuju u podcilju sistem izdvaja iz datoteke lema samo one koje sadrže bar jedan od tih predikata. Leme koje su eventualno relevantne za transformisanje, a nalaze se u tako izdvojenom podskupu, prikazuju se jedna po jedna u obliku kvantifikatorske formule, a korisnik daje odgovor da li želi da se tekući podcilj zameni pomoću tekuće prtikazane leme ili ne. U slučaju pozitivnog odgovora, sistem ispituje da li su ispunjeni potrebni uslovi za zamenu (kao što je detaljno opisano u odeljku 4.2), pa ukoliko je transformacija moguća, sistem je izvršava čime u stablu dokaza dobijamo novi podcilj. U protivnom, nastavlja se sa prikazom lema sa kojima je eventualno moguće izvršiti zamenu podcilja.

9. **Modifikacija podcilja pomoću valjanih formula.**

Komandom

MODIFY [THE[ROOT]][OF]<tree name> BY VALID FORMULAS .

modifikuje se dati podcilj pomoću valjanih formula. Sistem prikazuje korisniku jednu po jednu valjanu formulu iz postojeće datoteke valjanih formula, a korisnik odgovara da li želi transformaciju pomoću nje. U slučaju pozitivnog odgovora sistem ispituje da li su ispunjeni potrebni uslovi za zamenu, pa ukoliko jesu efektivno izvršava zamenu (na način kao što je opisano u odeljku 4.3) i generiše novi podcilj u stablu dokaza. Ukoliko sistem utvrdi da se transformacija podcilja sa izabranom valjanom formulom ne može izvršiti, prelazi na prikazivanje sledeće valjane formule.

Navedimo još i pomoćnu komandu koja predstavlja posebnu celinu u sistemu GRAPH, a može da se koristi kao pomoćna komanda u dokazivaču, s tim što se tekući podcilj prethodno iščupa iz drveta dokaza pod imenom rečenice.

10. Generisanje rečenica ekvivalentnih zadatoj.

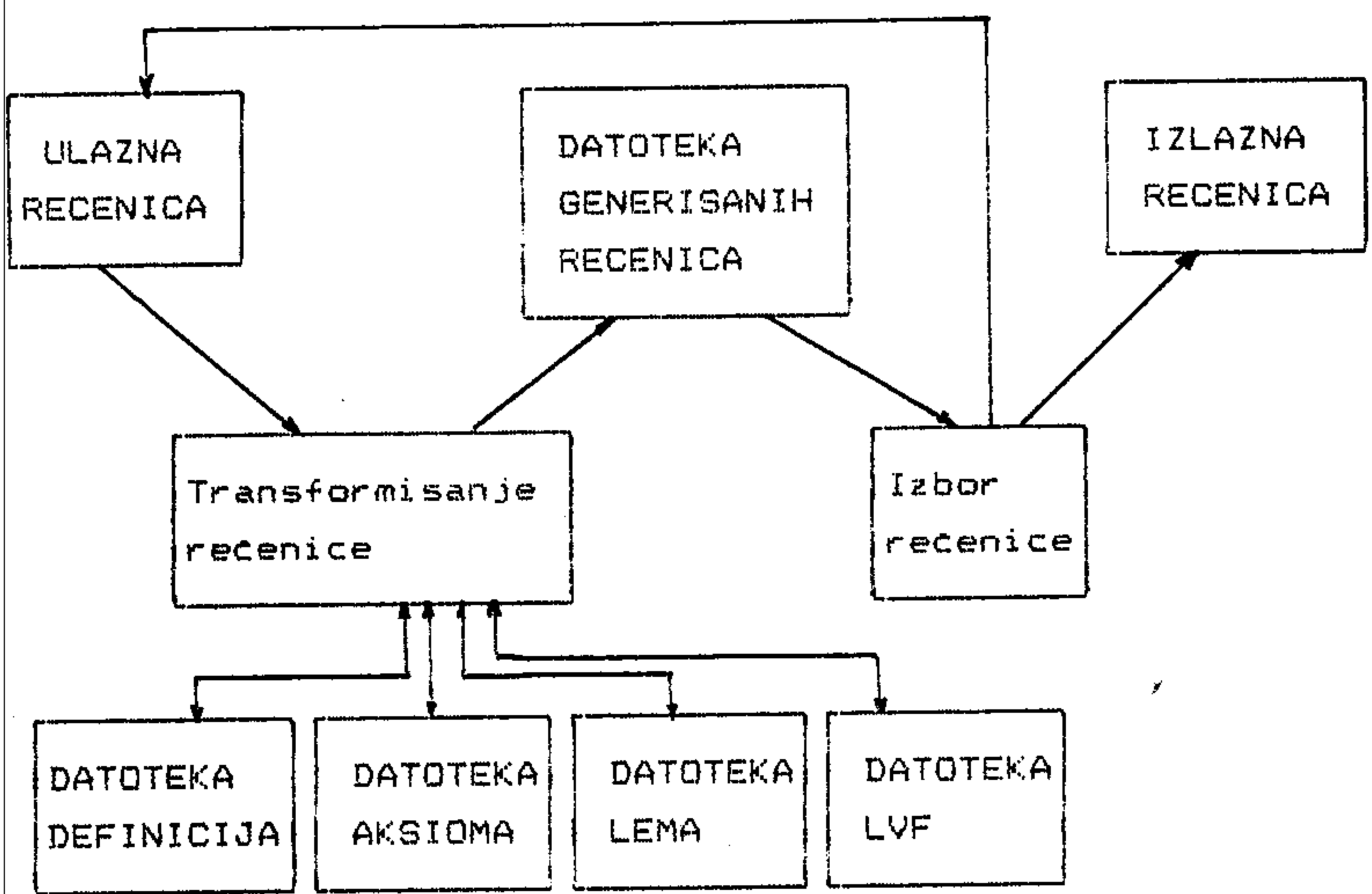
GENERATE SENTENCES EQUIVALENT [TO]<sentence name> .

Nakon što je sistem prihvatio ovu komandu on odgovara pitanjem korisniku kako da nastavi, a korisnik bira neku od sledećih 16 opcija.

1) automatski rad
2) interaktivni rad
3) automatski izbor rečenice koja će se dalje obradjivati
4) interaktivni izbor rečenice koja će se dalje obradjivati
5) automatska redukcija datoteke generisanih rečenica
6) interaktivna redukcija datoteke generisanih rečenica
7) zamena predikata po definiciji
8) zamena podformule korišćenjem definicije (u obrnutom smeru od 7) tj. zamena definiensa definiendumom)
9) zamena podformule korišćenjem aksioma
10) zamena podformule korišćenjem lema
11) zamena podformule korišćenjem LVF (logički valjanih formula)
12) dodavanje novih lema
13) zadavanje pojmova koje treba da sadrže generisane rečenice
14) automatski završetak rada
15) interaktivni završetak rada
16) informacija o postojećim opcijama

Na ulazu je rečenica koja se može transformisati na razne načine komandama 7) - 11), zatim komandama 3) ili 4) iz skupa generisanih biramo jednu rečenicu i onda sa njom nastavljamo rad.

Novogenerisane rečenice dodaju se datoteci dotle kreiranih rečenica, zatim se ponovo bira jedna rečenica koju ćemo obradjivati i postupak nastavljamo u dubinu, uz eventualno redukovanje datoteke izbacivanjem neinteresantnih rečenica sve do završetka rada, kada se bira jedna koja predstavlja konačni rezultat obrade. Shematski rad ovog modula se može prikazati slikom 5.



Slika 5.

Rečenica koja se obradjuje mora biti interno u sistemu prethodno prevedena sa jezika pravi GTCL na nivo kodirane kvantifikatorske formule. Transformacije se vrše na nivou kvantifikatorske formule, a zatim se ova rezultujuća rečenica inverznim prevodiocem prevodi na nivo engleskog jezika.

Interno u sistemu svaka rečenica je predstavljena vektorom kodiranih brojeva sa sledećom strukturom i značenjem:

LC	LF	kodirani engl.tekst	kod.kvant.formula	ND

referentni brojevi

gde su:

LC dužina vektora kodiranog engleskog teksta,

LF dužina vektora kodirane kvantifikatorske formule

Sledećih LC pozicija zauzima kodirani engleski tekst rečenice, a na sledećih LF pozicija smešten je vektor kodirane kvantifikatorske formule.

ND označava broj predikatskih slova koja učestvuju u kvantifikatorskoj formuli.

Sledećih ND pozicija zauzima vektor referentnih brojeva koji ukazuju na definicije pomenutih predikata.

Način kodiranja ovde nećemo iznositi, a opisan je u priručniku za rad sa sistemom <33>.

Obzirom na komunikaciju sa relativno obimnim datotekama aksioma, definicija i lema, nepodesno bi bilo primenjivati metod kompletne pretrage, pogotovo što su te datoteke smeštene na spoljnjoj memoriji čime se znatno produžava vreme pristupa.

U cilju smanjenja broja pristupa raznim definicijama, lemama i aksiomama rečenice su snabdevene referentnim brojevima koji ukazuju na definicije predikatskih slova u kvantifikatorskoj formuli. To omogućava da sistem znatno redukuje broj pristupa spoljnoj memoriji izdvajajući samo potencijalno relevantne definicije, aksiome i leme tj. samo one koje sadrže bar jedan od predikata u datoj podformuli rečenice. Ovime se odbacuju sve one definicije i leme za koje je unapred sigurno da se sa njima podcilj ne može transformisati.

Naravno, posle svake transformacije rečenice mora se revidirati i skup referentnih brojeva u skladu sa novim predikatima koji se pojavljuju u transformisanoj rečenici.

Sledeća naredba je slična prethodnoj, samo što rezultujuće rečenice sadrže bar jedan od unapred zadatih pojmova. Pojmovi se zadaju na nivou engleskog teksta.

GENERATE SENTENCES EQUIVALENT [TO]<sentence name> CONTAINING '<notions>' .

11. **Dokaz podcilja rezolucijom.** Komandom

PROVE <tree name> .

omogućava se prelaz na dokazivanje rezolucijom što ovde nećemo opisivati. Detaljan opis je dat u <45> i <33>.

Sem komandi navedenih od 1. - 11. kojima izvodimo dokaz, postoje i one koje u dokazivaču služe za pomeranje korena na odredjeni čvor ili gore, dole, levo ili desno od tekućeg čvora u drvetu dokaza, zatim postoje komande za prikazivanje dokazanih i nedokazanih podciljeva tj. statusa svih podciljeva na lišću drveta dokaza, za brisanje podrveta ispod odredjenog čvora u slučaju da nismo zadovoljni generisanim delom dokaza, zatim za prikazivanje drveta dokaza, prikaz formule tekućeg podcilja, i nekih podataka o samom drvetu dokaza itd.

Na primer komandom

TYPE <tree name> .

dobija se na ekranu drvo dokaza. Ostale komande se mogu naći u priručniku za rad sa sistemom GRAPH <33>.

6. NAJDUBLJE HEURISTIKE U AUTOMATSKOM I POLUAUTOMATSKOM REŽIMU DOKAZIVANJA TEOREMA

U narednom odeljku ćemo detaljno opisati heuristiku za odabiranje definicije ili leme kojom ćemo transformisati tekući podcilj. Zamena pomoću leme ili definicije prema tako uvedenoj heuristici definiše tzv. netrivijalne transformacije u izvodjenju dokaza. Heuristika bazira na pretpostavci da izbor ne zavisi samo od formule tekućeg podcilja već i od strukture cele relevantne formalne teorije.

U sistemu GRAPH se interno generiše multidigraf koji predstavlja grubu sliku strukture uvedene Aritmetičke teorije grafova odnosno postojećih definicija i lema.

Ideja te heuristike je da primenom multidigrafa vodimo akciju tako da dobijemo podciljeve na lišću drveta dokaza za koje je ispunjeno da je skup predikatskih slova u podformuli na desnoj strani od centralne implikacije podskup skupa predikatskih slova u podformuli leve strane implikacije podcilja. Ovo se postiže uvodjenjem trasa napada koje ćemo opisati u odeljku 6.1.

Idejna razrada i implementacija je radjena pod rukovodstvom i u saradnji sa prof. D. Cvetkovićem.

U odeljku 6.2 biće opisana heuristika za redukovanje broja kvantifikatora kojom se dalje povećava moć postojećeg dokazivača. Ovde su izložene modifikacije podcilja zasnovane na ekvivalentnim logičkim transformacijama, kao i prelazak na jače tvrdjenje uz korišćenje odredjenih valjanih formula koje sistem interno generiše.

Odeljak 6.3 opisuje jedan pristup u tretiranju n-arnih konjunkcija i disjunkcija.

Obe heuristike su uvedene u cilju poboljšanja nivoa unifikacije, kao i poboljšanja mogućnosti potvrdjivanja tačnosti podcilja.

Idejna razrada ovih heuristika je radjena samostalno.

6.1 HEURISTIKA ZA IZBOR RELEVANTNE DEFINICIJE I LEME

Osnovna ideja vodilja ove heuristike je u činjenici da izbor relevantne definicije odnosno leme pomoću koje ćemo transformisati dati podcilj kako bismo pomogli u kreiranju dokaza, ne treba i ne može da zavisi samo od formule tekućeg podcilja, već pritom treba voditi računa o strukturi cele formalne teorije u kojoj izvodimo dokaz.

U heuristici je implicitno korištena činjenica da u većini slučajeva u dokazu teoreme koristimo izvesnu abstrakciju tj. selekciju pretpostavki datih u hipotezi tvrdjenja, a potrebnih za izvodjenje zaključka. To znači da je čest slučaj da u teoremi oblika $H_1 \wedge H_2 \wedge \ldots \wedge H_n \Rightarrow C$ nećemo koristiti sve H_i-ove za izvodjenje C-a.

Na pojam apstrakcije u dokazivanju ukazano je u <61>. Dat je sledeći zgodan primer iz geometrije u ravni. Ako se u dokazu neke teoreme o svim kvadratima koristi samo činjenica da je kvadrat paralelogram kome su sve četiri stranice jednake, a ne i činjenica da su mu uglovi pravi, tada u suštini teorema nije o kvadratima već o rombovima. To je opšta osobina matematičkih dokaza. Naime, oni ne koriste sve osobine objekta o kojem se teorema dokazuje, već uvek samo neki izbor iz njih. Kako u opštem slučaju matematički objekti imaju beskonačan broj osobina, jasno je da u konačnom dokazu možemo koristiti samo konačan broj tih osobina. To nazivamo procesom apstrakcije, odnosno selekcijom osobina objekta o kojem dokazujemo teoremu.

Ilustrujmo to primerom iz aritmetike. Pretpostavimo da hoćemo da dokažemo nešto o broju 2, na primer, da mu je kvadratni koren iracionalan. Kako 2 ima beskonačan broj osobina u dokazu izdvajamo samo konačan skup od njih. Dakle, teorema nije o 2, već o nekom opštijem objektu 2^* za koji nemamo definisano ime kao u slučaju romba u prethodnom primeru. U gornjem dokazu koristićemo samo činjenicu da je 2 prirodan broj i da je prost. Znači korektnije bi bilo reći da dokaz nije o broju 2, vec za proizvoljan prost broj. No čak i to nije sasvim tačno jer su samo neke osobine prostih brojeva koriscene, kao što su u prethodnom primeru korišcene samo neke osobine rombova.

Dakle, dokaz implicitno uključuje ne samo dedukciju, vec apstrakciju i generalizaciju. Dokaz svake teoreme implicitno kreira novi matematički objekat i to upravo onaj koji je definisan premisama koje aktualno koristimo u dedukciji.

Napomenimo još da iskustvo drugih istraživača na ovom području govori da je malo uradjeno na odabiranju relevantne definicije. Na primer, Bledsoe ukazuje u <7> da primenu definicija treba pažljivo kontrolisati i primenjivati je kad druge strategije ne daju rezultat, ili kad se ustanovi da će primena definicije učiniti neko dobro. Preporučuje se primena definicije neobičnih predikata gde se taj pojam samo intuitivno oseća, a ne precizira kako izdvojiti takve predikate automatski. Preporučuje se, takodjer, zamena definicijom glavnog predikata u zaključku, a zamena definicijom u hipotezi treba da se vrši samo ako u nekom konjunktu iz hipoteze nadjemo moguće združenje za zaključak.

Heuristika koju uvodimo u ovom poglavlju predstavlja korak napred na području odabiranja relevantne definicije i leme. Medjutim, treba istaci da je njena moć prvenstveno u teorijama sa bogatom strukturom definicija, kakva je formalna teorija u kojoj je primenjena. Naravno, moguća su i razna njena poboljšanja kako bi se još više približila ljudskim kognitivnim sposobnostima.

Predjimo sada na sam opis i definisanje pojmova potrebnih da bi se izložila heuristika.

Formalnu teoriju koja je predstavljena odredjenim skupom aksioma, definicija i lema interno ćemo prikazati multidigrafom Γ koji sistemu daje globalno znanje te strukture. Multidigrafom zadajemo računaru informaciju o tome koje se predikatsko slovo pomocu kojeg definiše i delimično prenosimo strukturu logičkih operacija kroz znake + i - koji su obeležja grana što će biti detaljno objašnjeno u narednom izlaganju.

Obeležimo sa F formulu definiensa u definiciji nebazičnog predikatskog slova D. Primenimo na F, zatim, operator ogoljavanja L(F) kojim odbacujemo argumente unarnih, binarnih i ternarnih predikata, logičke operacije i kvantifikatore, zagrade i zareze, promenljive i ime grafa u sufiksu predikatskog naziva. Time dobijamo niz $D_1, D_2, \ldots, D_n$ redom predikatskih slova koja učestvuju u formuli F. Zatim u multidigrafu Γ koji predstavlja strukturu definicija, uvodimo redom grane od svakog čvora $D_1, D_2, \ldots, D_n$ do čvora D.

Nadalje uvedimo sledeću definiciju znaka za svaku podformulu date formule F:

Definicija 11. **Znak** podformule date formule F definiše se rekurzivno.

Cela formula F ima pozitivan znak.
Ako $A \wedge B$ ili $A \vee B$ imaju pozitivan (negativan) znak tada isti znak nasledjuju i A i B.
Ako je $\neg A$ pozitivnog (negativnog) znaka, tada A ima negativan (pozitivan) tj. suprotan znak.
Ako je $A \Rightarrow B$ pozitivnog (negativnog) znaka, tada je A negativnog (pozitivnog) tj suprotnog znaka, a B nasledjuje znak polazne podformule tj. ostaje pozitivna (negativna).
Ako je $(\forall x)A$ ili $(\exists x)A$ pozitivnog (negativnog) znaka tada je istog znaka i podformula A.

Uvodjenjem znaka, svakom predikatskom slovu iz L(F) možemo pridružiti znak odgovarajuće atomarne formule u formuli F u kojoj se to predikatsko slovo pojavljuje.

Tako uvedeni operator **pridelјivanja znaka predikatskom slovu** D koje učestvuje u formuli F, označimo sa sgnD.

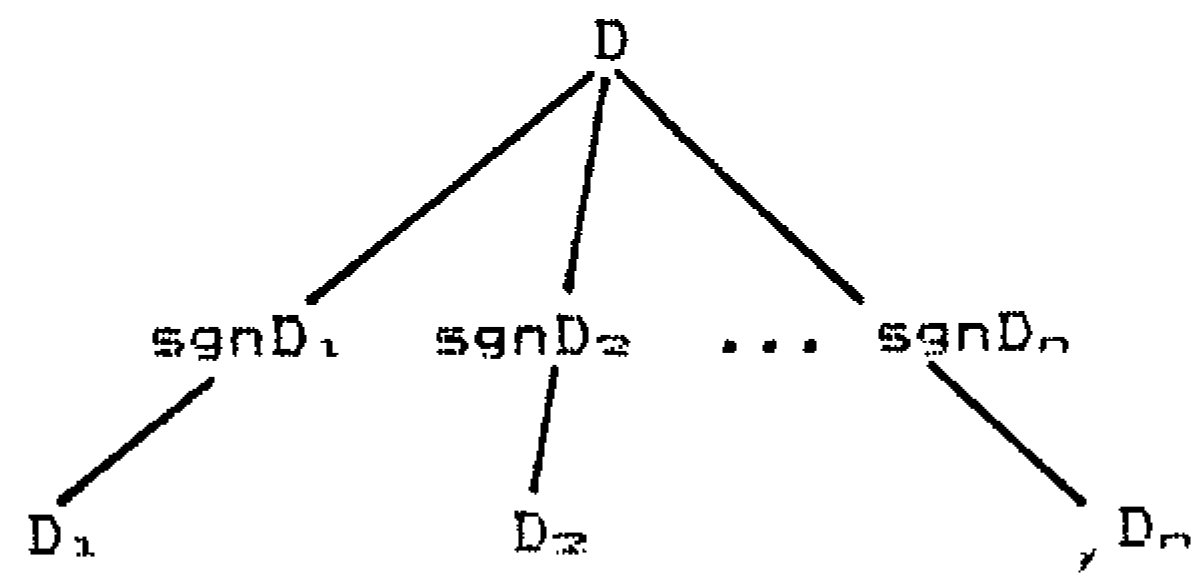
Definiciju kojom se definiše predikatsko slovo D možemo predstaviti ogoljenom shemom

$$(D,+) \longleftrightarrow (D_1, sgnD_1), (D_2, sgnD_2), \ldots (D_n, sgnD_n) \qquad (1)$$

koja se dobija primenom operatora ogoljavanja na formule definienduma i definiensa i operatora prideljivanja znaka za predikate definiensa.

Za dato predikatsko slovo D, predikatska slova $D_1, D_2,$ zvati **direktnim predhodnicima** od D, a obeležavat ćemo ih sa p(D).

Grane u multidigrafu Γ se uvode tako da od svakog prethodnika iz p(D) vodi usmerena grana ka čvoru sa obeležjem D. Grane su takodjer označene i nose kao obeležje znak sgn D_i. Za ogoljenu definiciju predstavljenu shemom (1), odgovarajući deo multidigrafa će izgledati kao što je prikazano na slici 6.



Slika 6

Na analogan način ćemo predstaviti lemu oblika $F_1 \Rightarrow F_2$ kojoj je implikacija vrhovna logička operacija. Neka je niz pojavljivanja predikatskih slova u F_1 i F_2 dobijen primenom operatora L redom na formule F_1 i F_2 respektivno dat sa

$$L(F_1) = M_1, M_2, \ldots, M_k \quad \text{i}$$

$$L(F_2) = N_1, N_2, \ldots, N_r.$$

Ponovo korišćenjem definicije znaka odredjujemo znakove elementarnih podformula u formulama F_1 i F_2 i redom ih prideljujemo odgovarajućim predikatskim slovima $M_1, M_2, \ldots M_k$

i $N_1, N_2, \ldots, N_r$, kao $\mathrm{sgn} M_i$ $i=1,k$ i $\mathrm{sgn} N_j$ $j=1,r$.

Najzad, možemo predstaviti lemu u obliku ogoljene sheme kao

$$(M_1, \mathrm{sgn} M_1), (M_2, \mathrm{sgn} M_2), \ldots, (M_k, \mathrm{sgn} M_k) \longrightarrow (N_1, \mathrm{sgn} N_1), \ldots \\ \ldots, (N_r, \mathrm{sgn} N_r) \qquad (2)$$

Uvedimo nadalje definicije pojmova put, antiput i znak puta u multidigrafu Γ.

Definicija 12. **Putem** nazivamo niz čvorova $B_1, B_2, \ldots, B_n$ koji su povezani granama koje vode od B_i ka B_{i+1} za $i=1,2,\ldots,n-1$ i pritom je B_i direktan prethodnik od B_{i+1}.

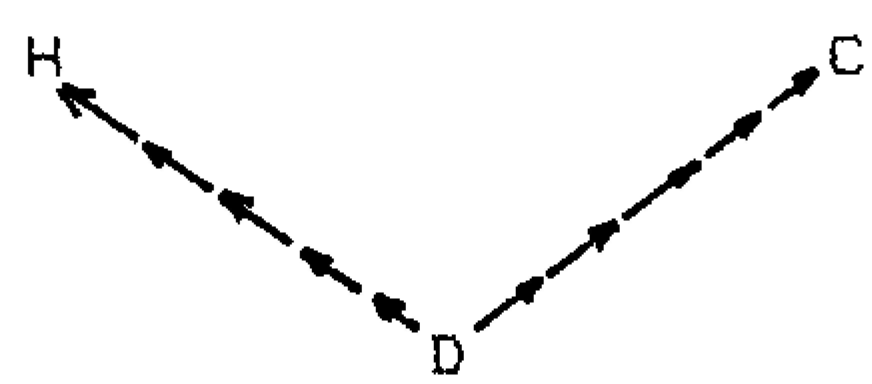
Definicija 13. Ako niz čvorova $B_1, B_2, \ldots, B_n$ predstavlja put tada niz $B_n, \ldots, B_1$ predstavlja **antiput**.

Dužina puta, odnosno antiputa je definisana standardno kao broj grana u putu, odnosno antiputu.

Definicija 14. **Znak puta** je pozitivan (negativan) ako se u njemu negativno označene grane javljaju paran (neparan) broj puta.

Definicija 15. U multidigrafu uvodimo **trasu napada** od predikatskog slova C ka slovu H ako postoji čvor D takav da od C do D postoji antiput, a od D do H put.

Trasu napada kao kompoziciju antiputa i puta ilustruje sledeća slika:



Slika 7

U specijalnom slučaju kad se D poklapa sa C (odnosno sa H) trasa napada se svodi na put (odnosno antiput).

Čvor D ćemo zvati prekidačem trase napada.

Definicija 16. **Dužina trase napada** od C ka H u oznaci d(C,H) je jednaka zbiru dužine antiputa od C do D i dužine puta od D do H. Prekidač trase napada biramo tako da dužina trase napada bude minimalna. U slučaju da trasa napada od C ka H ne postoji, dužina se definiše kao beskonačno veliki broj, dakle $d(C,H)=\infty$.

U dosadašnjem izlaganju smo grane multidigrafa, pa time i trasu napada, generisali samo korišćenjem grana u multidigrafu koje potiču od definicija. Medjutim, i leme se takodjer mogu koristiti u generisanju trase napada ali pritom moramo uvesti u igru označenost grana, a takodjer i znak predikatskih slova za koje definišemo trasu napada.

Pokazaćemo sada kako u igru uvodimo leme.

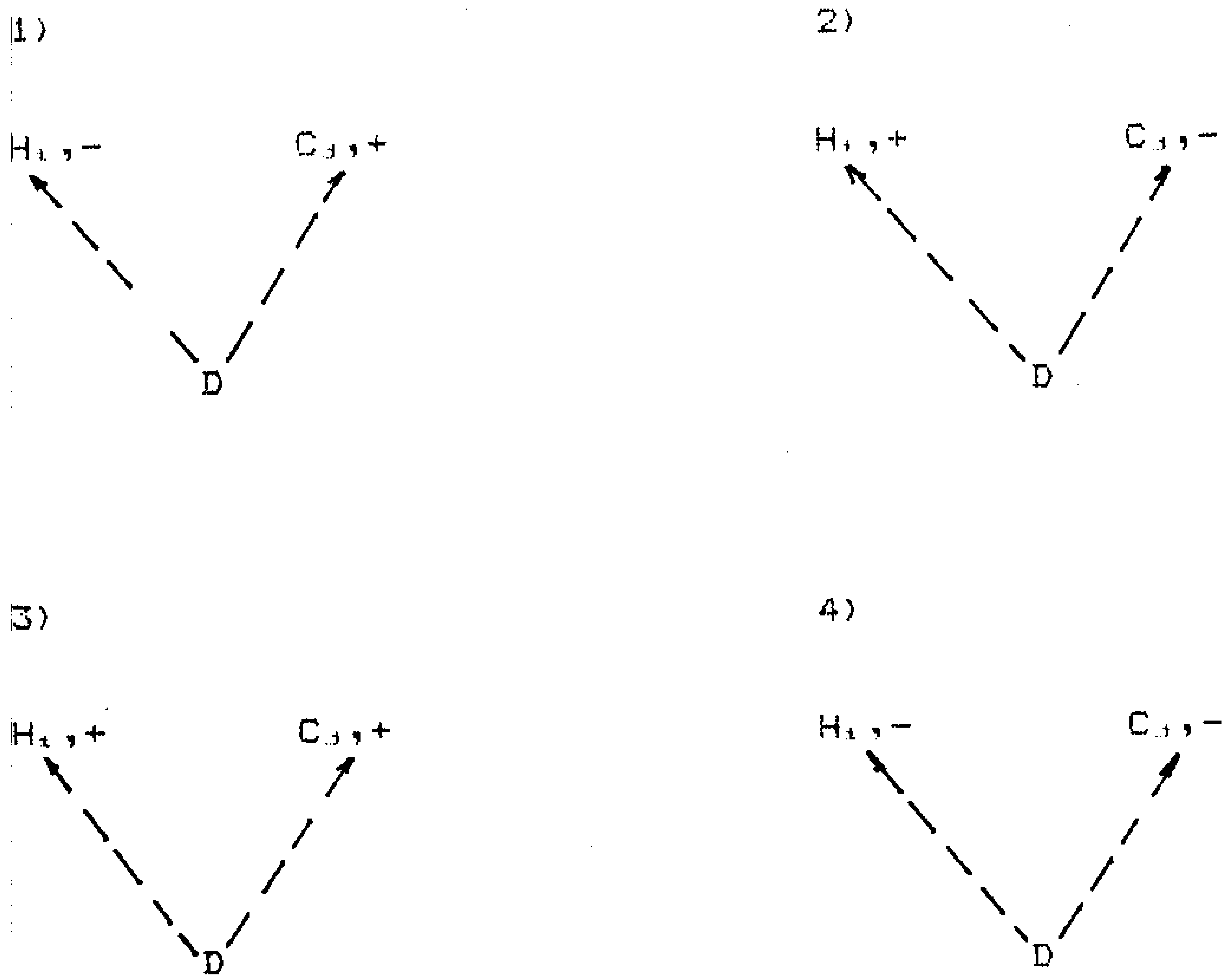
Neka je data formula F oblika $F_1 => F_2$ i neka su nizovi $L(F_1)$ i $L(F_2)$ respektivno dati sa $H_1,\ldots,H_m$ i $C_1,\ldots,C_n$.

Analogno kao i u slučaju lema, ogoljeni oblik formule F predstavljamo shemom (3):

$$(H_1,\mathrm{sgn}H_1),(H_2,\mathrm{sgn}H_2),\ldots,(H_m,\mathrm{sgn}H_m) \longrightarrow (C_1,\mathrm{sgn}C_1),\ldots,(C_n,\mathrm{sgn}C_n) \qquad (3)$$

Pri utvrdjivanju trase napada od C_j ka H_i prema dosadašnjem izlaganju potrebno je generisati redom predhodnike prvog, drugog itd. nivoa za H_i i C_j koristeći ogoljene definicije dok ne nadjemo prekidač tj. presečni predikat sa minimalnim zbirom nivoa.

Pritom se prenose i odgovarajući znaci koji su obeležja grana u trasama napada, kako su uvedeni u multidigrafu Γ. Oni su potrebni zbog utvrdjivanja mogućnosti upotrebe lema u generisanju trase napada. Prema znacima koji su prideljeni predikatskim slovima H_i i C_j u shemi formule F date sa (3), pri uključivanju leme u grane trasa napada razlikovaćemo sledeća četiri slučaja prikazana na slici 8:



Slika 8

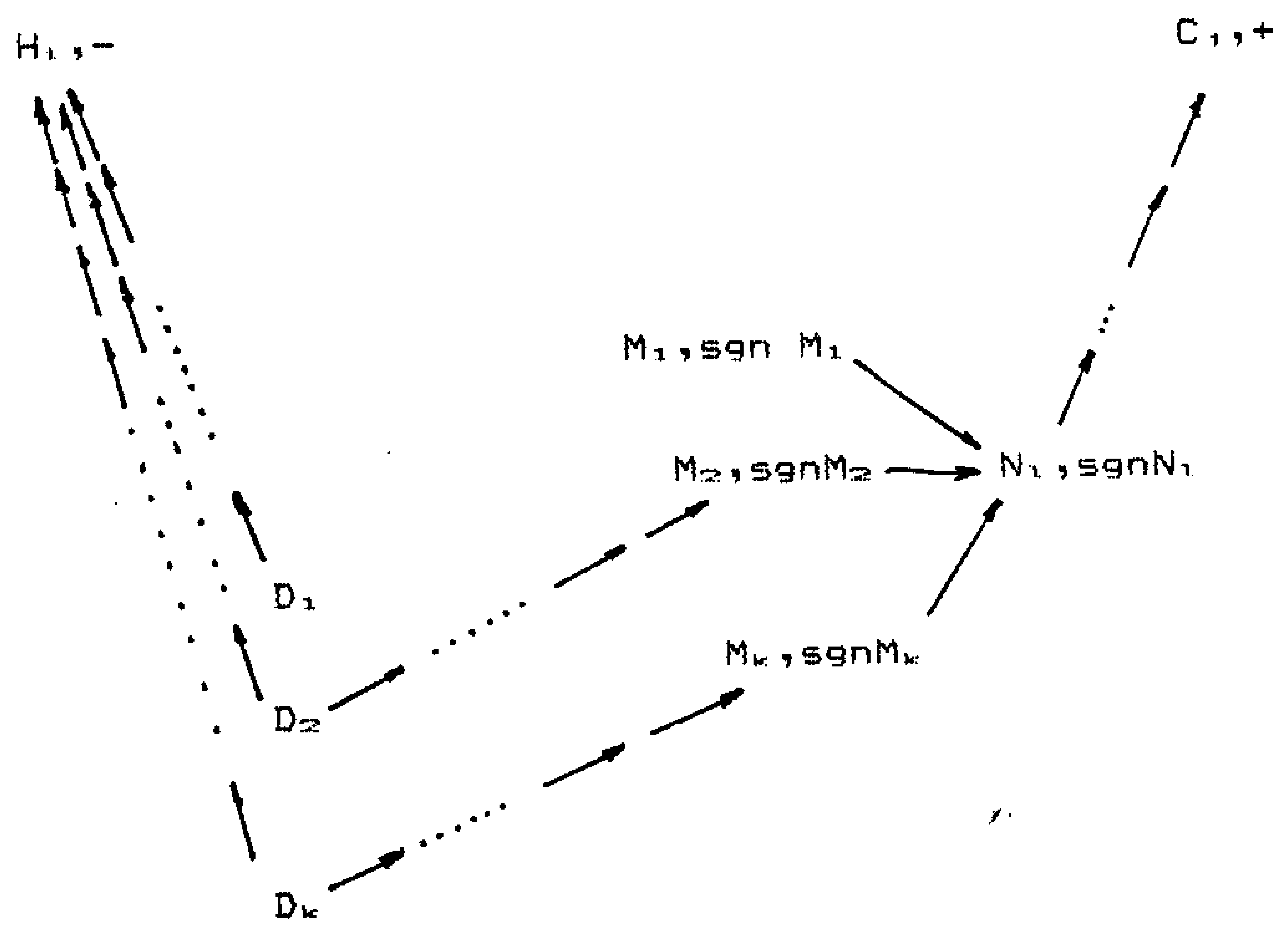
Predjimo sada na opis slučaja 1).

U oba dela trase napada, dakle u antiputu od C_j ka D i putu od D ka H_i možemo koristiti lemu čiji je ogoljeni oblik dat shemom (2) ako je ispunjeno sledeće:

-dotle kreirani antiput završava čvorom sa obeležjem N_l gde je $(1<l<r)$

-znak dotle kreiranog antiputa jednak je znaku $sgnN_l$ koji je pridružen tom predikatskom slovu u lemi sa shemom (2).

Kako uključivanje leme podrazumeva njeno eventualno koriščenje (ukoliko se ispune svi potrebni uslovi za njenu primenu) u igru ulaze sva predikatska slova $M_1, M_2, ..., M_k$. Dužinu dotle kreiranog antiputa uvećamo za 1 i dalje tražimo prosečnu trasu napada čija je dužina aritmetička sredina pojedinačnih trasa napada od odovarajućih M-ova sa pridruženim znakom ka H_i,- kao što ilustruje slika 9.

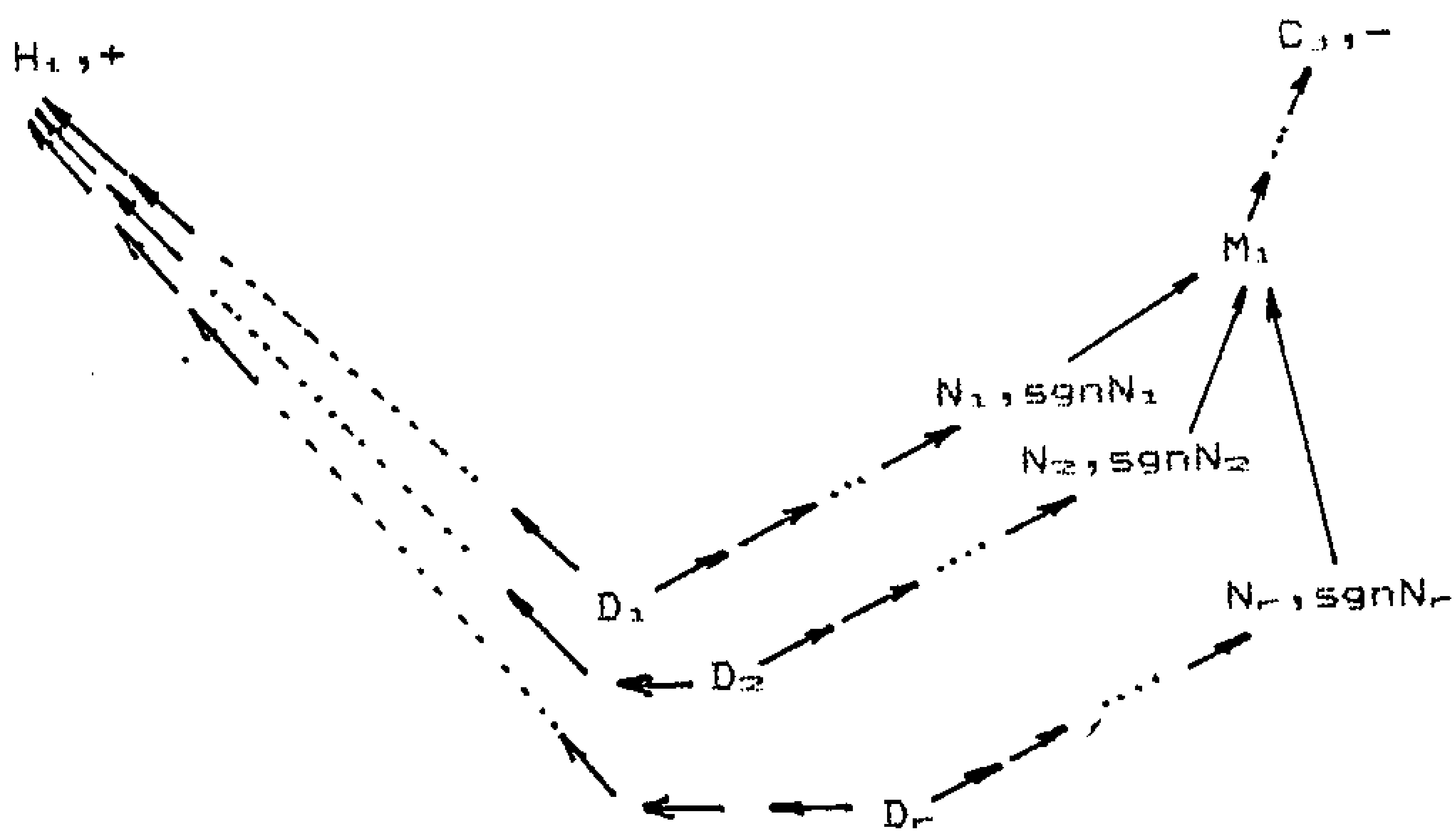


Slika 9

Ako dužinu antiputa od C_j do N_1 obeležimo sa l, a dužine trasa napada od M_n,sgnM_n do H_i,- obeležimo sa l_n gde je $n=1,2,...,k$, tada je dužina trase napada od C_j,+ ka H_i,- jednaka

$$d(C_j,H_i) = l + 1 + 1/k \sum_{n=1}^{k} l_n$$

Slično kao pod 1) u slučaju 2) u oba dela trase napada možemo koristiti lemu čiji je ogoljeni oblik dat sa (2). Na primer, u generisanju predhodnika od C_j,- (analogno važi za predhodnike od H_i,+) ukoliko se medju predhodnicima na antiputu pojavi čvor sa obeležjem M_1 i pritom je znak puta od C_j do M_1 jednak znaku pridruženom tom predikatu u lemi tj sgn M_1, možemo koristiti lemu sa ogoljenom shemom (2) konstruisanjem trasa napada od svih elemenata $N_1, N_2, \ldots, N_r$ ka H_i kao što ilustruje sledeća slika:



Slika 10

Dužina trase napada od C_j,- ka H_i,+ se sastoji od dužine antiputa od C_j do M_1 uvećane za jedan i prosečne dužine trasa napada od N_n,sgnN_n ka H_i,+ (za $n=1,2,\ldots,3$).

Ako sa l obeležimo dužinu puta M_1C_j, a dužine trasa napada od N_n,sgnN_n do H_i,+ redom sa l_n (gde je $n=1,2,\ldots,r$) tada je dužina trase napada od C_j,- ka H_i,+ jednaka

r

$$d(C_j,H_i) = 1 + 1 + 1/r \ (\sum_{n=1} l_n)$$

Preostaju još slučajevi navedeni kao 3) i 4) na slici 8. Kod njih se leme uključuju analogno kao u gornjem izlaganju. U trasi napada u traženju predhodnika od H_i,+ u slučaju 3) ili traženju predhodnika od C_i,− u slučaju 4) leme se uključuju u trase napada kao pod 1). U preostalim slučajevima u generisanju trase napada, leme koristimo kao pod 2).

Pošto smo izložili detalje o trasama napada i njihovim dužinama predjimo na definisanje kompleksnosti predikata, formule, i drveta dokaza.

Za predikat sa slovom $X \in \{ C_1, C_2, \ldots, C_n \}$ koji se pojavljuje u zaključku formule F date u ogoljenom obliku sa (3), dužinu najkraće trase napada od X do nekog predikatskog slova $H \in \{ H_1, H_2, \ldots, H_m \}$ obeležimo sa $d_M(X)$.

$$d_M(X)=\min \{ d(X,H_1), d(X,H_2), \ldots, d(X,H_m) \}$$

Uvedimo sada **kompleksnost predikatskog slova** X gde je $X \in \{ C_1, C_2, \ldots, C_n \}$ u formuli F datoj sa shemom (3)

$$c(X) = \begin{cases} 0 & \text{ako je } X \in \{ H_1, H_2, \ldots, H_n \} \\ d_M(X) & \text{ako se minimalna trasa napada svodi na put} \\ 1 + 1/s \sum_{Y \in P(X)} d_M(Y) & \text{inače} \end{cases}$$

gde je sa P(X) obeležen skup direktnih predhodnika od X, a s je njihov broj.

Kompleksnost formule F obeležavamo sa K i definišemo kao prosečnu kompleksnost predikatskih slova koja se pojavljuju u desnoj strani centralne implikacije formule F.

$$K(F)=1/n \sum_{i=1}^{n} c(C_i)$$

Pretpostavljeno je da je formula F data u ogoljenom obliku sa (3).

U slučaju da je formula F dokazana njena kompleksnost je jednaka 0.

Kompleksnost drveta T obeležavamo sa C, a definiše se kao prosečna kompleksnost formula na lišcu drveta T. Za dato drvo T obeležimo sa F_1, F_2,..., F_k formule na lišcu tog drveta. Tada je kompleksnost drveta data sa:

$$C(T)= 1/k \sum_{i=1}^{k} K(F_i)$$

Pretpostavimo sada da sistem izvršava netrivijalnu transformaciju podcilja F. (Netrivijalnom transformacijom smatramo primenu definicije ili leme). Rezultujući podcilj obeležimo sa F_{tr}. Nakon primene bloka trivijalnih transformacija na podcilj F_{tr} sistem će eventualno generisati drvo T novih podciljeva u odnosu na F_{tr}.

Odaberimo tri transformacije sa maksimalnom razlikom u kompleksnosti podcilja pre i posle primene netrivijalne transformacije $K(F)-K(F_{tr})$, a zatim biramo onu transformaciju za koju je C(T) minimalno.

6.2 HEURISTIKA ZA REDUKOVANJE KVANTIFIKATORA

Vecina autora koristi skolemizirane formule u interaktivnom dokazivanju teorema. Misljenja sam da je u tom obliku teze osetiti probleme na nivou unifikacije, te prema tome covek teze moze da ucestvuje i usmerava dokaz na adekvatan nacin. U ovom odeljku cemo izloziti metod kojim se dati podcilj transformise bilo u ekvivalentan ili u podcilj iz kojeg dati sledi, ali tako da dobijena formula sadrzi eventualno manje kvantifikatora i prostija je za dokaz. Formula sve vreme zadrzava prirodan oblik povoljan za korisnika, (sto znaci da ne izbacujemo implikaciju i kvantifikatore kao kod skolemizacije), a kako se transformacije odvijaju na slican nacin kako to covek radi u toku dokaza, lako mu je da intervenise i eventualno preusmeri dokaz u slucaju potrebe.

Pre opisa heuristike uvescemo sledece definicije:

Definicija 17. Formulu F_1 cemo zvati **jacom formulom** od formule F_2 (i dualno F_2 cemo zvati **slabijom formulom** od formule F_1) ukoliko vazi $F_1 \Rightarrow F_2$.

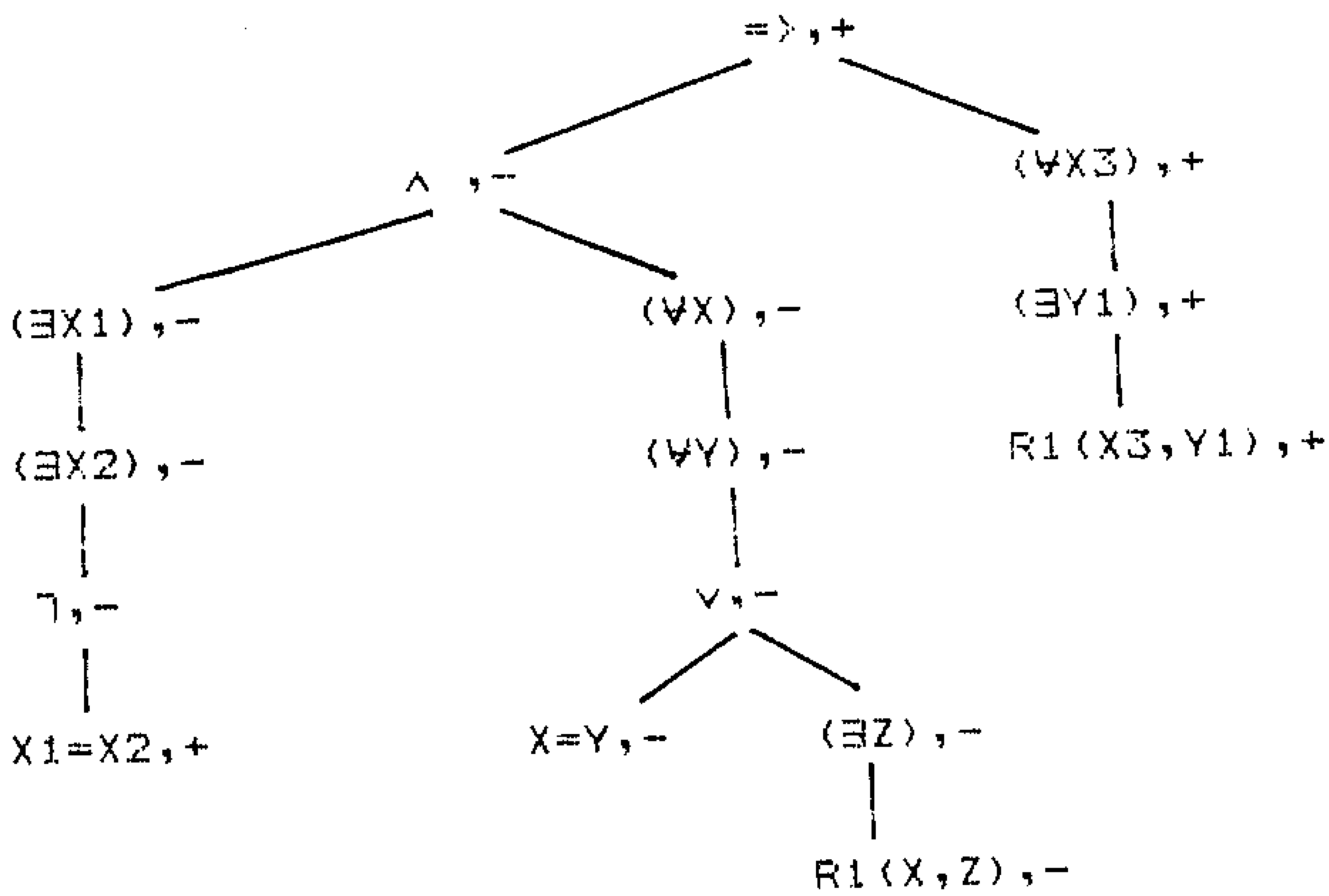
Uvedimo takodjer znak + ili − za svaku podformulu date formule kao sto je navedeno u definiciji 11. u odeljku 6.1.

U standardnoj reprezentaciji formule pomocu drveta, gde unutrasnji cvorovi drveta imaju obelezje odgovarajuce logicke operacije (ukljucujuci kvantifikatore), a cvorovi na liscu predikate, svakom cvoru prideljujemo jos i znak podformule za koju je taj cvor vrh, prema definiciji znaka.

Primer 3. Formula

$$(\exists X1)(\exists X2)\neg X1=X2 \wedge (\forall X)(\exists Y)(X=Y \vee (\exists Z)R1(X,Z)) \Rightarrow$$
$$\Rightarrow (\forall X3)(\exists Y1)R1(X3,Y1)$$

se predstavlja drvetom prikazanim na slici 11.



Slika 11

U izlaganju heuristike ćemo koristiti takodjer i sledeće tvrdjenje

Lema 1. Ako zamenimo jedno pojavljivanje pozitivne (negativne) podformule neke formule G sa jačom (slabijom) formulom rezultujuća formula je jača od polazne formule G.

Dokaz se lako izvodi indukcijom po broju logičkih operacija u formuli uz formulisanje analognog tvrdjenja za generisanje slabije formule.

Redukovanje kvantifikatora u formuli počinjemo slično kao u algoritmu za skolemizaciju. Naime, u prvom koraku izostavljamo univerzalne kvantifikatore koji predstavljaju vrhovne operacije u formuli. Zatim izostavljamo kvantifikatore koji mogu da dodju na vrh kao univerzalni. Ovo se izvodi bez transformisanja podcilja tj. ne pomerajući kvantifikatore ka vrhu. Nakon toga, ostatak kvantifikatora se pomera što bliže ka predikatima. U drugom koraku generišemo umesto date formule jaču formulu koristeći gore navedenu lemu i sledeće valjane formule:

$$(\forall x)F(x) \Rightarrow F(t_1) \wedge F(t_2) \wedge \ldots \wedge F(t_n)$$

$$F(t_1) \vee F(t_2) \vee \ldots \vee F(t_m) \Rightarrow (\exists x)F(x)$$

U režimu rada bez učešća čoveka sistem treba da odredi za koju podformulu sa vrhovnim univerzalnim kvantifikatorom i usto negativno označenu (ili dualno, podformulu sa vrhovnim egzistencijalnim kvantifikatorom i usto pozitivno označenu) da se opredeli, a takodjer treba da izabere broj n i izvrši izbor terma $t_1, t_2, \ldots t_n$, (u dualnom slučaju broj m i izbor terma $t_1, \ldots, t_m$).

Ovaj deo je sličan upotrebi pravila višestrukih kopija u <7>, naime, koristi se na analognim mestima u formuli, medjutim u <7> se ne daje mogućnost automatskog odlučivanja unapred o broju kopija. Takodjer za razliku od pravila višestrukih kopija ovaj metod rešava istovremeno i nivo unifikacije.

U interaktivnom radu čovek odabira podformulu tako što daje ime vezane promenljive uz odgovarajući kvantifikator. Ovime se na jedinstven način izdvaja jedna podformula zbog uvedene konvencije u sistemu GRAPH da se promenljive vezane različitim kvantifikatorima različito označavaju, a takodjer i različito od svih slobodnih. Sistem zatim treba da proveri da li je pozicija kvantifikatora u formuli podcilja takva da je transformacija dozvoljena. Ukoliko sistem nakon provere potvrdi da se radi o kvantifikatoru čiji znak podformule dopušta transformaciju, sistem ponudi korisniku izbor terma. Ako se korisnik eventualno ne složi sa izborom sistema biće zamoljen da predloži svoj izbor terma. Na ovaj način sretno se spaja nepogrešivost mašine i čovekova intuicija u globalnoj strategiji vodjenja dokaza.

Predjimo sada na opis pomenutih koraka u heuristici.

6.2.1 Eliminacija kvantifikatora zasnovana na ekvivalentnim logičkim transformacijama

Koristićemo sledeće logički valjane formule, gde x nema slobodnih pojavljivanja u A:

1) $A \vee (\forall x)B(x) \Leftrightarrow (\forall x)(A \vee B(x))$,
2) $A \vee (\exists x)B(x) \Leftrightarrow (\exists x)(A \vee B(x))$,
3) $A \wedge (\forall x)B(x) \Leftrightarrow (\forall x)(A \wedge B(x))$,
4) $A \wedge (\exists x)B(x) \Leftrightarrow (\exists x)(A \wedge B(x))$,
5) $A \Rightarrow (\forall x)B(x) \Leftrightarrow (\forall x)(A \Rightarrow B(x))$,
6) $A \Rightarrow (\exists x)B(x) \Leftrightarrow (\exists x)(A \Rightarrow B(x))$,
7) $(\forall x)B(x) \Rightarrow A \Leftrightarrow (\exists x)(B(x) \Rightarrow A)$,
8) $(\exists x)B(x) \Rightarrow A \Leftrightarrow (\forall x)(B(x) \Rightarrow A)$,
9) $\neg(\forall x)B(x) \Leftrightarrow (\exists x)\neg B(x)$,
10) $\neg(\exists x)B(x) \Leftrightarrow (\forall x)\neg B(x)$,
11) $(\forall x)(\forall y)F(x,y) \Leftrightarrow (\forall y)(\forall x)F(x,y)$,
12) $(\exists x)(\exists y)F(x,y) \Leftrightarrow (\exists y)(\exists x)F(x,y)$.

Ako primenimo redom te formule s leva u desno za transformisanje date formule možemo videti da se kvantifikatori pomeraju nagore u drvetu logičkih operacija. Kada koristimo logički valjane formule obeležene sa 7) – 10) lako se vidi da pomeranje kvantifikatora kroz negaciju ili implikaciju sleva uslovljava istovremeno i promenu znaka te podformule. Budući da možemo izostaviti univerzalni kvantifikator koji je vrhovni u drvetu logičkih operacija, interesuje nas da li za neki kvantifikator možemo ustanoviti bez sprovodjenja transformacija da li on može doći na vrh.

Kako vrhovna operacija ima pozitivan znak po definiciji, vidi se lako da je potreban uslov za to da kvantifikator bude univerzalan i označen sa + ili egzaistencijalni i označen sa −.

Najzad kako ne smemo menjati redosled univerzalnih i egzistencijalnih kvantifikatora sledi algoritam za izostavljanje kvantifikatora koji mogu ekvivalentnim transformacijama podcilja doći na vrh kao univerzalni, bez izvodjenja tih transformacija na podcilju.

ALGORITAM

Za svaki pozitivni univerzalni ili negativni egzistencijalni kvantifikator ispitaćemo put od tog čvora do vrha u drvetu logičkih operacija sa prideljenim znacima. Ako na tom putu nema čvorova sa pozitivnim egzistencijalnim ili negativnim univerzalnim kvantifikatorom polazni kvantifikator se može dovesti na vrh kao univerzalni.

Uklonićemo sve kvantifikatore zajedno sa odgovarajućim vezanim promenljivim koji zadovoljavaju gore navedeni uslov.

Posle toga ostatak kvantifikatora guramo ka predikatima tj. što je moguće više udesno. To se postiže primenom LVF 1) − 12) u smeru zdesna ulevo sve dok je moguće primeniti neku od navedenih transformacija.

Dati algoritam ćemo ilustrovati sledećim primerom:

Primer 4.

U formuli datoj u primeru 3. sledeći čvorovi su potencijalni kandidati za eliminaciju:

$\exists X1;-$, $\exists X2;-$, $\exists Z;-$ i $\forall X3;+$.

Prvi, drugi i četvrti zadovoljavaju uslove prezentiranog algoritma te se mogu ukloniti kao što je gore navedeno.

Rezultujuća formula je

$\neg X1=X2 \wedge (\forall X)(\forall Y)(X=Y \vee (\exists Z)R1(X,Z)) \Rightarrow (\exists Y1)R1(X3,Y1)$.

Nakon pomeranja preostalih kvantifikatora udesno tj. ka predikatima, formula dobija sledeci oblik po izvršenim transformacijama:

$\neg X1=X2 \wedge (\forall X)\ ((\forall Y)X=Y \vee (\exists Z)R1(X,Z)) \Rightarrow (\exists Y1)R1(X3,Y1)$.

6.2.2.Eliminacija kvantifikatora uvodjenjem jačeg podcilja

Koristeci rezultat dat u lemi 1. pokušavamo da zamenimo pozitivnu podformulu tipa $(\exists x)F(x)$ sa disjunkcijom $F(t_1) \vee F(t_2) \vee ... \vee F(t_n)$. Na analogan način negativnu podformulu tipa $(\forall x)F(x)$ zamenjujemo sa dualnom konjunkcijom. Da bismo primenili takvu transformaciju treba prethodno odrediti koliko disjunkta (ili analogno konjunkta) želimo, a zatim izvršiti izbor terma t_i za $i=1,2,..,n$. Da bismo to realizovali uvodimo sledecu heuristiku:

Obeležimo sa G formulu tekuceg podcilja, a sa $(A_1, sgnA_1)$, $(A_2, sgnA_2)$,....,$(A_r, sgnA_r)$ redom sve podformule u G koje su tipa $A_i \equiv (Q_i\ y_i)F_i(y_i)$ za $i=1,..,r$ gde Q_i označava univerzalni (egzistencijalni) kvantifikator ako je znak sgn A_i respektivno negativan (pozitivan).

Ako u formuli G postoji više od jedne takve podformule, izabracemo onu čiji kvantifikator napada manje predikata i koji deluje na kracu podformulu.

Bez gubitka opštosti pretpostavicemo da je izabrana formula negativna i tipa $(\forall y)F(y)$. Izdvojimo, nadalje, predikate iz domena dejstva tog kvantifikatora i markirajmo pozicije u tim predikatima gde se pojavljuje promenljiva y. Za svaki simetrični predikat markiracemo i poziciju gde je dozvoljena komutacija. Zatim potražimo sva pojavljivanja izdvojenih predikatskih slova u ostatku formule G. Po pronalaženju predikata izdvajamo slobodne promenljive i osnovne terme smeštene na markiranim pozicijama i medju njima biramo one koje nisu slobodne u formuli F. Slobodne promenljive iz F ćemo uključiti jedino u slučaju kad nijedan od predikata nije simetričan. Ako su promenljive na markiranim pozicijama vezane ovaj postupak primenjujemo na sledeći kvantifikator. Izborom terma $t_1,t_2,\dots,t_n$ odredili smo i logički valjanu formulu sa kojom vršimo transformaciju u cilju dobijanja novog podcilja koji je jači od zadanog podcilja G.

Ovaj novi podcilj se šalje u blok trivijalnih transformacija gde se podcilj eventualno razbija na nove i pojednostavljuje kao što je opisano u odeljku 5. i zatim pokušavamo utvrditi tačnost ugradjenim rutinama. Za sve podciljeve na lišću drveta dokaza ponavljamo postupak sve dok svi podciljevi ne budu dokazani ili dok više ništa od navedenog ne možemo izvesti.

Data heuristika je naročito efikasna u slučajevima kada su predikati relacije ekvivalencije ili su bar simetrični. Ovo je zasnovano na ideji da mora postojati interakcija predikata date podformule i onih koji su smešteni u ostatku formule. Simetričnost nekog predikata je posebno notirana u dokazivaču u sistemu GRAPH tako što predstavlja deo sistemskog znanja te se ne mora ponovo dokazivati u toku dokaza glavnog cilja.

Ilustrujmo gornje izlaganje primerom.

Primer 5. Nastavimo sa posmatranjem formule iz primera 3. i primera 4.

$$\neg X1=X2\wedge(\forall X)((\forall Y)X=Y\vee(\exists Z)R1(X,Y)) \Rightarrow (\exists Y1)R1(X3,Y1)$$

U drvetu formule tražimo čvorove sa negativno označenim univerzalnim kvantifikatorom, ili pozitivno označenim egzistencijalnim kvantifikatorom. U ovom primeru postoje tri takva čvora:

$\forall X;-$ $\forall Y;-$ $\exists Y1;+$.

Čvor $\forall Y;-$ će se razmatrati prvo. Jedini predikat napadnut od strane izdvojenog kvantifikatora je =. Kako je = simetričan predikat i usto binarni markiraćemo mu obe pozicije. Sada tražimo = u ostatku formule. Budući da se X1 i X2 pojavljuju kao slobodne promenljive, a nemaju pojavljivanja u datoj podformuli odredjujemo X1 i X2 kao tražene terme. Odgovarajuća valjana formula je oblika:

$$(\forall Y)F(Y) \dashrightarrow F(X1) \wedge F(X2).$$

Po primeni te valjane fomule rezultujuća transformisana formula je

$$\neg X1=X2 \wedge (\forall X)(X=X1 \wedge X=X2 \vee (\exists Z)R1(X,Z)) \Rightarrow (\exists Y1)R1(X3,Y1).$$

Nakon pokušaja primene bloka trivijalnih transformacija i utvrdjivanja tačnosti podcilja, podcilj ostaje neizmenjen. Tako ponovo probamo dalju eliminaciju kvantifikatora. Sada su kandidati sledeći čvorovi:

$\forall X;-$ i $\exists Y1;+$.

Izabran je $\exists Y1;+$. Markiramo predikat sa slovom R1 i zatim oba njegova mesta budući da je simetričan. Kako su markirana mesta u ostatku formule popunjena samo vezanim promenljivim prelazimo na sledećeg kandidata tj. na čvor sa obeležjem $\forall X;-$. Predikatska slova u podformuli su = i R1. Oba su simetrična te markiramo oba mesta. Na relevantnim mestima u ostatku formule su termi X1, X2, X3, Y1. Promenljiva Y1 je vezana ,a X1 i X2 su slobodne u datoj podformuli. Time dobijamo valjanu formulu:

$$(\forall X)F(X) \dashrightarrow F(X3)$$

kojom transformišemo podcilj. Rezultat je:

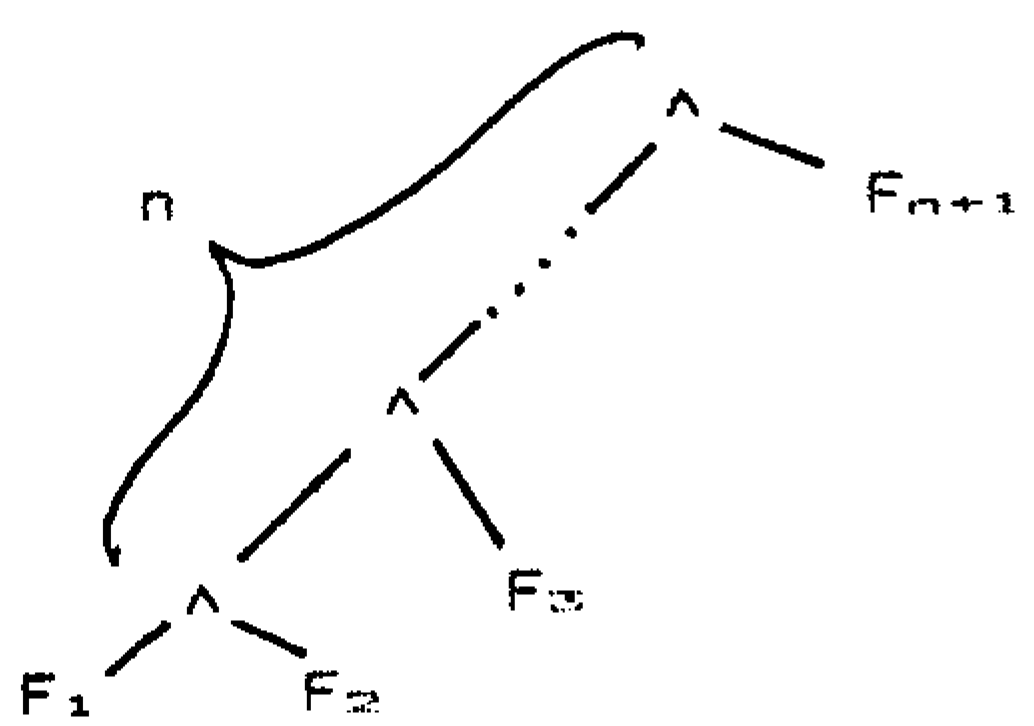
6.3 HEURISTIKA ZA MANIPULACIJU SA n-ARNIM KONJUNKCIJAMA I DISJUNKCIJAMA

Prema sadašnjem stanju implementacije u sistemu GRAPH operacije konjunkcije i disjunkcije se posmatraju kao binarne logičke operacije. To može rezultovati neuspelom unifikacijom, na primer, kada pokušavamo ujednačavanje strukture logičkih operacija strane leme i podformule tekućeg cilja samo zbog toga što neko poddrvo sa n konjunkcija (ili dualno n disjunkcija) nije bilo odgovarajućeg oblika.

Sistem, naravno, omogućava korisniku korišćenje valjanih formula, čime on biranjem primene onih kojima se iskazuje asocijativnost i komutativnost tih operacija transformiše tekući podcilj na željeni oblik.

U automatskom radu to nije predvidjeno.

Jedno poboljšanje bi bilo sprovodjenje kanonizacije svake formule podcilja kao i lema, definicija i aksioma tako što bi se u slučaju pojavljivanja više konjunkcija uvek generisalo krajnje levo drvo, kao na slici 12.



Slika 12

Ovo je analogno posmatranju celog tog poddrveta kao n-arne operacije konjunkcije.

Sada je problem sveden na nešto niži nivo, naime unifikacija ne mora da uspe ukoliko redosled formula na liscu te n-arne konjunkcije, (prema slici 12. to su F_1, F_2,... ,F_{n+1}, nije odgovarajuci.

Jasno je da uzimanjem u obzir svih permutacija dobijamo kompletnost postupka, ali i nepotrebno gubljenje vremena u vecem broju slučajeva, što pokazuje praksa.

Bledsoe je u <9> u interaktivnom radu dokazivača predložio da se permutacije ne generišu, već da se korisniku da mogucnost da i-ti konjunkt postavi na pocetak, čime nizom takvih transformacija korisnik može da odabere relevantni redosled konjunkta.

U cilju dalje automatizacije generisanja redosleda konjunkta, predlaže se sledeća strategija .

Pretpostavimo da Fi_1, Fi_2,..., Fi_k gde je $i_j \in \{ 1,2,...n \}$ predstavljaju elementarne formule u kojima učestvuje binarni predikat recimo R. U tom slučaju često se u dokazivanju pojavljuju neki specijalni redosledi tih elementarnih formula u odnosu na terme koji učestvuju kao argumenti predikata R.

Pomenimo sledeće slučajeve:

1) **m-PUT**

$R(t_1,t_2)$, $R(t_2,t_3)$,..., $R(t_m,t_{m+1})$

2) **m-ZVEZDA**

$R(t,t_1)$, $R(t,t_2)$,...,$R(t,t_m)$

3) **m-CIKL**

$R(t_1,t_2)$, $R(t_2,t_3)$,..., $R(t_m,t_1)$

Da bi postupak bio efikasniji dobro je na ovom nivou dozvoliti i korišćenje simetričnosti relacije R u slučaju da je R simetrična relacija,tako što sistem za svaki predikat zna da li je simetričan, u toku generisanja odgovarajuće strukture m-puta, m-zvezde ili m-cikla.

Analogan redosled se može zahtevati i za ternarne predikate gde se redosled odredjuje prema odgovarajuca dva mesta. Simetricnost po ta dva mesta, ukoliko je ispunjena, ukljucujemo kao u gornjem slucaju.

U interaktivnom radu se može na nivou komande zahtevati transformisanje formule podcilja tako da data relacija sa više pojavljivanja u n-arnoj konjunkciji ili disjunkciji u podcilju ima strukturu puta, zvezde ili cikla. Pritom bi sistem naravno utvrdjivao da li je to moguce postići.

U automatskom radu bi se poboljšao nivo unifikacije ukoliko bi se, po konstatovanju da u podformuli leme ili definicije za neku relaciju postoji struktura odredjenog tipa (put,zvezda ili cikl), transformisao tekući podcilj sa kojim vršimo ujednačavanje tako da i on ima tu strukturu. Ovo može da se uradi bilo po neuspeloj unifikaciji ili pre pokušaja unifikacije.

7. PRIMERI DOKAZA, ZAKLJUČAK I SMERNICE ZA DALJE USAVRŠAVANJE

Navedimo sada primere dokaza koji ilustruju rad dokazivača opisanog u dosadašnjem izlaganju i na sadašnjem nivou implementacije. Primeri dokaza su konkretni štampani dokumenti dobijeni radom na sistemu GRAPH.

Kao što je ranije pomenuto rečenice koje ulaze u dokazivač se saopštavaju sistemu na jeziku pravi GTCL koji je formalizovani podskup prirodnog engleskog jezika, a sistem ih prevodi u oblik kvantifikatorske formule i na tom nivou kreira dokaz. U sistemu postoji i inverzni prevodilac kojim se svaka rečenica sa nivoa kvantifikatorske formule može prevesti na engleski tekst tj. na nivo pravog GTCL.

Napomenimo još da je u predstavljanju podciljeva u obliku kvantifikatorskih formula u štampanim dokumentima korišćena sledeća konvencija o zapisu nekih logičkih opearacija i aritmetičkih relacija u skladu sa mogućnostima grafičkih simbola na štampaču.

$\wedge$	se prikazuje pomoću .AND.
$\vee$	se prikazuje pomoću .OR.
$\neg$	se prikazuje pomoću NOT.
$(\forall X)$	se prikazuje pomoću (ALL X)
$(\exists X)$	se prikazuje pomoću (EXT X)
$\leq$	se prikazuje pomoću <=
$\geq$	se prikazuje pomoću >=

Komunikacija korisnika sa računarom u toku izvodjenja dokaza je vodjena komandama opisanim u odeljku 5.1. Svaka komanda je pisana velikim slovima, a završava se pauzom iza koje sledi tačka. Komentari koji predstavljaju poruke sistema o toku izvodjenja dokaza su pisani malim slovima.

U listingu dokaza iz tehničkih razloga se preskače broj dva u numerisanju podciljeva.

Slede spiskovi aksioma, lema, definicija i valjanih formula sa kojima je sistem raspolagao u izvodjenju dokaza.

SHOW AXIOMS .

```
     1.
NOT.S1(X,X,U)
     2.
S1(X,Y,U)<=>S1(Y,X,U)
     3.
S1(X,Y,U).AND.S1(X,Y,V)=>U=V
     4.
S1(X,Y,U).AND.S1(X1,Y1,U)=>(X=X1.AND.Y=Y1).OR.(X=Y
1.AND.Y=X1)
     5.
(EXT N)(N>=1.AND.Q1(N))
     6.
Q1(N1).AND.Q1(N2)=>N1=N2
     7.
(EXT M)(M>=1.AND.Q2(M))
     8.
Q2(M1).AND.Q2(M2)=>M1=M2
     9.
K=K
    10.
K=L<=>L=K
    11.
K=L.AND.L=M=>K=M
    12.
NOT.0=K+1
    13.
K+1=L+1<=>K=L
    14.
K+0=K
    15.
K+(L+1)=(K+L)+1
    16.
K*0=0
    17.
K*(L+1)=K*L+K
    18.
K-L=M<=>K=M+L
    19.
K#L<=>NOT.K=L
    20.
K<L<=>(EXT M)(M#0.AND.K+M=L)
    21.
K<=L<=>K<L.OR.K=L
    22.
K>L<=>L<K
    23.
K>=L<=>L<=K
    24.
L@K<=>(EXT M)K=M*L
    25.
S2(X,Y,K+1)<=>(EXT Z)(S2(X,Z,K).AND.(EXT U)S1(Z,Y,
U))
```

```
SHOW DEFINITIONS .
    9.
S1(X,Y,U)
   10.
R1(X,Y)<=>(EXT U)S1(X,Y,U)
   11.
Q1(N)
   12.
Q2(M)
   13.
Q3(X)<=>(ALL Y)(NOT.R1(X,Y))
   14.
R1A1(X,Y)<=>X#Y.AND.NOT.R1(X,Y)
   15.
J1=N<=>Q1(N)
   16.
S2(X,Y,K)
   17.
R2(X,Y)<=>(EXT K)S2(X,Y,K)
   18.
P1<=>(ALL X)(ALL Y)(X#Y=>R1(X,Y))
   19.
P2<=>(ALL X)(ALL Y)R2(X,Y)
   20.
Q4(X)<=>1<=X.AND.(ALL N)(Q1(N)=>X<=N)
   21.
Q5(U)<=>1<=U.AND.(ALL M)(Q2(M)=>U<=M)
   22.
R3(X,U)<=>(EXT Y)S1(X,Y,U)
   23.
R4(U,V)<=>(EXT X)(R3(X,U).AND.R3(X,V))
   24.
P3<=>(ALL X)(ALL Y)NOT.R1(X,Y)
   25.
P4<=>Q1(1)
   26.
P5<=>(EXT X)(EXT Y)(EXT Z)(R1(X,Y).AND.R1(X,Z).AND
.R1(Y,Z))
   27.
R6<=>NOT.P2
   28.
S3(X,Y,K)<=>S2(X,Y,K).AND.(ALL L)(L<K=>NOT.S2(X,Y,
L))
   29.
Q6(K)<=>P2.AND.(EXT X)(EXT Y)S3(X,Y,K).AND.(ALL X1
)(ALL Y1)(ALL L)(S3(X1,Y1,L)=>L<=K)
   30.
R5(X,K)<=>(EXT Y)S3(X,Y,K).AND.(ALL Z)(ALL L)(S3(X
,Z,L)=>L<=K)
   31.
Q7(K)<=>P2.AND.(EXT X)R5(X,K).AND.(ALL Y)(ALL L)(R
5(Y,L)=>L>=K)
   32.
Q8(X)<=>(ALL K)(R5(X,K)=>Q7(K))
   33.
R6(X,K)
   34.
Q9(K)<=>(ALL X)R6(X,K)
   35.
P7<=>(EXT K)Q9(K)
   36.
P8<=>P2.AND.Q9(2)
```

```
SHOW LEMMAS .
     1.
R1(X,Y)<=>R1(Y,X)
     2.
S2(X,Y,0)<=>X=Y
     3.
S2(X,Y,1)<=>R1(X,Y)
     4.
R2(X,Y)<=>R2(Y,X)
     5.
R2(X,Y).AND.R2(Y,Z)=>R2(X,Z)
     6.
R4(U,V)=>R4(V,U)
     7.
P4<=>(ALL X)(ALL Y)X=Y
     8.
S2(X,Y,K)=>S2(Y,X,K)
     9.
S3(X,Y,K)=>S3(Y,X,K)
    10.
S3(X,Y,1)<=>R1(X,Y)
    11.
NOT.(ALL X)(ALL Y)R2(X,Y)
    12.
S2(X,Y,K)<=>(K=0.AND.X=Y).OR.(K>0.AND.(EXT Z)(S2(X
,Z,K-1).AND.R1(Z,Y)))
```

SHOW VALID FORMULAS .

```
P1=>(P2=>P3)<=>P1.AND.P2=>P3

P1=>(P2<=>P3)<=>(P1.AND.P2=>P3).AND.(P1.AND.P3=>P2
)

P1=>P2.AND.P3<=>(P1=>P2).AND.(P1=>P3)

NOT.NOT.P1<=>P1

P1.OR.(P2.AND.P3)<=>P1.OR.P2.AND.P1.OR.P3

(P1.AND.P2).OR.P3<=>P1.OR.P3.AND.P2.OR.P3

P1.AND.(P2.OR.P3)<=>(P1.AND.P2).OR.(P1.AND.P3)

P1.OR.P2.AND.P3<=>(P1.AND.P3).OR.(P2.AND.P3)

NOT.(P1.AND.P2)<=>NOT.P1.OR.NOT.P2

NOT.(P1.OR.P2)<=>NOT.P1.AND.NOT.P2

P1<=>P2<=>(P1=>P2).AND.(P2=>P1)

P1.AND.(P2.AND.P3)<=>P1.AND.P2.AND.P3

P1.OR.(P2.OR.P3)<=>P1.OR.P2.OR.P3

P1.OR.P2<=>P2.OR.P1

P1.AND.P2<=>P2.AND.P1

P1=>P2<=>NOT.P1.OR.P2

NOT.P1=>NOT.P2<=>P2=>P1

P1=>P2=>P3<=>NOT.P1.OR.P2=>P3

P1=>P2<=>NOT.P2=>NOT.P1
```

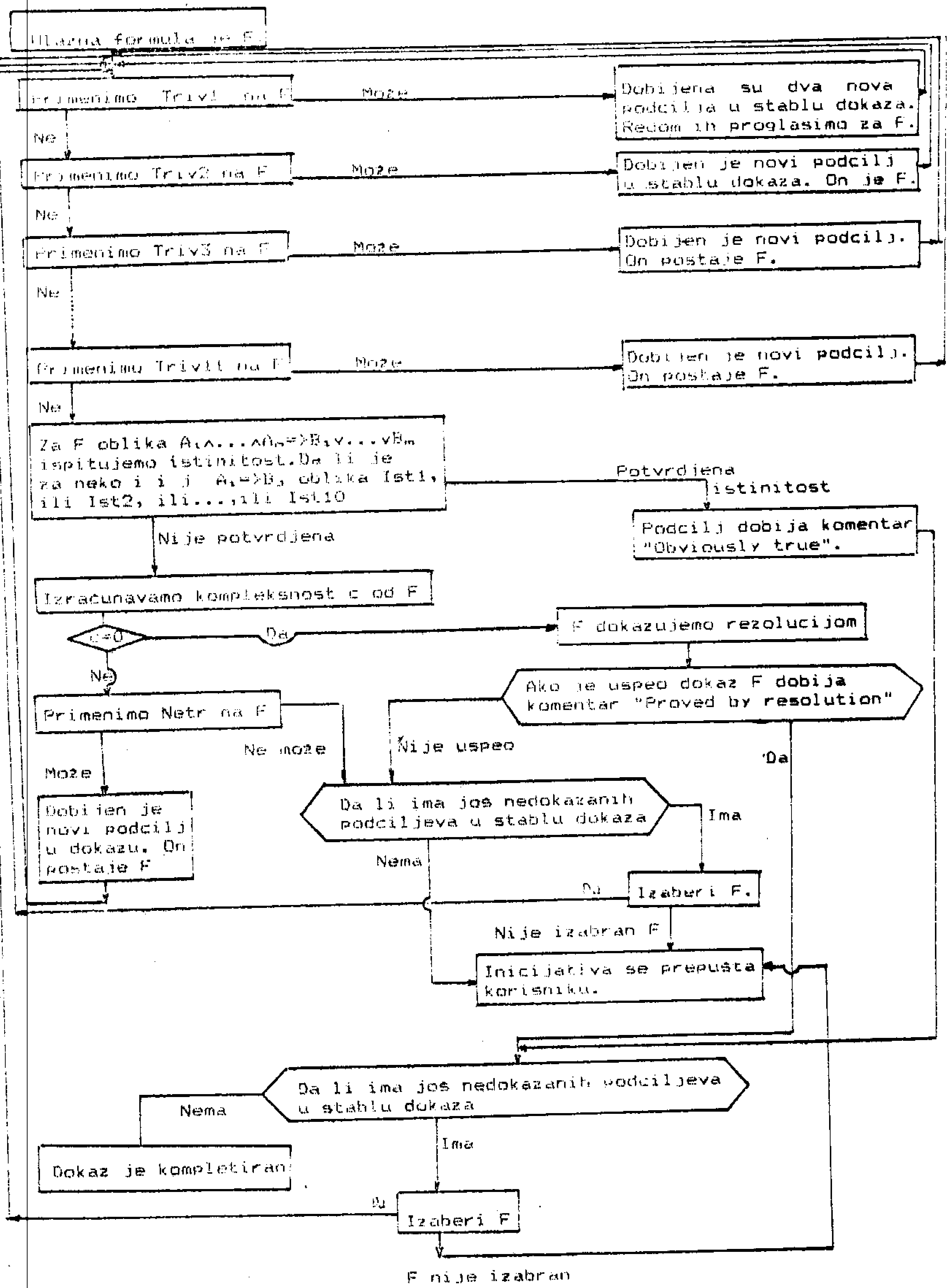
Rečenicu koja predstavlja ulaz u dokazivač, zadavala sam pod imenom koje počinje nizom slova SIR, a cifre u nastavku imena služe da bi se rečenice razlikovale medjusobno. Drvo dokaza se pamti pod imenom koje počinje slovom T, a u nastavku sledi niz cifara iz sufiksa recenice čiji je to dokaz. Na primer, za rečenicu pod imenom SIR3, odgovarajuće drvo dokaza će imati ime T3.

Drvo dokaza se prikazuje tako što se sinovi pomeraju dva mesta udesno ispod oca. U slučaju da je stablo dokaza razgranato i veliko po dubini, sistem ga ne reprodukuje u celini. Počev od čvora kome je za zapis kvantifikatorske formule tekućeg podcilja ostalo manje od 20 mesta po širini, sistem ispisuje "...". Taj deo dokaza možemo prikazati pomeranjem ukazatelja na neki čvor u drvetu dokaza iznad tog, a zatim štampanjem tako definisanog poddrveta.

Dokazi su izvedeni u automatskom režimu rada sa obeležjem 1. Kod jednostavnijih primera sistem je u potpunosti kompletirao dokaz bez intervencije čoveka, prema ugradjenoj koncepciji rada. Naravno, u komplikovanim dokazima korisnik mora da interveniše brisanjem dela izvedenog dokaza i da preusmeri dalji rad sistema po sopstvenom nahodjenju, najčešće zadavanjem pogodnog medjucilja ili transformisanjem tekućeg podcilja u ekvivalentan oblik. Posle intervencije korisnika sistem ponovo nastavlja rad prema ugradjenoj strategiji vodjenja dokaza.

Da bismo lakše pratili dokaze prikazaćemo algoritam automatskog rada sistema u režimu sa obeležjem 1 pomoću blok sheme. Koristićemo skraćenice Triv1,...,Triv11 za trivijalne transformacije opisne u odeljku 5.0.1, zatim Netr će označavati netrivijalnu transformaciju koja je opisana u odeljku 6.1, a Ist1,..., Ist10 predstavljaju 10 zadatih oblika po jednog konjunkta i disjunkta respektivno u levoj i desnoj strani od centralne implikacije tekućeg podcilja kojem ispitujemo tačnost, navedenih u odeljku 5.0.2.

F označava tekući podcilj u drvetu dokaza koji se trenutno obradjuje. U početku je F rečenica koju dokazujemo.



Tekuća formula je F.
Primenimo Triv1 na F
Može
Dobijena su dva nova podcilja u stablu dokaza. Redom ih proglasimo za F.
Ne
Primenimo Triv2 na F
Može
Dobijen je novi podcilj u stablu dokaza. On je F.
Ne
Primenimo Triv3 na F
Može
Dobijen je novi podcilj. On postaje F.
Ne
Primenimo Trivit na F
Može
Dobijen je novi podcilj. On postaje F.
Ne
Za F oblika A₁∧...∧Aₙ=>B₁∨...∨Bₘ ispitujemo istinitost. Da li je za neko i i j Aᵢ=>Bⱼ oblika Ist1, ili Ist2, ili...,ili Ist10
Potvrdjena istinitost
Podcilj dobija komentar "Obviously true".
Nije potvrdjena
Izracunavamo kompleksnost c od F
c=0
Da
F dokazujemo rezolucijom
Ne
Ako je uspeo dokaz F dobija komentar "Proved by resolution"
Primenimo Netr na F
Ne može
Nije uspeo
Da
Može
Da li ima jos nedokazanih podciljeva u stablu dokaza
Ima
Dobijen je novi podcilj u dokazu. On postaje F
Nema
Da
Izaberi F.
Nije izabran F
Inicijativa se prepušta korisniku.
Da li ima jos nedokazanih podciljeva u stablu dokaza
Nema
Ima
Dokaz je kompletiran
Da
Izaberi F
F nije izabran

Predjimo sada na primere dokaza.

Primer 7. Na ulazu je recenica SIR1:

"IF GRAPH IS COMPLETE THEN GRAPH IS TOTALLY DISCONNECTED IN COMPLEMENT" .

Prevod sistema u oblik kvantifikatorske formule daje:

P1=>P3A1

Podsetimo se da predikat P1 oznacava pojam "graf je kompletan", a predikat P3 oznacava pojam "graf je totalno nepovezan", kao sto je navedeno u definicijama datim u poglavlju 3.

Nastavak A1 u sufiksu predikatskog slova P3 odnosi se na operaciju komplementa grafa. Predikat P3A1 oznacava pojam komplement grafa je totalno nepovezan.

Dokaz izgleda ovako:

```
TYPE T1 .
1       P1=>P3A1
 3        P1=>(ALL X)(ALL Y)NOT.R1A1(X,Y)
  4         P1=>(ALL Y)NOT.R1A1(X,Y)
   5          P1=>NOT.R1A1(X,Y)
    6           P1=>NOT.(X#Y.AND.NOT.R1(X,Y))
     7            P1=>NOT.X#Y.OR.NOT.NOT.R1(X,Y)
      8             P1=>NOT.X#Y.OR.R1(X,Y)
       9              P1=>X=Y.OR.R1(X,Y)
        10              (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1))=>X=Y.OR.
                        R1(X,Y)
         11  *            (ALL X1)(ALL Y1)(NOT.X1=Y1=>R1(X1,Y1))=>X
                          =Y.OR.R1(X,Y) Proved by resolution
```

Objasnicemo sada detalje izvodjenja dokaza.

Prelaz sa podcilja 1 na podcilj sa rednim brojem 3 postignut je izvodjenjem netrivijalne transformacije i to zamenom P3A1 pomoću definicije. Zamena se izvodi tako što sistem ugradjenom heuristikom za izbor relevantne definicije odnosno leme (opisane u 6.1) zaključi da treba da izvrši zamenu P3A1 po definiciji. U sistemu u datoteci definicija postoji definicija predikata P3 koja u obliku kvantifikatorske formule izgleda ovako:

$$P3 \Leftrightarrow (\forall X)(\forall Y)\neg R1(X,Y)$$

Sistem dodaje nastavak grafovske operacije komplementiranja svim nazivima predikatskih slova i zatim vrši transformaciju smene definienduma definiensom.

Prelaz sa podcilja 3 na 4 i od 4 na 5 postignut je primenom izvodjenja trivijalne transformacije kojom se uklanjaju univerzalni kvantifikatori koji su vrhovne logičke operacije u desnoj strani centralne implikacije podcilja.

Podcilj 6 je ponovo rezultat izvodjenja netrivijalne transformacije. Ovog puta korišćena je definicija koja u obliku kvantifikatorske formule izgleda ovako:

$$R1A1(X,Y) \Leftrightarrow X \neq Y \wedge \neg R1(X,Y)$$

Podcilj 7 je proizvod primene trivijalne transformacije na podcilj 6 kojom se negacija primenom tautologije

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

gura ka podformulama kroz konjunkciju.

Podcilj 8 takodjer je rezultat primene trivijalne transformacije gde koristimo tautologiju:

$$\neg\neg A \longrightarrow A$$

kojom eliminišemo dvojnu negaciju.

Podcilj 9 je dobijen primenom trivijalne transformacije kojom se podformula oblika leve strane zamenjuje desnom stranom po ujednačavanju promenljivih pomoću pravila

$$\neg x \neq y \longrightarrow x = y$$

Podcilj 10 je dobijen primenom netrivijalne transformacije i to zamenom P1 pomoću definicije čija kvantifikatorska formula glasi:

$$P1 \Leftrightarrow (\forall X)(\forall Y)(X \neq Y \Rightarrow R1(X,Y))$$

U podcilju 11 ugradjenon inteligentnom strategijom vodjenja dokaza pošto je kompleksnost podcilja 0 a ugradjenim rutinama nije potvrdjena tačnost podcija predvidjeno je da se dokaz kompletira rezolucijskim dokazivačem. Sistem je po obavljenom poslu uz taj podcilj štampao i odgovarajuću poruku.

Primer 8. Zadata je rečenica SIR3:

"If X≠Y and X is isolated then X and Y are adjacent in complement" .

Njena kvantifikatorska formula je:

$$X \neq Y \land Q3(X) \Rightarrow R1A1(X,Y).$$

Dokaz će biti reprodukovan u celosti onako kako je tekao na računaru. Kao što se može videti iz fotokopije štampanog dokumenta na sledećoj strani, izvodjenje dokaza sa imenom T3 je kompletirano bez intervencije korisnika i bez upotrebe rezolucijskog dokazivača.

Tekst pisan malim slovima je komentar sistema o toku izvodjenja dokaza.

Podcilj 3 je dobijen primenom netrivijalne transformacije. Nadalje, podciljevi 4 i 5 su rezultat trivijalne transformacije razbijanja na podciljeve. Podcilj 6 je dobijen iz podcilja 5 primenom netrivijalne transformacije. Istinitost podciljeva 4 i 6 je potvrdjena automatski, ugradjenim modulom za ispitivanje tačnosti.

```
TYPE TRANSLA SIR3 .
X#Y.AND.Q3(X)=>R1A1(X,Y)


Type Your next command, please!

CREATE T3 PROOF SIR3 .

Considering subgoal nr.    1
1       X#Y.AND.Q3(X)=>R1A1(X,Y)

Considering subgoal nr.    1

Subgoal nr.    3 created

Considering subgoal nr.    3

Subgoals nrs.    4 and    5 created

Considering subgoal nr.    5

Considering subgoal nr.    4
1       X#Y.AND.Q3(X)=>R1A1(X,Y)
  3       X#Y.AND.Q3(X)=>X#Y.AND.NOT.R1(X,Y)
    4       X#Y.AND.Q3(X)=>X#Y
    5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)

Considering subgoal nr.    4

Considering subgoal nr.    5
1       X#Y.AND.Q3(X)=>R1A1(X,Y)
  3       X#Y.AND.Q3(X)=>X#Y.AND.NOT.R1(X,Y)
    4   *   X#Y.AND.Q3(X)=>X#Y Obviously true
    5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)

Subgoals proven:
    4

Subgoals not proven:
    5

Shall I continue? From which subgoal ?    5

Considering subgoal nr.    5
5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)

Considering subgoal nr.    5

Subgoal nr.    6 created

Considering subgoal nr.    6
5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)
  6       X#Y.AND.(ALL Y1)NOT.R1(X,Y1)=>NOT.R1(X,Y)

Considering subgoal nr.    6
5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)
  6   *   X#Y.AND.(ALL Y1)NOT.R1(X,Y1)=>NOT.R1(X,Y) Obviously true

Subgoals proven:
    4    6

Subgoals not proven:

The proof has been completed !
```

```
Type Your next command, please!

MOVE ROOT T3 TO POINT 1 .

Type Your next command, please!

TYPE T3 .
1        X#Y.AND.Q3(X)=>R1A1(X,Y)
  3        X#Y.AND.Q3(X)=>X#Y.AND.NOT.R1(X,Y)
    4   *   X#Y.AND.Q3(X)=>X#Y Obviously true
    5       X#Y.AND.Q3(X)=>NOT.R1(X,Y)
      6   *   X#Y.AND.(ALL Y1)NOT.R1(X,Y1)=>NOT.R1(X,Y) Obviously t
              rue
```

Primer 9. Recenica sa imenom SIR21 zadata je sa: "If the graph is totally disconnected then graph has not a triangle" .

U dokazu pod imenom T21, kao što se vidi iz štampanog dokumenta na sledećoj strani, prelaz iz podcilja 1 na 3 i prelaz od podcilja 11 na 12 je izveden primenom netrivijalnih transformacija, dok su ostali podciljevi proizvod primene odgovarajućih trivijalnih transformacija. Dokaz podcilja 12 dovršen je primenom rezolucijskog dokazivača.

Kompletan postupak dokazivanja na sistemu i sam dokaz biće prikazani reprodukovanjem štampanog dokumenta na narednim stranama.

```
TYPE SIR21 .
IF GRAPH IS TOTALLY DISCONNECTED THEN GRAPH HAS NOT
 A TRIANGLE

Type Your next command, please!

TYPE TRANSLA SIR21 .
P3=>NOT.P5

Type Your next command, please!

CREATE T21 PROOF SIR21 .

Considering subgoal nr.     1
1       P3=>NOT.P5

Considering subgoal nr.     1

Subgoal nr.     3 created

Considering subgoal nr.     3

Subgoal nr.     4 created

Considering subgoal nr.     4

Subgoal nr.     5 created

Considering subgoal nr.     5

Subgoal nr.     6 created

Considering subgoal nr.     6

Subgoal nr.     7 created

Considering subgoal nr.     7

Subgoal nr.     8 created

Considering subgoal nr.     8

Subgoal nr.     9 created

Considering subgoal nr.     9

Subgoal nr.    10 created

Considering subgoal nr.    10

Subgoal nr.    11 created

Considering subgoal nr.    11
1       P3=>NOT.P5
  3       P3=>NOT.(EXT X)(EXT Y)(EXT Z)(R1(X,Y).AND.R1(X,Z).AND.R1(
          Y,Z))
    4       P3=>(ALL X)NOT.(EXT Y)(EXT Z)(R1(X,Y).AND.R1(X,Z).AND.R
            1(Y,Z))
      5       P3=>(ALL X)(ALL Y)NOT.(EXT Z)(R1(X,Y).AND.R1(X,Z).AND
              .R1(Y,Z))
        6       P3=>(ALL X)(ALL Y)(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).A
                ND.R1(Y,Z))
          7       P3=>(ALL Y)(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).AND.R1
                  (Y,Z))
            8       P3=>(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).AND.R1(Y,Z)
                    )
              9       P3=>NOT.(R1(X,Y).AND.R1(X,Z).AND.R1(Y,Z))
                10      P3=>NOT.(R1(X,Y).AND.R1(X,Z)).OR.NOT.R1(Y,Z
                        )
                  11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(
                          Y,Z)
```

```
Subgoals proven:

Subgoals not proven:
   11

Shall I continue? From which subgoal ?    11

Considering subgoal nr.    11
11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(Y,Z)

Considering subgoal nr.    11

Subgoal nr.   12 created

Considering subgoal nr.    12
11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(Y,Z)
  12      (ALL X1)(ALL Y1)NOT.R1(X1,Y1)=>NOT.R1(X,Y).OR.NOT.R1(X,Z)
          .OR.NOT.R1(Y,Z)

Considering subgoal nr.    12

Trying to prove subgoal nr.    12 by resolution
(ALL X1)(ALL Y1)NOT.R1(X1,Y1)=>NOT.R1(X,Y).OR.NOT.
R1(X,Z).OR.NOT.R1(Y,Z)

Clauses :
NOT.R1(X1,Y1)&
R1(0501,0502)&
R1(0501,0503)&
R1(0502,0503)&

The  number  of clauses arising from the negation of the sentence
is equal to      4

Considering clause nr.    1

The proof has been completed !

The  refutation  proof  is  given  by  the  following sequence of
clauses:

Central clause:
NOT.R1(X1,Y1)&

Auxiliary clause:
R1(0501,0502)&

Resolution substitution:
0501/X1,0502/Y1&

Empty clause:

The proof is printed.
11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(Y,Z)
  12  *   (ALL X1)(ALL Y1)NOT.R1(X1,Y1)=>NOT.R1(X,Y).OR.NOT.R1(X,Z)
          .OR.NOT.R1(Y,Z) Proved by resolution

Subgoals proven:
   12

Subgoals not proven:

The proof has been completed !


Type Your next command, please!

TYPE T21 .
11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(Y,Z)
  12  *   (ALL X1)(ALL Y1)NOT.R1(X1,Y1)=>NOT.R1(X,Y).OR.NOT.R1(X,Z)
          .OR.NOT.R1(Y,Z) Proved by resolution


Type Your next command, please!
```

```
MOVE T21 TO POINT 1 .

Type Your next command, please!

TYPE T21 .
1       P3=>NOT.P5
  3       P3=>NOT.(EXT X)(EXT Y)(EXT Z)(R1(X,Y).AND.R1(X,Z).AND.R1(
          Y,Z))
    4       P3=>(ALL X)NOT.(EXT Y)(EXT Z)(R1(X,Y).AND.R1(X,Z).AND.R
            1(Y,Z))
      5       P3=>(ALL X)(ALL Y)NOT.(EXT Z)(R1(X,Y).AND.R1(X,Z).AND
              .R1(Y,Z))
        6       P3=>(ALL X)(ALL Y)(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).A
                ND.R1(Y,Z))
          7       P3=>(ALL Y)(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).AND.R1
                  (Y,Z))
            8       P3=>(ALL Z)NOT.(R1(X,Y).AND.R1(X,Z).AND.R1(Y,Z)
                    )
              9       P3=>NOT.(R1(X,Y).AND.R1(X,Z).AND.R1(Y,Z))
                10      P3=>NOT.(R1(X,Y).AND.R1(X,Z)).OR.NOT.R1(Y,Z
                        )
                  11      P3=>NOT.R1(X,Y).OR.NOT.R1(X,Z).OR.NOT.R1(
                          Y,Z)
                    12  *   (ALL X1)(ALL Y1)NOT.R1(X1,Y1)=>NOT.R1(X
                            ,Y).OR.NOT.R1(X,Z).OR.NOT.R1(Y,Z) Prove
                            d by resolution
```

Primer 10. Rečenica sa nazivom SIRT1 zadata je sa: "Point X and line U are incident iff there exists Y such that X and Y are joined by line U and for all Y1, Y2 (if X and Y1 are joined by line U and X and Y2 are joined by line U then Y1=Y2)" .

Dokaz rečenice SIRT1 je izveden i zapamćen pod imenom TT1. Sledi štampani izveštaj:

```
TYPE TRANSLA SIRT1 .
R3(X,U)<=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1
(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)

Type Your next command, Please!

CREATE TT1 PROOF SIRT .

The requested sentence is not in the memory.

Type Your next command, Please!

CREATE TT1 PROOF SIRT1 .

Considering subgoal nr.     1

Subgoals nrs.     3 and     4 created

Considering subgoal nr.     4

Considering subgoal nr.     3

Subgoals nrs.     5 and     6 created

Considering subgoal nr.     6

Subgoal nr.     7 created

Considering subgoal nr.     7

Subgoal nr.     8 created

Considering subgoal nr.     8

Considering subgoal nr.     5
1       R3(X,U)<=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).
        AND.S1(X,Y2,U)=>Y1=Y2)
  3       R3(X,U)=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U)
          .AND.S1(X,Y2,U)=>Y1=Y2)
    5       R3(X,U)=>(EXT Y)S1(X,Y,U)
    6       R3(X,U)=>(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1
            =Y2)
      7       R3(X,U)=>(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
        8       R3(X,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
  4       (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X
          ,Y2,U)=>Y1=Y2)=>R3(X,U)

Considering subgoal nr.     5

Subgoal nr.     9 created

Considering subgoal nr.     9
```

subgoal nr. ~~10 created~~

```
Considering subgoal nr.   10

Considering subgoal nr.    8

Subgoal nr.   11 created

Considering subgoal nr.   11

Subgoal nr.   12 created

Considering subgoal nr.   12
8       R3(X,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
  11      (EXT Y)S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
    12      S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)

Considering subgoal nr.   12

Considering subgoal nr.    4

Subgoal nr.   13 created

Considering subgoal nr.   13
4       (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y
        2,U)=>Y1=Y2)=>R3(X,U)
  13      (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X
          ,Y2,U)=>Y1=Y2)=>(EXT Y3)S1(X,Y3,U)

Considering subgoal nr.   13
1       R3(X,U)<=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).
        AND.S1(X,Y2,U)=>Y1=Y2)
  3       R3(X,U)=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U)
          .AND.S1(X,Y2,U)=>Y1=Y2)
    5       R3(X,U)=>(EXT Y)S1(X,Y,U)
      9       (EXT Y1)S1(X,Y1,U)=>(EXT Y)S1(X,Y,U)
        10  *  S1(X,Y1,U)=>(EXT Y)S1(X,Y,U) Obviously true
    6       R3(X,U)=>(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1
            =Y2)
      7       R3(X,U)=>(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
        8       R3(X,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
          11      (EXT Y)S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=
                  Y2)
            12      S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
  4       (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X
          ,Y2,U)=>Y1=Y2)=>R3(X,U)
    13  *  (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1
            (X,Y2,U)=>Y1=Y2)=>(EXT Y3)S1(X,Y3,U) Obviously true

Subgoals proven:
  10   13

Subgoals not proven:
  12

Shall I continue? From which subgoal ?


Type Your next command, please!

MOVE ROOT OF TT1 TO POINT 12 .
```

Prelaz od podcilja 12 na 14 je izveden komandom odbacivanja pretpostavke, a prelaz od podcilja 14 na 15 zamenom po aksiomi. Sem ovih intervencija korisnika, sistem je dokaz izveo samostalno.

```
14      S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2
  15      (X=X.AND.Y1=Y2).OR.(X=Y2.AND.Y1=X)=>Y1=Y2
    16      X=X.AND.Y1=Y2=>Y1=Y2
      20      Y1=Y2=>Y1=Y2
        21  *  Y2=Y2 Obviously true
    17      X=Y2.AND.Y1=X=>Y1=Y2
      18      Y1=Y2=>Y1=Y2
        19  *  Y2=Y2 Obviously true
14      S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2
  15      (X=X.AND.Y1=Y2).OR.(X=Y2.AND.Y1=X)=>Y1=Y2
    16      X=X.AND.Y1=Y2=>Y1=Y2
      20      Y1=Y2=>Y1=Y2
        21  *  Y2=Y2 Obviously true
    17      X=Y2.AND.Y1=X=>Y1=Y2
      18      Y1=Y2=>Y1=Y2
        19  *  Y2=Y2 Obviously true

Subgoals proven!
   10   13   19   21

Subgoals not proven!

The proof has been completed !


Type Your next command, please!

MOVE TT1 TO RQINT 1 .


Type Your next command, please!

TYPE TT1 .
1       R3(X,U)<=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).
        AND.S1(X,Y2,U)=>Y1=Y2)
  3       R3(X,U)=>(EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U)
          .AND.S1(X,Y2,U)=>Y1=Y2)
    5       R3(X,U)=>(EXT Y)S1(X,Y,U)
      9       (EXT Y1)S1(X,Y1,U)=>(EXT Y)S1(X,Y,U)
        10  *  S1(X,Y1,U)=>(EXT Y)S1(X,Y,U) Obviously true
    6       R3(X,U)=>(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1
            =Y2)
      7       R3(X,U)=>(ALL Y2)(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
        8       R3(X,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
          11      (EXT Y)S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=
                  Y2)
            12      S1(X,Y,U)=>(S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2)
              14      S1(X,Y1,U).AND.S1(X,Y2,U)=>Y1=Y2
                15      (X=X.AND.Y1=Y2).OR.(X=Y2.AND.Y1=X)=>Y1=Y2
                  16      X=X.AND.Y1=Y2=>Y1=Y2
                    20      Y1=Y2=>Y1=Y2
                      21  *  Y2=Y2 Obviously true
                  17      X=Y2.AND.Y1=X=>Y1=Y2
                    18      Y1=Y2=>Y1=Y2
                      19  *  Y2=Y2 Obviously true
  4       (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1(X
          ,Y2,U)=>Y1=Y2)=>R3(X,U)
    13  *  (EXT Y)S1(X,Y,U).AND.(ALL Y1)(ALL Y2)(S1(X,Y1,U).AND.S1
           (X,Y2,U)=>Y1=Y2)=>(EXT Y3)S1(X,Y3,U) Obviously true
```

Primer 11. Na ulazu u dokazivač je rečenica SIR11:
"If the graph is trivial then the graph is connected" .

U obliku kvantifikatorske formule rečenica glasi:

P4 => P2

Sledi reprodukovani dokaz datog tvrdjenja. U izvodjenju je korisnik intervenisao uvodjenjem medjucilja čime su iz podcilja 7 izvedeni podciljevi 8 i 9.

```
TYPE T11 .
1     P4=>P2
  3     P4=>(ALL X)(ALL Y)R2(X,Y)
    4     P4=>(ALL Y)R2(X,Y)
     5      P4=>R2(X,Y)
       6      P4=>(EXT K)S2(X,Y,K)
         7      (ALL X1)(ALL Y1)X1=Y1=>(EXT K)S2(X,Y,K)
           8      (ALL X1)(ALL Y1)X1=Y1=>S2(X,Y,O)
             10  *  (ALL X1)(ALL Y1)X1=Y1=>X=Y Obviously true
           9    *  S2(X,Y,O)=>(EXT K)S2(X,Y,K) Obviously true
```

Navešćemo još jedan primer dokaza kompletiran bez intervencije čoveka, dakle u potpunosti korišćenjem ugradjene inteligentne strategije vodjenja dokaza u sistemu.

Primer 12. Na ulazu u dokazivač zadata je rečenica SIR7:
"IF GRAPH IS COMPLETE AND GRAPH IS NOT TRIVIAL THEN GRAPH IS OF DIAMETER 1" .

Prevod sistema u oblik kvantifikatorske formule glasi:

$$P1 \wedge \neg P4 \Rightarrow Q6(1)$$

Sam dokaz je predstavljen na sledećoj strani, a korake u dokazu nećemo komentarisati.

TYPE T7 .

```
     P1.AND.NOT.P4=>Q6(1)
3      P1.AND.NOT.P4=>P2.AND.(EXT X)(EXT Y)S3(X,Y,1).AND.(ALL X1
       )(ALL Y1)(ALL L)(S3(X1,Y1,L)=>L<=1)
 4       P1.AND.NOT.P4=>P2.AND.(EXT X)(EXT Y)S3(X,Y,1)
  9        P1.AND.NOT.P4=>P2
   11  *     P2.AND.NOT.P4=>P2 Obviously true
  10       P1.AND.NOT.P4=>(EXT X)(EXT Y)S3(X,Y,1)
   12        (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1)).AND.NOT.P4=>(EXT
              X)(EXT Y)S3(X,Y,1)
    13         (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1)).AND.NOT.P4=>(E
               XT X)(EXT Y)R1(X,Y)
     14          (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1)).AND.NOT.(ALL
                  X2)(ALL Y2)X2=Y2=>(EXT X)(EXT Y)R1(X,Y)
      15           (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1)).AND.(EXT X
                   2)NOT.(ALL Y2)X2=Y2=>(EXT X)(EXT Y)R1(X,Y)
       16            (ALL X1)(ALL Y1)(X1#Y1=>R1(X1,Y1)).AND.(EXT
                      X2)(EXT Y2)NOT.X2=Y2=>(EXT X)(EXT Y)R1(X,Y
                     )
        17             (ALL X1)(ALL Y1)(NOT.X1=Y1=>R1(X1,Y1)).AN
                       D.(EXT X2)(EXT Y2)NOT.X2=Y2=>(EXT X)(EXT
                       Y)R1(X,Y) Proved by resolution
 5       P1.AND.NOT.P4=>(ALL X1)(ALL Y1)(ALL L)(S3(X1,Y1,L)=>L<=
         1)
  6        P1.AND.NOT.P4=>(ALL Y1)(ALL L)(S3(X1,Y1,L)=>L<=1)
   7         P1.AND.NOT.P4=>(ALL L)(S3(X1,Y1,L)=>L<=1)
    8          P1.AND.NOT.P4=>(S3(X1,Y1,L)=>L<=1)
     18          (ALL X)(ALL Y)(X#Y=>R1(X,Y)).AND.NOT.P4=>(S3(X1
                 ,Y1,L)=>L<=1)
      19           (ALL X)(ALL Y)(X#Y=>R1(X,Y)).AND.NOT.(ALL X2)
                   (ALL Y2)X2=Y2=>(S3(X1,Y1,L)=>L<=1)
       20            (ALL X)(ALL Y)(X#Y=>R1(X,Y)).AND.(EXT X2)NO
                     T.(ALL Y2)X2=Y2=>(S3(X1,Y1,L)=>L<=1)
        21             (ALL X)(ALL Y)(X#Y=>R1(X,Y)).AND.(EXT X2)
                       (EXT Y2)NOT.X2=Y2=>(S3(X1,Y1,L)=>L<=1)
         22              (ALL X)(ALL Y)(X#Y=>S3(X,Y,1)).AND.(EXT
                          X2)(EXT Y2)NOT.X2=Y2=>(S3(X1,Y1,L)=>L<
                         =1)
          23               (ALL X)(ALL Y)(NOT.X=Y=>S3(X,Y,1)).AN
                           D.(EXT X2)(EXT Y2)NOT.X2=Y2=>(S3(X1,Y
                           1,L)=>L<=1)
           24                (ALL X)(ALL Y)(NOT.X=Y=>S3(X,Y,1)).
                             AND.(EXT X2)(EXT Y2)NOT.X2=Y2=>(S3(
                             X1,Y1,L)=>L<1.OR.L=1)
            25                 (ALL X)(ALL Y)(NOT.X=Y=>S3(X,Y,1)
                               ).AND.(EXT X2)(EXT Y2)NOT.X2=Y2=>
                               (S3(X1,Y1,L)=>L=0.OR.L=1)
             26                  (ALL X)(ALL Y)(NOT.S3(X,Y,0)=>S
                                 3(X,Y,1)).AND.(EXT X2)(EXT Y2)N
                                 OT.X2=Y2=>(S3(X1,Y1,L)=>L=0.OR.
                                 L=1) Proved by resolution
```

Primer 13 . Na ulazu u dokazivač zadana je rečenica SIR12: "If the graph is connected then for all X X is not isolated or the graph is trivial" .

Prevodjenjem na oblik kvantifikatorske formule dobijamo:

$$P2 \Rightarrow (\forall X)\neg Q3(X) \vee P4$$

Dokaz je izveden bez intervencije korisnika i kompletiran primenom rezolucijskog dokazivača. Sledi prikaz dokaza T12.

```
TYPE T12 ..
1       P2=>(ALL X)NOT.Q3(X).OR.P4
  3       P2=>(ALL X)NOT.Q3(X).OR.(ALL X1)(ALL Y1)X1=Y1
    4       (ALL X2)(ALL Y)R2(X2,Y)=>(ALL X)NOT.Q3(X).OR.(ALL X1)(A
            LL Y1)X1=Y1
      5       (ALL X2)(ALL Y)(EXT K)S2(X2,Y,K)=>(ALL X)NOT.Q3(X).OR
              .(ALL X1)(ALL Y1)X1=Y1
        6       (ALL X2)(ALL Y)(EXT K)((K=0.AND.X2=Y).OR.(K>0.AND.(
                EXT Z1)(S2(X2,Z1,K-1).AND.R1(Z1,Y))))=>(ALL X)NOT.Q
                3(X).OR.(ALL X1)(ALL Y1)X1=Y1
          7       (ALL X2)(ALL Y)(EXT K)((K=0.AND.X2=Y).OR.(K>0.AND
                  .(EXT Z1)(S2(X2,Z1,K-1).AND.R1(Z1,Y))))=>(ALL X)N
                  OT.(ALL Y2)NOT.R1(X,Y2).OR.(ALL X1)(ALL Y1)X1=Y1
            8       (ALL X2)(ALL Y)(EXT K)((K=0.AND.X2=Y).OR.(K>0.A
                    ND.(EXT Z1)(S2(X2,Z1,K-1).AND.R1(Z1,Y))))=>(ALL
                     X)(EXT Y2)NOT.NOT.R1(X,Y2).OR.(ALL X1)(ALL Y1)
                    X1=Y1
              9       (ALL X2)(ALL Y)(EXT K)((K=0.AND.X2=Y).OR.(K>0
                      .AND.(EXT Z1)(S2(X2,Z1,K-1).AND.R1(Z1,Y))))=>
                      (ALL X)(EXT Y2)R1(X,Y2).OR.(ALL X1)(ALL Y1)X1
                      =Y1
```

Primer 14 . Zadata je rečenica SIRO2:
"If the graph is not connected then the graph is connected in complement" .

Ovo tvrdjenje predstavlja osrednje tešku teoremu koju dokazuju studenti iz oblasti teorije grafova i najkomplikovaniji primer dokaza izvedenog pomoću dokazivača u sistemu GRAPH.

Izvedeni dokaz je kao što se vidi prilično dugačak, pa je najpre reprodukovan sa ucrtanim granama u drvetu dokaza kako bi se lakše video u celini, ali je zbog dubine morao biti prikazan na dve strane uz znatnije umanjivanje. Nakon toga, prikazan je detaljno sa svim poddrvetima koja iz tehničkih razloga (kao što je ranije objašnjeno) ne mogu da se prikažu.

Izvodjenje dokaza je teklo uz znatniju intervenciju korisnika, što se s obzirom na komplikovanost primera moglo i pretpostaviti.

Navešćemo pomoćne rečenice koje je korisnik zadao da bi sistem izvršio potrebna grananja bilo analizom slučajeva ili ubacivanjem medjuciljа, a zatim sledi prikaz interaktivnog dokaza.

```
Primary variable is SP2   of type S.
R2(X,Y)

Primary variable is SP21  of type S.
(EXT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y))

Primary variable is SP22  of type S.
R2A1(X,Z).AND.R2A1(Z,Y)

Primary variable is SP26  of type S.
S2A1(Z,Y,1)

Primary variable is SP27  of type S.
NOT.S2(X,Y,0).AND.NOT.S2(X,Y,1)

Primary variable is SP28  of type S.
S2A1(X,Y,1)

Primary variable is SP29  of type S.
(NOT.R2(X,X1).AND.NOT.R2(X1,Y)).OR.(NOT.R2(X,Y1).A
ND.NOT.R2(Y1,Y))

Primary variable is SP23  of type S.
S2(X,Z,0).AND.S2(X,Z,1)

Primary variable is SP25  of type S.
NOT.S2(Z,Y,0).AND.NOT.S2(Z,Y,1)

Primary variable is SP24  of type S.
NOT.S2(X,Z,0).AND.NOT.S2(X,Z,1)

Primary variable is SP202 of type S.
S2A1(X,Z,1)
```

```
1   NOT.P2=>P2A1
3     NOT.P2=>(ALL X)(ALL Y)R2A1(X,Y)
4       NOT.P2=>(ALL Y)R2A1(X,Y)
5         NOT.P2=>R2A1(X,Y)
6           NOT.(ALL X1)(ALL Y1)R2(X1,Y1)=>R2A1(X,Y)
7             (EXT X1)NOT.(ALL Y1)R2(X1,Y1)=>R2A1(X,Y)
8               (EXT X1)(EXT Y1)NOT.R2(X1,Y1)=>R2A1(X,Y)
9                 (EXT Y1)NOT.R2(X1,Y1)=>R2A1(X,Y)
10                  NOT.R2(X1,Y1)=>R2A1(X,Y)
11                    R2(X,Y)=>(NOT.R2(X1,Y1)=>R2A1(X,Y))
14                      R2(X,Y).AND.NOT.R2(X1,Y1)=>R2A1(X,Y)
15                        R2(X,Y).AND.NOT.R2(X1,Y1)=>(EXT Z)(NO
                          T.R2(X,Z).AND.NOT.R2(Z,Y))
19                          R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT.R2(
                            X,X1).AND.NOT.R2(X1,Y)).OR.(NOT.R2(
                            X,Y1).AND.NOT.R2(Y1,Y))
23                            R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT.R
                              2(X,X1).AND.NOT.R2(X1,Y)).OR.NOT.
                              R2(X,Y1).AND.(NOT.R2(X,X1).AND.NO
                              T.R2(X1,Y)).OR.NOT.R2(Y1,Y)
24                              R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT
                                .R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                                NOT.R2(X,Y1)
26                                R2(X,Y).AND.NOT.R2(X1,Y1)=>NO
                                  T.R2(X,X1).OR.NOT.R2(X,Y1).AN
                                  D.NOT.R2(X1,Y).OR.NOT.R2(X,Y1
                                  )
27                                  R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                    NOT.R2(X,X1).OR.NOT.R2(X,Y1
                                    )
29                                    NOT.(NOT.R2(X,X1).OR.NOT.
                                      R2(X,Y1))=>NOT.(R2(X,Y).A
                                      ND.NOT.R2(X1,Y1))
30                                      NOT.NOT.R2(X,X1).AND.NO
                                        T.NOT.R2(X,Y1)=>NOT.(R2
                                        (X,Y).AND.NOT.R2(X1,Y1)
                                        )
31                                        R2(X,X1).AND.NOT.NOT.
                                          R2(X,Y1)=>NOT.(R2(X,Y
                                          ).AND.NOT.R2(X1,Y1))
                                    ...     R2(X,X1).AND.NOT.NO
                                            T.R2(X,Y1)=>NOT.(R2
                                            (X,Y).AND.NOT.R2(X1
                                            ,Y1))
28   *                              R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                    NOT.R2(X1,Y).OR.NOT.R2(X,Y1
                                    ) True by symmetry and tran
                                    sitivity of R2
25                              R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT
                                .R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                                NOT.R2(Y1,Y)
35                                R2(X,Y).AND.NOT.R2(X1,Y1)=>NO
                                  T.R2(X,X1).OR.NOT.R2(Y1,Y).AN
                                  D.NOT.R2(X1,Y).OR.NOT.R2(Y1,Y
                                  )
36   *                              R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                    NOT.R2(X,X1).OR.NOT.R2(Y1,Y
                                    ) True by symmetry and tran
                                    sitivity of R2
37   *                              R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                    NOT.R2(X1,Y).OR.NOT.R2(Y1,Y
                                    ) True by symmetry and tran
                                    sitivity of R2
20                          (NOT.R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                            (NOT.R2(X,Y1).AND.NOT.R2(Y1,Y))=>(E
                            XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y))
21   *                        NOT.R2(X,X1).AND.NOT.R2(X1,Y)=>(E
                              XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y)
                              ) Proved by resolution
22   *                        NOT.R2(X,Y1).AND.NOT.R2(Y1,Y)=>(E
                              XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y)
                              ) Proved by resolution
16                        (EXT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y))=
                          >R2A1(X,Y)
17                          (NOT.R2(X,Z).AND.NOT.R2(Z,Y))=>R2A1
                            (X,Y)
18                            NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2A1
                              (X,Z).AND.R2A1(Z,Y)
39                              NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2
                                A1(X,Z)
41                                NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>
                                  (EXT K)S2A1(X,Z,K)
43                                  NOT.(EXT K1)S2(X,Z,K1).AND.
                                    NOT.R2(Z,Y)=>(EXT K)S2A1(X,
                                    Z,K)
44                                    (ALL K1)NOT.S2(X,Z,K1).AN
                                      D.NOT.R2(Z,Y)=>(EXT K)S2A

42          30  40
```

```
          44      (ALL K1)NOT.S2(X,Z,K1).AN
                  D.NOT.R2(Z,Y)=>(EXT K)S2A
                  1(X,Z,K)
            45      (ALL K1)NOT.S2(X,Z,K1).
                    AND.NOT.(EXT K2)S2(Z,Y,
                    K2)=>(EXT K)S2A1(X,Z,K)
              46      (ALL K1)NOT.S2(X,Z,K1
                      ).AND.(ALL K2)NOT.S2(
                      Z,Y,K2)=>(EXT K)S2A1(
                      X,Z,K)
                ...     (ALL K1)NOT.S2(X,Z,
                        K1).AND.(ALL K2)NOT
                        .S2(Z,Y,K2)=>(EXT K
                        )S2A1(X,Z,K)
    40    NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2
          A1(Z,Y)
      42    NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>
            (EXT K)S2A1(Z,Y,K)
        58    NOT.R2(X,Z).AND.NOT.(EXT K1
              )S2(Z,Y,K1)=>(EXT K)S2A1(Z,
              Y,K)
          59    NOT.R2(X,Z).AND.(ALL K1)N
                OT.S2(Z,Y,K1)=>(EXT K)S2A
                1(Z,Y,K)
            60    NOT.(EXT K2)S2(X,Z,K2).
                  AND.(ALL K1)NOT.S2(Z,Y,
                  K1)=>(EXT K)S2A1(Z,Y,K)
              61    (ALL K2)NOT.S2(X,Z,K2
                    ).AND.(ALL K1)NOT.S2(
                    Z,Y,K1)=>(EXT K)S2A1(
                    Z,Y,K)
                ...   (ALL K2)NOT.S2(X,Z,
                      K2).AND.(ALL K1)NOT
                      .S2(Z,Y,K1)=>(EXT K
                      )S2A1(Z,Y,K)
  30  *  R2A1(X,Z).AND.R2A1(Z,Y)=>R2A1(X,Y
         ) True by transitivity of R2
12  NOT.R2(X,Y)=>(NOT.R2(X1,Y1)=>R2A1(X,Y))
  13  NOT.R2(X,Y).AND.NOT.R2(X1,Y1)=>R2A1(X,Y
      )
    51  NOT.(EXT K)S2(X,Y,K).AND.NOT.R2(X1,Y1
        )=>R2A1(X,Y)
      52  (ALL K)NOT.S2(X,Y,K).AND.NOT.R2(X1,
          Y1)=>R2A1(X,Y)
        53  (ALL K)NOT.S2(X,Y,K).AND.NOT.R2(X
            1,Y1)=>(EXT K1)S2A1(X,Y,K1)
          54  (ALL K)NOT.S2(X,Y,K).AND.NOT.R2
              (X1,Y1)=>NOT.S2(X,Y,0).AND.NOT.
              S2(X,Y,1)
            56  *  (ALL K)NOT.S2(X,Y,K).AND.NOT.
                   R2(X1,Y1)=>NOT.S2(X,Y,0) Obvi
                   ously true
            57  *  (ALL K)NOT.S2(X,Y,K).AND.NOT.
                   R2(X1,Y1)=>NOT.S2(X,Y,1) Obvi
                   ously true
          55  NOT.S2(X,Y,0).AND.NOT.S2(X,Y,1)
              =>(EXT K1)S2A1(X,Y,K1)
            84  NOT.X=Y.AND.NOT.S2(X,Y,1)=>(E
                XT K1)S2A1(X,Y,K1)
              85  NOT.X=Y.AND.NOT.S2(X,Y,1)=>
                  S2A1(X,Y,1)
                87  NOT.X=Y.AND.NOT.R1(X,Y)=>
                    S2A1(X,Y,1)
                  88  NOT.X=Y.AND.NOT.R1(X,Y)
                      =>R1A1(X,Y)
                    89  X#Y.AND.NOT.R1(X,Y)=>
                        R1A1(X,Y)
                      ...  X#Y.AND.NOT.R1(X,Y)
                           =>R1A1(X,Y)
            86  *  S2A1(X,Y,1)=>(EXT K1)S2A1(X
                   ,Y,K1) Obviously true
```

```
'YPE T02 .
          NOT.P2=>P2A1
 3          NOT.P2=>(ALL X)(ALL Y)R2A1(X,Y)
    4         NOT.P2=>(ALL Y)R2A1(X,Y)
      5         NOT.P2=>R2A1(X,Y)
        6         NOT.(ALL X1)(ALL Y1)R2(X1,Y1)=>R2A1(X,Y)
          7         (EXT X1)NOT.(ALL Y1)R2(X1,Y1)=>R2A1(X,Y)
            8         (EXT X1)(EXT Y1)NOT.R2(X1,Y1)=>R2A1(X,Y)
              9         (EXT Y1)NOT.R2(X1,Y1)=>R2A1(X,Y)
                10        NOT.R2(X1,Y1)=>R2A1(X,Y)
                  11        R2(X,Y)=>(NOT.R2(X1,Y1)=>R2A1(X,Y))
                    14        R2(X,Y).AND.NOT.R2(X1,Y1)=>R2A1(X,Y)
                      15        R2(X,Y).AND.NOT.R2(X1,Y1)=>(EXT Z)(NO
                                T.R2(X,Z).AND.NOT.R2(Z,Y))
                        19        R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT.R2(
                                  X,X1).AND.NOT.R2(X1,Y)).OR.(NOT.R2(
                                  X,Y1).AND.NOT.R2(Y1,Y))
                          23        R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT.R
                                    2(X,X1).AND.NOT.R2(X1,Y)).OR.NOT.
                                    R2(X,Y1).AND.(NOT.R2(X,X1).AND.NO
                                    T.R2(X1,Y)).OR.NOT.R2(Y1,Y)
                            24        R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT
                                      .R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                                      NOT.R2(X,Y1)
                              26        R2(X,Y).AND.NOT.R2(X1,Y1)=>NO
                                        T.R2(X,X1).OR.NOT.R2(X,Y1).AN
                                        D.NOT.R2(X1,Y).OR.NOT.R2(X,Y1
                                        )
                                27        R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                          NOT.R2(X,X1).OR.NOT.R2(X,Y1
                                          )
                                  29        NOT.(NOT.R2(X,X1).OR.NOT.
                                            R2(X,Y1))=>NOT.(R2(X,Y).A
                                            ND.NOT.R2(X1,Y1))
                                    30        NOT.NOT.R2(X,X1).AND.NO
                                              T.NOT.R2(X,Y1)=>NOT.(R2
                                              (X,Y).AND.NOT.R2(X1,Y1)
                                              )
                                      31        R2(X,X1).AND.NOT.NOT.
                                                R2(X,Y1)=>NOT.(R2(X,Y
                                                ).AND.NOT.R2(X1,Y1))
                                        ...       R2(X,X1).AND.NOT.NO
                                                  T.R2(X,Y1)=>NOT.(R2
                                                  (X,Y).AND.NOT.R2(X1
                                                  ,Y1))
                                28  *     R2(X,Y).AND.NOT.R2(X1,Y1)=>
                                          NOT.R2(X1,Y).OR.NOT.R2(X,Y1
                                          ) True by symmetry and tran
                                          sitivity of R2
                            25        R2(X,Y).AND.NOT.R2(X1,Y1)=>(NOT
                                      .R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                                      NOT.R2(Y1,Y)
                              35        R2(X,Y).AND.NOT.R2(X1,Y1)=>NO
                                        T.R2(X,X1).OR.NOT.R2(Y1,Y).AN
                                        D.NOT.R2(X1,Y).OR.NOT.R2(Y1,Y
                                        )
                                36  *     R2(X,Y).AND.NOT.R2(X1,Y1)=>
```

```
                        NOT.R2(X,X1).OR.NOT.R2(Y1,Y
                        ) True by symmetry and tran
                        sitivity of R2
                37  *   R2(X,Y).AND.NOT.R2(X1,Y1)=>
                        NOT.R2(X1,Y).OR.NOT.R2(Y1,Y
                        ) True by symmetry and tran
                        sitivity of R2
       20       (NOT.R2(X,X1).AND.NOT.R2(X1,Y)).OR.
                (NOT.R2(X,Y1).AND.NOT.R2(Y1,Y))=>(E
                XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y))
         21  *   NOT.R2(X,X1).AND.NOT.R2(X1,Y)=>(E
                 XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y)
                 ) Proved by resolution
         22  *   NOT.R2(X,Y1).AND.NOT.R2(Y1,Y)=>(E
                 XT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y)
                 ) Proved by resolution
     16      (EXT Z)(NOT.R2(X,Z).AND.NOT.R2(Z,Y))=
             >R2A1(X,Y)
       17      (NOT.R2(X,Z).AND.NOT.R2(Z,Y))=>R2A1
               (X,Y)
         18      NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2A1
                 (X,Z).AND.R2A1(Z,Y)
           39      NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2
                   A1(X,Z)
             41      NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>
                     (EXT K)S2A1(X,Z,K)
               43      NOT.(EXT K1)S2(X,Z,K1).AND.
                       NOT.R2(Z,Y)=>(EXT K)S2A1(X,
                       Z,K)
                 44      (ALL K1)NOT.S2(X,Z,K1).AN
                         D.NOT.R2(Z,Y)=>(EXT K)S2A
                         1(X,Z,K)
                   45      (ALL K1)NOT.S2(X,Z,K1).
                           AND.NOT.(EXT K2)S2(Z,Y,
                           K2)=>(EXT K)S2A1(X,Z,K)
                     46      (ALL K1)NOT.S2(X,Z,K1
                             ).AND.(ALL K2)NOT.S2(
                             Z,Y,K2)=>(EXT K)S2A1(
                             X,Z,K)
                       ...     (ALL K1)NOT.S2(X,Z,
                               K1).AND.(ALL K2)NOT
                               .S2(Z,Y,K2)=>(EXT K
                               )S2A1(X,Z,K)
           40      NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>R2
                   A1(Z,Y)
             42      NOT.R2(X,Z).AND.NOT.R2(Z,Y)=>
                     (EXT K)S2A1(Z,Y,K)
               58      NOT.R2(X,Z).AND.NOT.(EXT K1
                       )S2(Z,Y,K1)=>(EXT K)S2A1(Z,
                       Y,K)
                 59      NOT.R2(X,Z).AND.(ALL K1)N
                         OT.S2(Z,Y,K1)=>(EXT K)S2A
                         1(Z,Y,K)
                   60      NOT.(EXT K2)S2(X,Z,K2).
                           AND.(ALL K1)NOT.S2(Z,Y,
                           K1)=>(EXT K)S2A1(Z,Y,K)
                     61      (ALL K2)NOT.S2(X,Z,K2
                             ).AND.(ALL K1)NOT.S2(
                             Z,Y,K1)=>(EXT K)S2A1(
                             Z,Y,K)
                       ...     (ALL K2)NOT.S2(X,Z,
```

```
                                              K2).AND.(ALL K1)NOT
                                              .S2(Z,Y,K1)=>(EXT K
                                              )S2A1(Z,Y,K)
                      38  *  R2A1(X,Z).AND.R2A1(Z,Y)=>R2A1(X,Y
                             ) True by transitivity of R2
              12      NOT.R2(X,Y)=>(NOT.R2(X1,Y1)=>R2A1(X,Y))
                13      NOT.R2(X,Y).AND.NOT.R2(X1,Y1)=>R2A1(X,Y
                        )
                  51      NOT.(EXT K)S2(X,Y,K).AND.NOT.R2(X1,Y1
                          )=>R2A1(X,Y)
                    52      (ALL K)NOT.S2(X,Y,K).AND.NOT.R2(X1,
                            Y1)=>R2A1(X,Y)
                      53      (ALL K)NOT.S2(X,Y,K).AND.NOT.R2(X
                              1,Y1)=>(EXT K1)S2A1(X,Y,K1)
                        54      (ALL K)NOT.S2(X,Y,K).AND.NOT.R2
                                (X1,Y1)=>NOT.S2(X,Y,0).AND.NOT.
                                S2(X,Y,1)
                          56  *   (ALL K)NOT.S2(X,Y,K).AND.NOT.
                                  R2(X1,Y1)=>NOT.S2(X,Y,0) Obvi
                                  ously true
                          57  *   (ALL K)NOT.S2(X,Y,K).AND.NOT.
                                  R2(X1,Y1)=>NOT.S2(X,Y,1) Obvi
                                  ously true
                        55      NOT.S2(X,Y,0).AND.NOT.S2(X,Y,1)
                                =>(EXT K1)S2A1(X,Y,K1)
                          84      NOT.X=Y.AND.NOT.S2(X,Y,1)=>(E
                                  XT K1)S2A1(X,Y,K1)
                            85      NOT.X=Y.AND.NOT.S2(X,Y,1)=>
                                    S2A1(X,Y,1)
                              87      NOT.X=Y.AND.NOT.R1(X,Y)=>
                                      S2A1(X,Y,1)
                                88      NOT.X=Y.AND.NOT.R1(X,Y)
                                        =>R1A1(X,Y)
                                  89      X#Y.AND.NOT.R1(X,Y)=>
                                          R1A1(X,Y)
                                    ...     X#Y.AND.NOT.R1(X,Y)
                                            =>R1A1(X,Y)
                            86  *   S2A1(X,Y,1)=>(EXT K1)S2A1(X
                                    ,Y,K1) Obviously true
```

Type Your next command, please!

MOVE ROOT T02 TO POINT 30 .

Type Your next command, please!

TYPE T02 .

```
30    NOT.NOT.R2(X,X1).AND.NOT.NOT.R2(X,Y1)=>NOT.(R2(X,Y).AND.NOT
      .R2(X1,Y1))
  31    R2(X,X1).AND.NOT.NOT.R2(X,Y1)=>NOT.(R2(X,Y).AND.NOT.R2(X1
        ,Y1))
    32    R2(X,X1).AND.R2(X,Y1)=>NOT.(R2(X,Y).AND.NOT.R2(X1,Y1))
      33    R2(X,X1).AND.R2(X,Y1)=>NOT.R2(X,Y).OR.NOT.NOT.R2(X1,Y
            1)
        34  *  R2(X,X1).AND.R2(X,Y1)=>NOT.R2(X,Y).OR.R2(X1,Y1) Tru
               e by symmetry and transitivity of R2
```

Type Your next command, please!

MOVE ROOT T02 TO POINT 45 .

```
Type Your next command, please!

TYPE T02 .
5       (ALL K1)NOT.S2(X,Z,K1).AND.NOT.(EXT K2)S2(Z,Y,K2)=>(EXT K)S
        2A1(X,Z,K)
  46     (ALL K1)NOT.S2(X,Z,K1).AND.(ALL K2)NOT.S2(Z,Y,K2)=>(EXT K
         )S2A1(X,Z,K)
    47     (ALL K1)NOT.S2(X,Z,K1).AND.(ALL K2)NOT.S2(Z,Y,K2)=>NOT.
           S2(X,Z,0).AND.NOT.S2(X,Z,1)
      49  *  (ALL K1)NOT.S2(X,Z,K1).AND.(ALL K2)NOT.S2(Z,Y,K2)=>NO
             T.S2(X,Z,0) Obviously true
      50  *  (ALL K1)NOT.S2(X,Z,K1).AND.(ALL K2)NOT.S2(Z,Y,K2)=>NO
             T.S2(X,Z,1) Obviously true
    48     NOT.S2(X,Z,0).AND.NOT.S2(X,Z,1)=>(EXT K)S2A1(X,Z,K)
      75     NOT.X=Z.AND.NOT.S2(X,Z,1)=>(EXT K)S2A1(X,Z,K)
        76     NOT.X=Z.AND.NOT.R1(X,Z)=>(EXT K)S2A1(X,Z,K)
          77     NOT.X=Z.AND.NOT.R1(X,Z)=>S2A1(X,Z,1)
            79     X#Z.AND.NOT.R1(X,Z)=>S2A1(X,Z,1)
              80     X#Z.AND.NOT.R1(X,Z)=>R1A1(X,Z)
                81     X#Z.AND.NOT.R1(X,Z)=>X#Z.AND.NOT.R1(X,Z)
                  82  *  X#Z.AND.NOT.R1(X,Z)=>X#Z Obviously true
                  83  *  X#Z.AND.NOT.R1(X,Z)=>NOT.R1(X,Z) Obviousl
                         y true
          78  *  S2A1(X,Z,1)=>(EXT K)S2A1(X,Z,K) Obviously true

Type Your next command, please!

MOVE ROOT T02 TO POINT 60 .

Type Your next command, please!

TYPE T02 .
60      NOT.(EXT K2)S2(X,Z,K2).AND.(ALL K1)NOT.S2(Z,Y,K1)=>(EXT K)S
        2A1(Z,Y;K)
  61     (ALL K2)NOT.S2(X,Z,K2).AND.(ALL K1)NOT.S2(Z,Y,K1)=>(EXT K
         )S2A1(Z,Y,K)
    62     (ALL K2)NOT.S2(X,Z,K2).AND.(ALL K1)NOT.S2(Z,Y,K1)=>NOT.
           S2(Z,Y,0).AND.NOT.S2(Z,Y,1)
      64  *  (ALL K2)NOT.S2(X,Z,K2).AND.(ALL K1)NOT.S2(Z,Y,K1)=>NO
             T.S2(Z,Y,0) Obviously true
      65  *  (ALL K2)NOT.S2(X,Z,K2).AND.(ALL K1)NOT.S2(Z,Y,K1)=>NO
             T.S2(Z,Y,1) Obviously true
    63     NOT.S2(Z,Y,0).AND.NOT.S2(Z,Y,1)=>(EXT K)S2A1(Z,Y,K)
      66     NOT.S2(Z,Y,0).AND.NOT.S2(Z,Y,1)=>S2A1(Z,Y,1)
        68     NOT.Z=Y.AND.NOT.S2(Z,Y,1)=>S2A1(Z,Y,1)
          69     NOT.Z=Y.AND.NOT.R1(Z,Y)=>S2A1(Z,Y,1)
            70     NOT.Z=Y.AND.NOT.R1(Z,Y)=>R1A1(Z,Y)
              71     NOT.Z=Y.AND.NOT.R1(Z,Y)=>Z#Y.AND.NOT.R1(Z,Y)
                72     NOT.Z=Y.AND.NOT.R1(Z,Y)=>Z#Y
                  74  *  NOT.Z=Y.AND.NOT.R1(Z,Y)=>NOT.Z=Y Obviousl
                         y true
                73  *  NOT.Z=Y.AND.NOT.R1(Z,Y)=>NOT.R1(Z,Y) Obviou
                       sly true
      67  *  S2A1(Z,Y,1)=>(EXT K)S2A1(Z,Y,K) Obviously true

Type Your next command, please!

MOVE ROOT T02 TO POINT 55 .

Type Your next command, please!
```

```
:YPE T02 .
;5      NOT.S2(X,Y,0).AND.NOT.S2(X,Y,1)=>(EXT K1)S2A1(X,Y,K1)
  84      NOT.X=Y.AND.NOT.S2(X,Y,1)=>(EXT K1)S2A1(X,Y,K1)
    85      NOT.X=Y.AND.NOT.S2(X,Y,1)=>S2A1(X,Y,1)
      87      NOT.X=Y.AND.NOT.R1(X,Y)=>S2A1(X,Y,1)
        88      NOT.X=Y.AND.NOT.R1(X,Y)=>R1A1(X,Y)
          89      X#Y.AND.NOT.R1(X,Y)=>R1A1(X,Y)
            90      X#Y.AND.NOT.R1(X,Y)=>X#Y.AND.NOT.R1(X,Y)
              91  *  X#Y.AND.NOT.R1(X,Y)=>X#Y Obviously true
              92  *  X#Y.AND.NOT.R1(X,Y)=>NOT.R1(X,Y) Obviously tr
                     ue
    86  *  S2A1(X,Y,1)=>(EXT K1)S2A1(X,Y,K1) Obviously true
```

Ovaj poslednji primer, i neki dalji eksperimenti sa dokazivačem, ukazali su na slučajeve gde heuristika za izbor netrivijalne transformacije podcilja nije bila uspešna. Naime, ona nije efikasna u slučajevima, kada su termi koji se pojavljuju kao argumenti predikata, značajni za zamenu podcilja i kada pre izvršenja netrivijalne transformacije podcilj treba primenom neke valjane formule transformisati u ekvivalentan oblik.

Budući da je u strategiji primene netrivijalne transformacije svesno odbačen nivo argumenata, bar u dosadašnjem stepenu razvoja dokazivača, u takvim slučajevima korisnik poništi deo drveta dokaza za koji smatra da nije poželjan. Zatim odgovarajućom komandom izvrši željenu transformacju podcilja, a zatim sistem ponovo nastavlja rad prema ugradjenoj strategiji ukoliko je režim rada automatski ili poluautomatski.

Napomenimo još neke smernice za dalje usavršavanje. Jedan od mogućih pravaca bio bi kreiranje dokaza kao AND/OR strukture drveta. U ILI granama bi se mogle uvesti drugačije koncipirane varijante inteligentne strategije vodjenja dokaza za tekući podcilj. Najjednostavniji pristup drugačijoj strategiji bio bi variranje redosleda i elemenata postojećeg niza trivijalnih transformacija, kao i eventualno drugačije koncipirane heuristike za izvodjenje netrivijalnih transformacija. Cilj bi bio dokazan kad bi bar jednom od ILI varijanti bio potvrdjen na sadašnji način.

Takodjer moguće je uvesti i nove korake u izvodjenju dokaza u cilju redukovanja i eventualno potpune eliminacije rezolucijskog dokazivača.

Pre svega blok za dokazivanje tačnosti podcilja opisan u 5.0.2 treba dopuniti slučajevima n-arnih kvantifikatora. Na primer, treba uključiti oblik

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)F(x_1,x_2,\ldots x_n) \Rightarrow F(t_1,t_2,\ldots t_n)$$

itd...

Pomenimo i sledeći tip pravila kojim bi se sadašnji sistem unapredio.

Tekućem podcilju oblika implikacije dodajmo kao konjunkt razne logičke posledice podformule leve strane.

Jedna od mogućnosti je sledeće pravilo:

Kad leva strana sadrži kao konjunkte dve formule sledećeg oblika:

$$(\forall X)(F(X) \Rightarrow G(X)) \quad \text{i} \quad F(t)$$

gde je t slobodna promenljiva ili osnovni term (term bez promenljivih), dodati kao konjunkt levoj strani i posledicu G(t).

Pravilo se može uopštiti i za slučaj n-arnih kvantifikatora.

Ako su konjunkti leve strane oblika:

$$(\forall X_1)(\forall X_2)\ldots(\forall X_n)(\ F(X_1,X_2,\ldots,X_n) \Rightarrow G(X_1,X_2,\ldots,X_n)) \ \text{i}$$

$$F(t_1,t_2,\ldots,t_n)$$

gde su t_i slobodne promenljive ili osnovni termi može se dodati kao konjunkt i posledica $G(t_1,t_2,\ldots,t_n)$.

S obzirom da bi nekontrolisano dodavanje raznih posledica kao konjunkta dovelo do rogobatnih podciljeva, a time unazadilo, a ne u unapredilo efektivnost dokazivača, vrlo je značajno primeniti ovo pravilo u odgovarajućem trenutku.

U postojećem dokazivaču trebalo bi ga primeniti nakon što smo kompleksnost podcilja sveli na nulu, dakle na podciljeve u razgranatom stablu dokaza i to nakon eliminacije kvantifikatora opisanih u 6.2.1, a pre onih opisanih u 6.2.2.

Predjimo sad na kritički osvrt i opštu ocenu dokazivača.

Heuristike ugradjene u dokazivač kao i inteligentna strategija vodjenja dokaza, razradjene su i ugradjene postupno kroz eksperimente sa dokazivačem, na primerima iz aritmetičke teorije grafova. Naravno, moguća su, a i predvidjena dalja usavršavanja sistema.

Dokazivač je radjen po ugledu na slične dokazivače razvijene u svetu i nadovezuje se na radove Newella, Gelerntera, a posebno Bledsoe-a, s tim što se u implementaciji počelo od samog početka, za razliku od sistema koji predstavljaju poboljšanja nekih delova postojećih.

Preuzeta je globalna strategija rada unazad od tvrdjenja koje dokazujemo, ka aksiomama, ali tako da se dokaz odvija na visokom nivou, uz korišćenje lema i bez spuštanja na nivo aksioma, kad god može.

Takodjer, preuzeto je i razbijanje na podciljeve, čime se dokaz datog podcilja svodi na dokazivanje podciljeva koji su po pravilu jednostavniji za dokaz. Tu su i razna pojednostavljenja i delimična kanonizacija formule, s tim što je to prilagodjeno konkretnoj teoriji u kojoj izvodimo dokaz.

Poredjenje sa sličnim dokazivačima razvijenim u svetu teško je izvesti budući da su im domeni različiti.

Takodjer, iz literature opisanih dokazivača nije moguce do kraja videti i sagledati celokupnu koncepciju rada i unutrašnju organizaciju.

Najbolja poredjenja se mogu izvršiti za dokazivače koji rade u istom domenu. Medjutim, i u takvim slučajevima (dokazivači u oblasti teorije skupova Bledsoe-a, Pastre-a i Browna) najčešce se konstatuje da su dokazivači neuporedivi što se tiče efikasnosti jer je zbog različito sprovedene unifikacije kao i opšte organizacije dokazivanja jedan bolje dokazao jedne primere, a drugi druge.

Postojeci dokazivač u sistremu GRAPH mogao bi da se primeni i na druge teorije, s tim što u tom slučaju treba poštovati sintaksu jezika GTCL, kao i uvedene konvencije. Trebalo bi, takodjer, u tom slučaju izbaciti domenski orjentisane heuristike. Čak i heuristika za izbor releventne definicije odnosno leme, koja je ključna u postojecem dokazivaču, i koja po svojoj prirodi nije direktno vezana za teoriju grafova, u nekoj teoriji koja ne bi imala bogatu strukturu definicija, kao što je to slučaj sa AGT, ne bi doprinela efektivnosti dokazivača.

Poput sličnih dokazivača razvijenih u svetu ni ovaj nije kompletan. Dakle, postoje teoreme koje ugradjenim koracima dokazivač neće uspeti da dokaže čak ni pod pretpostavkom neograničenih resursa.

Valjanost je, naravno, ispunjena. Naime, svaki kompletirani dokaz izveden pomoću dokazivača obezbedjuje da je polazno tvrdjenje teorema u AGT.

Kao i kod sličnih sistema VI i sva naredna poboljšanja ovog dokazivača imaće ograničenja kognitivnih mogućnosti tvoraca, kao i njihove strpljivosti i pedantnosti u tehničkim detaljima realizacije.

L I T E R A T U R A

<1> Allen J., Luckham D. An interactive theorem proving program. Machine Intelligence Vol 5(1970), Meltzer & Michie (eds), American Elsevier, New York, 321-336.

<2> Ballantine M. Some notes on computer generation of count-erexamples in topology. Univ. of Texas, Math. Dept. Memo ATP 24, Feb. 1977.

<3> Bibel W., Schreiber J. Proof search in a Gentzen-like system of first order logic. Proc. of the International Computing Symposium (1975), North Holland Publ. 205-212.

<4> Bibel W., Kovalski R. Fifth conference on Automated Deduction. Les Arcs, France, 1980. Lecture Notes in Computer Science 87. Springer-Verlag Berlin Heidelberg New York 1980.

<5> Bledsoe W. W. Splitting and reduction heuristics in automatic theorem proving. Artificial Intelligence 2 (1971), 55-77.

<6> Bledsoe W. W., Boyer R. S., Henneman W.H. Computer proofs of limit theorems. Artificial Intelligence 3 (1972), 27-60.

<7> Bledsoe W.W., Bruell P. A man-machine theorem proving. Artificial Intelligence 5 (1974), 51-72.

<8> Bledsoe W.W. Non-resolution theorem proving. Artificial Intelligence 9 (1977), 1-35.

<9> Bledsoe W. W., Tyson Mabry. The UT Interactive Prover. Uni-versity of Texas, Math. Department Memo ATP 17A, June 1978.

<10> Bledsoe W.W., Hines L.M. Variable elimination and chaining in a resolution-based prover for inequali-ties. University of Texas, Math. Dept. Memo ATP 56A, April 1980.

<11> Bledsoe W.W., Henchen L.J. What is automated theorem proving. Journal of Automated Reasoning 1 (1985) 23-28.

<12> Bobrow D.G., Hayes P.J. (Eds) Artificial intelligen-ce - where are we. Artificial Intelligence 25 (1985) 375-416.

<13> Brown F.M. Doing arithmetic without diagrams. Arti-ficial Intelligence 8 (1977), 175-200.

<14> Brown F.M. Towards the automation of set theory and its logic. Artificial Intelligence 10 (1978), 281-316.

<15> Brown F.M. An investigation into the goals of research in automatic theorem proving as related to mathematical reaso-ning. Artificial Intelligence 14 (1980), 221-242.

<16> Bundy A. Doing arithmetics with diagrams. Proc. third IJCAI 1973. 130-138.

<17> Bundy A. Analysing Mathematical Proofs. Proc. fourth IJCAI, 1975.

<18> Chang C. Lee R.C. Symbolic logic and mechanical theorem proving. Academic Press, New York - London, 1973.

<19> Church R.M., Church K.W. Plans, goals and search strategies for the selection of a move in chess. U Chess Skill in Man and Machine. (Edt) Frey P.W. Springer Verlag New York Hei-delberg Berlin, 131-156.

<20> Chinthayama Sets of independent axioms for ternary Boolean algebra. Notices of the American Mathematical Society, Vol. 16(1969), 654.

<21> Cvetkovic D. A project for using computers in further development of graph theory. The Theory and Applications of Graphs, Proc. of the 4-th Internat. Conf. on the Theory and Application of Graphs, Kalamazoo, 1980, ed. G. Chartrand et al., John Willey & Sons, New York 1982, 285-296.

<22> Cvetkovic D. Prevodjenje matematičkog teksta na jezik for-mula predikatskog računa pomoću računara. Proc. 4th International Yugoslav Symposium of Computer Technology and Problems of Informatics, Informatica 81, Ljubljana, 1981, 3 108, 1-3.

<23> Cvetkovic D., Pevac I. Generisanje rečenica ekvivalentnih zadatoj rečenici. Proc. 4th International Yugoslav Symposium of Computer Technology and Problems of Informatics, Informatica 81, Ljubljana 1981, 3 107, 1-3.

<24> Cvetkovic D., Kraus L., Simic S. Discussing graph theory with a computer I, Implementation of graph theoretic algorithms. Univ. Beograd, Publ. Elektrotehn. Fak., Ser. Mat. Fiz. No 716 - No 734 (1981), 100-104.

<25> Cvetkovic D. Discussing graph theory with a computer II, Theorems suggested by the computer. Publ. Inst. Math. (Beograd) 33(47) (1983), 29-34.

<26> Cvetkovic D., Pevac I. Discussing graph theory with a computer III, Man-machine theorem proving. Publ. Inst. Math. (Beograd) 34(48) (1983), 37-47.

<27> Cvetkovic D. Discussing graph theory with a computer IV, Knowledge organization and examples of theorem proving. Prosiding IV jugoslovenskog seminara iz teorije grafova, Novi Sad 1983. 43-68.

<28> Cvetkovic D., Jovanovic M., Kraus L. Discussing graph theory with a computer V, Graph theory bibliography. Predato za štampu u Zborniku ETAN-a.

<29> Cvetkovic D. Discussing Graph theory with a computer VI, Theorems proved with the aid of the computer. U rukopisu.

<30> Cvetkovic D., Pevac I. Algorithms for transforming first order formulas in their natural form. Publ. Elektrotehn. Fak. Univ. Beograd. Ser. Mat. Fiz. No 735 - No 762 (1982), 155-160.

<31> Cvetkovic D., Pevac I. Some heuristics in automatic theorem proving. Publ. Inst. Math. (Beograd) 35(49), 1984, 167-171.

<32> Cvetkovic D., Pevac I. Man-machine theorem proving in graph theory. Artificial Intelligence to appear.

<33> Cvetkovic D., Kraus L. GRAPH an expert system for the classification and extension of the knowledge in the field of graph theory, User's manual, University of

Belgrade, Faculty of Electrical Engineering, Belgrade 1983.

<34> Cvetkovic D. Further experiences in computer aided research in graph theory, Graphs, Hypergraphs and Applications, Proc. Conf. Graph Theory, Eyba October 1984, ed. Sachs, Teubner Verlagsgesellschaft, Leipzig 1985.

<35> Darlington J. Some theorem-proving strategies based on the resolution principle. Machine Intelligence, Vol. 2(1968), Dale & Michie (eds), American Elsevier, New York, 57-71.

<36> Davis M. The prehistory and early history of automated deduction. u Automation of Reasoning: Classical Papers on Computational Logic 1957-70 (eds. J.Siekmann & Wrightson), Vol. 1 (1983) Springer Verlag, New York, 1-28.

<37> Frey P.W. (Edt) Chess Skill in Man and Machine. Springer Verlag 1982.

<38> Gelernter H. A note on sintactic symmetry and the manipulation of formal systems by machine. Information and Control, No. 2(1959).

<39> Gelernter H. Realization of a geometry theorem proving machine. Proc. Int. Conf. on Information Processing (1959).

<40> Georgeff M. P. Strategies in heuristic search. Artificial Intelligence 20(1983), 393-425.

<41> Guard J., Oglesby F., Bennett J., Settle L. Semi-automated mathematics. J.ACM, Vol. 16(1969), 49-62.

<42> Henchen L., Wos L. Automated theorem proving: 1965-1970. u Automation of Reasoning: Classical Papers on Computational Logic 1957-70 (eds. J.Siekmann & Wrightson), Vol. 12 (1983) Springer-Verlag, New York, 1-24.

<43> Herbrand J. Investigations in proof theory. From Frege to Godel: A Source Book in Mathematical Logic od J. van Heijenoort, Harvard University Press, 1967, 525-581.

<44> Hotomski P.Z. Sposobi vstroenija pravila indukcii v proceduri avtomaticeskogo dokazateljstva teorem s rezolucijej. Publ. Inst. Math. (Beograd), 33(47) 1983, 89-95.

<45> Hotomski P.Z. Metode i pravila za [illegible] dokazivanje teorema u teorijama prvog reda [illegible]atickom indukcijom, Doktorska disertacija, Beograd, 19[illegible].

<46> Kovalski R. Search strategies for theorem proving. Machine Intelligence, Vol. 5(1970), Meltzer & Michie (eds), American Elsvier, New York, 181-200.

<47> Kovalski R. Logic for Problem Solving, Elsevier North Holland, New York 1979.

<48> Loveland D. A linear format for resolution. Proc. of the IRIA Symposium on Automatic Demonstration, Versailles, France, 1968, Springer-Verlag Publ., 147-162.

<49> Lankford D.S. Complete sets of reductions for computational logic. The University of Texas, Austin, Math. Dept. Memo ATP 21 Januar 1975.

<50> Lenat D. B. Automated theory formation in mathematics. Proc. Fifth IJCAI, Cambridge, Mass., August 1977.

<51> Lenat D. B. Heuretics: Theoretical and experimental study of heuristic rules. Heuristic Programing Project, Stanford University, 1982.

<52> Lenat D. B. The nature of heuristics. Artificial Intelli-gence 19 (1982) 189-249.

<53> Lenat D. B. Theory formation by heuristic search. The nature of heuristics II: Background and examples. Artificial Intelligence 21 (1983) 31-59.

<54> Lenat D. B. EURISCO: A program that learns new heuristics and domain concepts. The nature of heuristics III: program design and results. Artificial Intelligence 21 (1983) 61-98

<55> Loveland D W., Shostak R.E. Simplifying interpreted formu-las. Proc. Fifth Conf. on Automated Deduction, Les Arcs, 1980. 97-109.

<56> Loveland D. (edt) Sixth conference on Automated Deduction. New York, USA, June 1982, Springer Verlag Berlin-Heidelberg-New York 1982.

<57> Loveland D. Automated theorem proving: a quarter sentury review. u Automated Theorem Proving: After 25 Years American Mathematical Society Contemporary Mathematical Series (eds. Bledsoe and Loveland) 1983.

<58> Loveland D. Automated Theorem Proving: A Logical Basis. North Holland, Amsterdam, New York, Oxford 1978.

<59> Luckham D. Refinement theorems in resolution theory. AI Memo-81, AI Project, Stanford University, Stanford, California, 1969.

<60> Marzollo A. (edt) Topics in Artificial Intelligence. Springer - Verlag Wien, New York 1976.

<61> Meltzer B. Proof, abstraction and semantics in mathematics and artificial intelligence. U Topics in Artificial Intellgence. (edt) Marzollo Springer - Verlag Wien, New York 1976. 1-9.

<62> Nevins A. A human oriented logic for automatic theorem proving. J.ACM, Vol. 21(1974), 606-621.

<63> Newel A., Shaw J., Simon H. Empirical exploration with the logic theory machine. Computers and Thought, Feigenbaum and Feldman (eds), McGraw Hill Publ., New York, 1963, 109-133.

<64> Nilsson N.J. Principles of Artificial Intelligence. Tioga Press, Palo Alto, California. 1980.

<65> Pastre D. Automatic theorem proving in set theory. Artificial Intelligence 10 (1978), 1-27.

<66> Pearl Judea (edt) Search and Heuristics. North Holland Amsterdam, New York, Oxford 1983.

<67> Pevac Irena Heuristic for avoiding skolemization in theorem proving. Publ. Inst. Math. (Beograd) 38(52), 1985, 207-213.

<68> Plaisted D.A. Theorem proving by abstraction, Artificial Intelligence 16 (1981), 47-108.

<69> Polya G. Mathematics and Plausible Reasoning. Princeton University Press, Princeton, N.J. 1954.

<70> Prawitz D. An improved proof procedure. Theoria 26, 102-139.

<71> Pravitz D. Natural Deduction. A proof theoretical study. Almqvist & Wiksell Stockholm, Goteborg, Upsala 1965.

<72> Prešić S. Elementi matematičke logike. Matematička biblioteka, Zavod za izdavanje udžbenika SRS Beograd 1968.

<73> Prešić M., Prešić S. Uvod u matematičku logiku - Teorija i zadaci. Matematički institut, Beograd 1979.

<74> Robinson J.A. Logic: Form and Function. The Mechanization of Deductive Reasoning. Edinburg University Press. Edinburg 1979.

<75> Robinson G.A., Wos L. Paramodulation and theorem proving in first order theory with equality. Machine Intelligence, Vol. 4(1969), Meltzer & Michie (eds), American Elsevier, New York, 135-150.

<76> Robinson J.A., A machine oriented logic based on resolution principle. J.ACM, Vol. 12(1965), 23-41.

<77> Robinson J.A., A review of automatic theorem proving. Proc. of the Symposia in Applied Mathematics, Vol. 19(1967), American Mathematical Society, Providence, Rhode Island, 1-18.

<78> Scarrott G. (edt) The Fifth Generation Computer Project, State of the Art Report. Pergamon 1983.

<79> Shannon C. E. Programing a computer for playing chess. Philisophical Magazine, 41 (1950) 256-275.

<80> Siekman J., Wrightson G. (eds) Automaton of Reasoning 1 & 2 Springer Verlag Berlin - Heidelberg - New York 1983.

<81> Simon H. A. Search and reasoning in problem solving. Artificial Intelligence 21 (1983) 7-29.

<82> Stojkovic V., Cvetkovic D. Analiza pomocu racunara recenica dela engleskog jezika formalizovanog za upotrebu u teoriji grafova. XXV Jugosl. konferencija ETAN-a Mostar 1981, sv.III, 271-278.

<83> Wos L., Overbeek R., Lusk E., Boyle J. Automated Reasoning: Introduction and Applications. Prentice Hall, Englewood Cliffs (1984).

<84> Wos L., Carson D., Robinson G. The unit preference strategy in theorem proving. Proc. of the Fall Joint Computer Con-ference, 1964, Thompson Book Company, New York, 615-621.

<85> Wos L., Carson D., Robinson G. Efficiency and completeness of the set-of-support strategy in theorem proving. J.ACM, Vol. 12(1965) 536-541.

<86> Wos L., Robinson G., Carson D., Shalla L. The concept of demodulation in theorem proving. J.ACM, Vol. 14(1967), 698-704.

<87> Wos L. et all An overview of automated reasoning and rela-ted fields. Journal of automated reasoning 1 (1985) 5-48.

SPISAK PROGRAMA SA POZIVIMA POTPROGRAMA

U delu koji sledi navodi se abecedno lista potprograma koje sam ja implementirala u toku programske realizacije dokazivača u sistemu GRAPH. Pored imena potprograma, za svaki je posle znaka > ispisana i odgovarajuća lista potprograma koje taj poziva ne računajući višestruke pozive.

Nekoliko potprograma (GENERE, AXCHNG, DECIS, NEPOTZ) je idejno razradjeno u saradnji sa profesorom Cvetkovićem budući da je bilo neophodno uklopiti ih u postojeći deo sistema ili nadovezati na implementirane potprograme.

Programi FREAD, FSTORE za učitavanje i storiranje zapisa na fajlove smeštene na spoljnoj memoriji, nadalje FOPEN i FCLOSE za otvaranje i zatvaranje odredjene datoteke koja nije stalno otvorena dok taj deo sistema radi, kao i programi MESSAG i MESSI koji ispisuju poruke korisniku sistema, preuzeti su iz ostalih delova sistema, a autor je mr Laslo Kraus. U realizaciji je korišten i niz potprograma koji omogućavaju dinamičko korištenje memorijskog prostora kao što su AUXVC, MPOS, MDEL, koji se pozivaju iz većine dole navedenih potprograma, no budući da im je cilj samo ušteda prostora, a ne i realizacija odredjenog zadatka njihovi pozivi su izostavljeni sa spiska radi bolje preglednosti i lakšeg snalaženja.

U sklopu dokazivača nalazi se i deo potprograma, gde sam učestvovala u idejnoj razradi dok je implementaciju uradio profesor Dragoš Cvetković.

Ti programi ovde nisu navedeni, a njihova dokumentacija je sastavni deo programske dokumentacije sistema GRAPH.

ABECEDNI SPISAK PROGRAMA SA POZIVIMA POTPROGRAMA

ANAPR > ARGUM, ARGE, FREVAR, UNIJA, VELIC
ANARE >
APSVRH > BOUNDV, VRHI
ARGE >
ARGUM >
AXCHNG > LESHNG
BACTR > CURENT, STORSF, NEXTI
BACTZN > PREBAC, APSVRH, SIGNUS, NEXTI
BOUNDV >
BRKODI >
BRUNI >
CHANAD >
CHECKT > TEKUCI
CORESP > BOUNDV, VEVIC, FREVAR, KPOLJE, POLJE, ANAPR, ANARE, IDENT, VECTOR, UNIJA, ZAMKV, SUPKV, MASDKA, CHANAD, CORGR
CORGR > RAZPR, CHANG
CURENT > VRHI
DECIS >
DFCHNG > TKVAN, ZAMENA, FSTORE
ELIMKA > VFCHKO
ELSKUP >
EQCHNG > ISPEQU, CHANG, TRUREL
EQF > JEDN, ISFOR
EQUALF > OPERAC, OGOKV, FORGO, VELIC
EXTKV > BOUNDV
FORGO > KPOLJE, POLJE, CHANAD
FORPRE > ZATVAR, NEPOTZ, ELIMKA, PRONEG, SKOLEM, BRUNI, KONJUD, IZBACZ, SQUEE, PREFIX, ZVEZDA
FREVAR >
FSMENA > TEKUCI, CHANAD, PREBAC
FUNKC >
GENERE > AXCHNG, LECHNG, ZAMENA, VFCHNG,
ISFOR > BOUNDV, VECTOR, UNIJA, ZAMKV, SUPKV, NEPOTZ

ISKVA > APSVRH

ISPEQU > TEKFOR

ISPIT > PRENOS, ISFOR, ISKVA, VALG, VAL1

IZBACZ >

JEDN >

JESKUP > ELSKUP

KONJUD > VFCHKO

KPOLJE >

KVANTC > APSVRH

LECHNG > LESHNG

LEFTRI > APSVRH, BACTZN, ADDOP

LEKVA

LESHNG > KPOLJE, POLJE, OPERAC, PODFOR, SUBFOR, SKUP, RELPRE, FNDLMS, FREAD, FDISPL, ADQUES, STRANA, VRH, FREEVR, JESKUP, EQUALF, CORESP, LEVSM, LGRAF, CHANAD, NEPOTZ, REFLE, FSTORE

LESHNT > KPOLJE, POLJE, OPERAC, PODFOR, SUBFOR, SKUP, RELPRE, FNDLMS, STRANA, VRH, FREEVR, JESKUP, VRHI, EQUALF, CORESP, LEVSM, LGRAF, CHANAD, NEPOTZ, REFLE

LEVSM > BOUNDV, VELIC, FREVAR, VECTOR, UNIJA, ZAMKV, SUPKV, MASKA, CHANAD

LEVEL >

LGRAF > RAZPR, CHANG, PREBAC

MOGPE >

MASKA >

MOGPE >

MULTIZ > PAIR, CHANAD

NEGAT > TESTFO

NEPOTZ > MULTIZ, PAIR, PREBAC, APSVRH, SQUEE

NEXTI >

NONKVA >

ODEF >

ODRZN > PREBAC, APSVRH, NEXTI

OGOKV > BOUNDV, CHANAD

OPERAC >

PAIR >
PNAME >
PODFOR >
POMSUB > KPOLJE, POLJE, OPERAC, PODFOR
POLJE > FREVAR, ANAPR, ANARE
PREBAC >
PREDSM > MASKA, CHANAD
PRENOS > NEPOTZ
PRONEG > VFCHKO, NEPOTZ, NONKVA, PREBAC
RAZPR >
REDUC >
REFBR > REFRAZ, UNIJA, SKUP, FSTORE
REFER > REFRAZ
REFLE > REFRAZ, UNIJA
REFRAZ > FREAD, DPOS, ODEF, SKUP
SFCHNG > KPOLJE, POLJE, OPERAC, VECTOR, PODFOR, SUBFOR,
FREAD, DDPOS, SKUP, FREAD, DPOS, ODEF, EQUALF,
CORESP, PREDSM, SHANAD, NEPOTZ, REFER, FSTORE
SKOLEM > EXTKV, FREEVR, FUNKC, CHANG, SHANAD
SKUP >
SLEZAM > VRH, VRHRE, BACTR,EQF, MOGPE, PREBAC,
FSMENA, NEPOTZ, REFBR, STORSF
SLEZKO > VRH, VRHRE, BACTR, EQF, CHECKT, FSMEKO, NEPOTZ,
STORSF
STORSF >
STRANA > VRH, SUBN
SUBFOR >
SUBN > BOUNDV, POMSUB
SUPKV > CHANAD
TEKFOR >
TEKUCI >
TERPRO >
TKVAN >
TRARBR > FREAD, DPOS, FNDDOP
TRYE > NEPOTZ, VRHAV, TEKFOR, ISPIT
UNIJA >

UZAG >
VAL1 > NEPOTZ, CHANG, ISFOR, APSVRH
VALG > CHANG, PREBAC, ISFOR
VECTOR >
VELIC >
VEZANE >
VFCHNG > FREAD, FDISPL, STRANA, SLEZAM
VFCHKO > FREAD, STRANA, SLEZKO
VISEP >
VRH > SUBN
VRHI > SUBN
VRHAV > APSVRH, KONJUD, BRKODI
VRHRE > SUBN
ZAG > POMSUB, CHANAD
ZAMENA > FREAD, DPOS, DDPOS, ODEF, CHANG, PREBAC, ANAPR,
FREEVR, UNIJA, BOUNDV, ZAMKV, UZAG, CHANAD,
NEPOTZ, TRARBR, VISEP
ZAMKV >
ZATVAR > FREEVR, TESTFO, PREBAC
ZVEZDA > CHANG

FUNKCINALNI OPIS PODPROGRAMA

ANAPR - Analizira se tekuci predikat u datoj kvantifikatorskoj formuli zadat pomocu lokacije početka i kraja. Rezultat je broj argumenata predikata, nizovi argumenata predikata kao i niz svih promenljivih u argumentima.

ANARE - Za datu kvantifikatorsku formulu i lokaciju početka relacionog predikata u njoj odredjuje početak i kraj domena dejstva tog relacionog predikata kao i levi i desni argument te relacije.

APSVRH - Za datu kvantifikatorsku formulu odredjuje se kod vrhovne logičke operacije u njoj ako je to "¬", "∧", "∨", "=>", "<=>", kao i lokacija vrhovne logičke operacije. U slučaju kad je vrhovni kvantifikator odredjuje se i vezana promenljiva uz taj kvantifikator.

ARGE - Za datu kvantifikatorsku formulu i lokaciju početka terma odredjuje lokaciju kraja tog terma i izdvaja term.

ARGUM - Za datu kvantifikatorsku formulu i lokaciju početka argumenta nalazi lokaciju kraja argumenta i izdvaja ga za relacione predikate.

AXCHNG - Za datu kvantifikatorsku formulu vrši zamenu podformule koriscenjem aksioma. Rezultat smešta u datoteku generisanih rezultujucih formula.

BACTR - Sintaksna analiza drveta tj. "backtracking" za ispitivanje podudarnosti drveta logičkih operacija. Za datu kvantifikatorsku formulu i lokaciju početka i kraja izdvojene podformule u njoj i datu lemu odredjuje da li je drvo logičkih opearcija leme poddrvo drveta logičkih operacija podformule. Ukoliko jeste vrši se prideljivanje odgovarajućih formula pojedinim slovima na lišću leme.

BACTZN - Program za "backracking" kojim se odredjuje znak predikata na lišću drveta kvantifikatorske formule. Na izlazu dobijamo prema redosledu pojavljivanja predikata vektor sufiksa predikatskih naziva predikata kao i referentne brojeve označene odgovarajućim znakom.

BOUNDV – Za datu kvantifikatorsku formulu daje sva pojavljivanja vezanih promenljivih sa lokacijama i domene dejstva pojedinih kvantifikatora.

BRKODI – Izračunava broj konjunkcija u pretpostavci i broj disjunkcija u zaključku formule.

BRUNI – Briše sve univerzalne kvantifikatore.

CHANAD – Za datu kvantifikatorsku formulu i lokaciju u njoj izbacuje počev od date lokacije k simbola a zatim na to mesto ubacuje dati niz.

CHECKT – Ujdenačava logičke konstante "tačno" i "netačno" u transformacijama pojednostavljenja zadate formule pomoću valjanih formula.

CORESP – Za datu kvantifikatorsku formulu i datu lemu ili desnu stranu definicije utvrdjuje mogućnost zamene podformule pomoću nje. U slučaju kad je zamena moguća odredjuje substituciju za slobodne promenljive i grafovske nastavke.

CORGR – Ispituje mogućnost usaglašavanja grafovskih promenljivih pri zameni podformule pomoću leme.

CURENT – Ispituje jednakost vrhovnih logičkih operacija u dve formule.

DFCHNG – Zamenjuje po jedan predikat u formuli odgovarajućom definicijom.

DECIS – Odlučuje o izboru bloka u automatskom radu u procesu generisanja ekvivalentnih rečenica.

ELIMKA – Eliminiše ekvivalenciju i implikaciju na osnovu odgovarajuće valjane formule u pripremi za rezoluciju.

ELSKUP – Ispituje da li je k element skupa predstavljenog datim vektorom.

EQCHNG – Vrši jednakostnu substituciju u trivijalnim transformacijama.

EQF – Ispituje da li jednakim slovima odgovaraju jednake pridružene formule do na preimenovanje vezanih promenljivih izbacivanje nepotrebnih zagrada.

EQUALF - Ispituje podudarnost ogoljenih oblika dve formule.

EXTKV - Daje domen dejstva prvog egzistencijalnog kvantifikatora i kod vezane promenljive uz njega.

FORGO - Pravi ogoljeni oblik kvantifikatorske formule tako što izbacuje argumente predikata i promenljive uz kvantifikatore.

FORPRE - Transformiše kvantifikatorsku formulu iz standardnog oblika na oblik sastavaka primenom skolemizacije u cilju pripreme formule za primenu pravila rezolucije.

FREVAR - Za datu kvantifikatorsku formulu nalazi slobodne promenljive, sva njihova pojavljivanja i lokacije u formuli kao i lokacije predikata tipa P, Q, R, S.

FSMEKO - Po utvrdjivanju da je zamena sa datom valjanom formulom oblika ekvivalencije, na primer s leva u desno, dozvoljena, desna strana se zamenjuje tako što se slova zamene nizom prethodno utvrdjenih korespodentnih formula.

FSMENA - Po utvrdjivanju da je zamena sa tekućom valjanom formulom oblika ekvivalencije, na primer s leva u desno, dozvoljena, desna strana se zamenjuje tako što se slova zamene nizom prethodno utvrdjenih korespodentnih formula.

FUNKC - Uvodi Skolemove funkcije u pripremi formule za rezoluciju.

GENERE - Za datu kvantifikatorsku formulu program generiše datoteku rečenica koje su ekvivalentne zadatoj na osnovu odredjenog broja ekvivalentnih transformacija pomoću aksioma, definicija, lema i valjanih formula datih u sistemu.

IDENT - Utvrdjuje unifikaciju u zameni formule pomoću leme ili definicije.

ISFOR - Upredjuje jednakost dve formule do na preimenovanje vezanih promenljivih i apstrahovanje suvišnih zagrada.

ISKVA - Ako je kvantifikator vrhovna operacije tada odredjuje lokacije pojavljivanja vezane promenljive uz njega.

ISPEQU - Za datu kvantifikatorsku formulu i datu pod-formulu utvrdjuje da li je podformula konjunkt leve strane do na primenu substitucije slobodnih promenljivih.

ISPIT - Utvrdjuje tačnost podcilja korišćenjem odgovarajućih valjanih formula uporedjivanjem po jednog konjunkta pretpostavke i jednog disjunkta iz zaključka.

IZBACZ - Izbacuje sve zagrade iz formule.

JEDN - Za dati niz traži klase jednakih elemenata i pomoću indikatora izveštava o postojanju takvih.

JESKUP - Za dva skupa ispituje da li su u relaciji biti pravi podskup, u relaciji jednakosti, u relaciji biti pravi nadskup ili su neuporedivi.

KONJUD - Za datu kvantifikatorsku formulu primenjuje zamenu valjanom formulom sa zadatim rednim brojem sve dok je zamena moguća.

KPOLJE - Za datu kvantifikatorsku formulu daje kod i lokaciju svih predikata kako onih tipa P, Q, R, S, tako i relacionih <, >, = itd.

LECHNG - Vrši pripremu za program LESHNG.

LEFTRI - Odredjuje znake predikata leve i desne strane formule koje je oblika implikacije ili ekvivalencije.

LEKVA - Ispituje da li u formuli ima kvantifikatora i odredjuje im broj.

LESHNG - Za datu kvantifikatorsku formulu vrši trans-formisanje pomoću grafovskih lema.

LESHNT - Za datu kvantifikatorsku formulu vrši trans-formisanje pomoću grafovskih lema uz zabranu zamene u slučajevima inverznih trasa napada.

LEVEL - Za svaku zagradu u formuli odredjuje njen nivo.

LEVSM - Po utvrdjenoj substituciji primenjuje je na slobodne promenljive iz formule sa kojom zamenjujemo uz prethodno preimenovanjew vezanih u nove vezane promenljive ukoliko je potrebno.

LGRAF - Po završenoj substituciji slobodnih promenljivih usaglašava grafovske nastavke.

MASKA - Za datu promenljivu vadi odgovarajuću substituciju.

MOGPE - Za desnu stranu valjane formule ispituje da li svako slovo ima pojavljivanje u njenoj levoj strani.

MULTIZ - Iz date kvantifikatorske formule izbacuje višestruke zagrade.

NEGAT - Negira formulu u pripremi za rezoluciju.

NEPOTZ - Za datu kvantifikatorsku formulu briše nepotrebne zagrade.

NONKVA - Provlači negaciju kroz kvantifikator sve dok je moguće u pripremi za rezoluciju.

ODEF - Za dati referentni broj koji ukazuje na definiciju odredjuje predikat koji se njome definiše, vadi definiendum i njegove slobodne promenljive.

ODRZN - Odredjuje znak podformule.

OGOKV - Ogoljava formulu od kvantifikatora i zagrada.

OPERAC - Za datu kvantifikatorsku formulu nalazi logičke operacije i njihove lokacije u formuli.

PAIR - Traži odgovarajuće parove zagrada u datoj formuli izostavljajući pri tom one koje ogradjuju kvantifikatore, kao i one koje utvrdjuju prioritet aritmetičkih operacija na nivou terma.

PNAME - Za datu kvantifikatorsku formulu i lokaciju početka predikata u njoj daje predikatski naziv i njegovu dužinu.

PODFOR - Za datu kvantifikatorsku formulu odredjuje domene dejstva za $\neg$, $\wedge$, $\vee$, $=>$, $<=>$.

POMSUB - Za datu kvantifikatorsku formulu i utvrdjene domene dejstav kvantifikatora daje niz logičkih operacija, njihove lokacije i domene dejstva.

POLJE - Za datu kvantifikatorsku formulu daje domene dejstva svih predikata.

PREBAC - Izbacuje unarne logičke operacije ukoliko su one vrhovne, a u protivnom prebacuje stari vektor u novi.

PREDSM - U predikatu definienduma, pri zameni podformule koja se može ujednačiti sa definiensom pomoću definienduma vrši zamenu slobodnih promenljivih pomoću nadjene supstitucije.

PRENOS - Prebacuje niz koji predstavlja kvantifikatorsku formulu u novi niz tako što rezultujući čisti od nepotrebnih zagrada.

PRONEG - Provlači negaciju ka predikatima eliminacijom dvojne negacije i primenom De Morganovih pravila.

RAZPR - Odredjuje elemente iz sufiksa predikatskih slova prema zadatom kodu : skup grafovskih promenljivih, skup operacija nad grafovima.

REDUC - Za zadati par nizova pravi dva odgovarajuća skupa, tj. izbacuje elemente koji se ponavljaju.

REFBR - Po izvršenoj zameni kvantifikatorske formule nalazi referentne brojeve rezultujuće formule.

REFER - Po izvršenoj zameni kvantifikatorske formule pri zameni definiensa definiendumom nalazi referentne brojeve u rezultatu.

REFLE - Po izvršenoj zameni kvantifikatorske formule pomoću leme, nalazi novi niz referentnih brojeva.

REFRAZ - Razdvaja referentne brojeve za datu podformulu i za ostatak formule bez te podformule.

SFCHNG - Za datu kvantifikatorsku formulu zamenjuje podformulu u njoj koriscenjem definicije i to zdesna ulevo tj. definiens definiendumom.

SIGNUS - Pri zameni u dokazivaču po trasama napada svakom referentnom broju pridružuje znak odgovarajućeg predikata u formuli. Referentni brojevi se navode prema redosledu pojavljivanja u formuli. Takodjer, odredjuje i lokacije pojavljivanja tih predikata.

SKOLEM - Vrši skolemizaciju formule koja je prethodno očišćena od implikacija i ekvivalencija i prevedena u preneks oblik.

SKUP - Od datog niza pravi skup, tj. izbacuje jednake elemente.

SLEZAM - Program za zamenu podformule ekvivalentnom na osnovu tekuće valjane formule oblika ekvivalencije. Rezultat se smešta u rezultujuću datoteku.

SLEZKO - Program za zamenu podformule ekvivalentnom na osnovu zadate valjane formule oblika ekvivalencije.

STORSF - Vrši sukcesivno pridelјivanje vektora formula odgovarajućim slovima iz valjane formule.

STRANA - Za datu kvantifikatorsku formulu oblika implikacije ili ekvivalencije nalazi obe njene strane.

SUBFOR - Odredjuje sve podformule date kvantifikatorske formule.

SUBN - Za datu kvantifikatorsku formulu daje sve domene logičkih operacija kao i njihove kodove i lokacije.

SUPKV - Za datu kvantifikatorsku formulu i utvrdjene lokacije pojavljivanja vezanih promenljivih vrši preimenovanje vezanih korištenjem elemenata zadatog niza novih vezanih promenljivih.

TEKFOR - Izdvaja i-ti konjunkt u pretpostavci ili i-ti disjunkt u zaključku, tako što mu utvrdjuje lokaciju početka i kraja u odnosu na formulu.

TEKUCI - Vadi j-tu kolonu dvodimenzinalne matrice zapisane kao jednodimenzionalni vektor.

TERPRO - Zabranjuje unifikaciju pri substituciji u kojoj promenljivoj na pr. tipa X odgovara promenljiva ili term sa promenljivom tipa U, V, W ili tipa K, L, M, N.

TKVAN - Za datu kvantifikatorsku formulu daje vezane promenljive u njoj.

TRARBR - Transformiše skup referentnih brojeva za slučaj zamene predikata za grafovskim nastavkom predikatom sa nastavkom koji se definiše preko R1.

TRYE - Proba da li se neki konjunkt spreže sa nekim disjunktom.

UNIJA - Pravi uniju od elemenata dva vektora. Rezultat je niz uredjen po veličini.

UZAG - Ubacuje datu formulu u zagrade.

VAL1 - Za vrednost parametra 1 ispituje da li su dati konjunkt i dati disjunkt oblika $(\exists X)\ F(X)$, $F(t)$ gde je t term koji se svodi na promenljivu ili konstantu. Ako je vrednost parametra 2 ispituje da li je disjunkt oblika $F(t_1)$

$\wedge$ $F(t_2)$. Kad parametar ima vrednost 3 ispituje da li su konjunkt i disjunkt oblika $F(t)$ i $(\forall X)F(X)$ i najzad za vrednost 4 potprogram ispituje da li je konjunkt oblika $F(t_1) \vee F(t_2)$.

VALG – Ispituje podudarnost konjunkta i disjunkta.

VECTOR – Za datu kvantifikatorsku formulu daje sve vezane promenljive.

VELIC – Ulazni niz uredjuje po veličini.

VEZANE – Dodaje paru vektora na završetak zadate dve promenljive.

VFCHNG – Vrši transformaciju formule koristenjem datoteke valjanih formula oblika ekvivalencije.

VFCHKO – Vrši transformaciju zadate formule koristenjem odredjene valjane formule oblika ekvivalencije tako što zamenu obavlja samo jedamput. Ukoliko valjana formula sadrži konstante "tačno" i "netačno" program je automatski pojednostavljuje.

VISEP – Ispituje da li u zadatoj formuli sem tekućeg predikata ima još pojavljivanja predikata sa istim predikatskim slovom.

VRH – Za datu kvantifikatorsku formulu nalazi vrhovnu logičku operaciju, njen redni broj i lokaciju u formuli ukoliko postoji.

VRHI – Za datu kvantifikatorsku formulu nalazi vrhovnu operaciju, lokaciju i njen domen dejstva.

VRHAV – Po transformisanju formule sa n-arnom konjunkcijom u leftmost drvo, za vrhovnu n-arnu konujnkciju u pretpostavci odredjuje broj i lokacije binarnih znakova konjunkcije. Za drugu vrednost ulaznog parametra radi isto za disjunkcije u zaključku date formule.

VRHRE – Za operaciju koja je jednaka zadatom kodu traži u datoj formuli početak i kraj domena dejstva operacija sa istim kodom.

ZAG – Za svaku operaciju utvrdjuje da li je domen dejstva u zagradama i da li su one potrebne ili suvišne. Ako su suvišne, odbacuje ih.

ZAMENA - Zamenjuje i-ti predikat definicijom sleva udesno tj. definiendum definiensom po utvrdjivanju substitucije slobodnih promenljivih i preimenovanju vezanih promenljivih, ukoliko je potrebno.

ZAMKV - Za dati niz zabranjenih promenljivih i dati niz vezanih promenljivih pravi novi niz koji preimenuje tako da ne sadrži zabranjene promenljive.

ZATVAR - Zatvara formulu dopisivanjem univerzalnih kvantifikatora ispred nje.

ZVEZDA - Završetak pripreme formule za rezoluciju preimenuje znak ∧ u * i dopisuje * na kraj formule, čime se dobija niz sastavaka razdvojenih sa *.