Univerzitet u Beogradu
Matematički fakultet

Mladen S. Nikolić

Usmeravanje pretrage u automatskom dokazivanju teorema

Doktorska disertacija

Beograd, 2013.

University of Belgrade
Faculty of Mathematics

Mladen S. Nikolić

Guiding Search in Automated Theorem Proving

PhD disertation

Belgrade, 2013.

Mentor:
dr Predrag Janičić, vanredni profesor, Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:
dr Filip Marić, docent, Univerzitet u Beogradu, Matematički fakultet
dr Zoran Ognjanović, naučni savetnik, Matematički institut SANU

Datum odbrane: _____

Naslov disertacije: Usmeravanje pretrage u automatskom dokazivanju teorema

Rezime: U ovom radu se razmatra problem usmeravanja pretrage u automatskom dokazivanju teorema. Rad se sastoji od dva dela čija je dodirna tačka CDCL sistem pretrage, koji se intenzivno koristi kod modernih SAT rešavača.

U prvom delu rada razmatran je problem jednostavnog usmeravanja pretrage — izborom rešavača, njegovih heuristika i njihovih parametara, a u zavisnosti od svojstava instance koju je potrebno rešiti. Osnova predloženih metoda za izbor algoritama je sintaksna sličnost formula koja se odražava na njihovu grafovsku strukturu. Ova sličnost je prvi put pouzdano ustanovljena i analizirana pomoću originalne mere sličnosti grafova (koja se pokazala korisnom i u drugim domenima). Praktični pristupi merenju sličnosti formula se zbog računske efikasnosti ipak zasnivaju na numeričkim atributima iskaznih formula. Predložene su dve jednostavne metode izbora algoritma zasnovane na algoritmu k najbližih suseda. Prva tehnika, `ArgoSmArT` se zasniva na klasifikaciji instance u jednu od unapred zadatih familija za koje su poznati algoritmi koji ih efikasno rešavaju. Instanca se rešava algoritmom koji odgovara familiji u koju je instanca klasifikovana. Druga tehnika, `ArgoSmArT k-NN` se zasniva na nalaženju nekoliko sličnih instanci u trening skupu za koje je poznato vreme rešavanja pomoću svih algoritama kojima sistem raspolaže. Instanca se rešava algoritmom koji se najbolje ponaša na pronađenim instancama. Tehnika `ArgoSmArT` je pogodnija za izbor konfiguracije SAT rešavača, a `ArgoSmArT k-NN` za izbor samog SAT rešavača. Tehnika `ArgoSmArT k-NN` se pokazala značajno efikasnijom od najvažnijeg i pritom vrlo složenog sistema za izbor SAT rešavača — sistema `SATzilla`. Pored problema izbora KNF SAT rešavača i njihovih heuristika, razmatran je i problem izbora ne-KNF SAT rešavača u kojem fokus nije bio na tehnikama izbora rešavača, pošto se predložene tehnike direktno primenjuju i na taj problem, već na atributima kojima se ne-KNF instance mogu opisati, a koji do sad nisu predloženi. Rezultati u ovom domenu su pozitivni, ali za sada ograničeni. Osnovni razlog za to je nedostatak većeg broja ne-KNF rešavača raznovrsnog ponašanja, što ne iznenađuje s obzirom da je ova vrsta rešavača tek u svom povoju.

Pored konstrukcije efikasnog sistema za izbor SAT rešavača, prikazana je i metodologija poređenja SAT rešavača zasnovana na statističkom testiranju hipoteza. Potreba za ovakvom metodologijom proizilazi iz velike varijacije vremena rešavanja jedne formule od strane jednog SAT rešavača, što može dovesti do različitih redosleda SAT rešavača prilikom poređenja njihovih performansi ili rangiranja, što je i eksperimentalno demonstrirano. Predložena metodologija pruža ocenu statističke značajnosti testiranja i ocenu veličine efekta, poput verovatnoće da jedan SAT rešavač bude brži od drugog.

Drugi deo rada se odnosi na uopštavanje CDCL sistema pretrage na fragmente logike prvog reda. Predloženi sistem može predstavljati osnovu za efikasno dokazivanje u nekoj logici ukoliko je u njoj moguće definisati pravila rezolucije i faktorisanja. Ova pravila su definisana za jedno uopštenje koherentne logike. Za ovaj sistem su dokazana svojstva potpunosti i saglasnosti. Sistem ima nekoliko važnih karakteristika koje su posledica prethodno sprovedene analize izazova u dokazivanju u koherentnoj logici. Sistem omogućava rezonovanje prvog reda, umesto bazno, što je karakteristika svih postojećih dokazivača za koherentnu logiku. Takođe, sistem koristi povratne skokove i učenje lema. Prilikom dizajna sistema, posebna pažnja je posvećena mogućnosti da se na osnovu rada dokazivača generišu čitljivi dokazi. Ova mogućnost je jedna od osnovnih prednosti koherentne logike, ali to nije lako postići pri korišćenju CDCL sistema pretrage. Jedno od svojstava ovog sistema proisteklo iz potrebe za čitljivim dokazima je očuvanje kvantifikatora prilikom dokazivanja i vrlo je nekarakteristično za postojeće CDCL sisteme. Takođe, prednost formulisanja CDCL sistema je mogućnost prenošenja heuristika koje su se već pokazale korisne u rešavanju SAT problema.

Nad predloženim sistemom, definisana je procedura `Calypso`, za dokazivanje u proširenoj koherentnoj logici, koju je moguće primeniti i na standardnu koherentnu logiku. Predloženo je proširenje algoritma Rete koje omogućava nalaženje literala koji se mogu propagirati, detekciju konflikata i predlaganje literala za pravilo Decide. Procedura `Calypso` je implementirana u jeziku C++. Evaluirana je na reprezentativnim problemima zapisanim u koherentnoj logici i na njima se pokazala superiornom u odnosu na druge dokazivače za koherentnu logiku, kao i dokazivač `Vampire`, najefikasniji dokazivač za logiku prvog reda.

Na osnovu rezultata izloženih u ovom radu, može se zaključiti da je potvrđena njegova osnovna teza — da se sistem pretrage koji se koristi u CDCL SAT rešavačima može značajno unaprediti jednostavnim usmeravnjem, a da se takođe može uspešno formulisati i za fragmente logike prvog reda kao što je koherentna logika, kao i da su dati konkretni odgovori na pitanje kako se to može uraditi.

Ključne reči: automatsko dokazivanje teorema, pretraga, SAT rešavači, koherentna logika, istraživanje podataka

Naučna oblast: računarstvo

Uža naučna oblast: automatsko dokazivanje teorema

UDK broj: 004.83:007.5(043.3)

Disertation title: Guiding search in automated theorem proving

Abstract: In this thesis the problem of guiding search in automated theorem proving is considered. The thesis consists of two parts that have the CDCL search system, the system intensively used by modern SAT solvers, as their common topic.

In the first part of the thesis a simple approach to guiding search is considered — guiding by the selection of the solver, its heuristics, and their parameters, based on the properties of an instance to be solved. The basis of the proposed methods for algorithm selection is syntactical similarity of formulae which is reflected in their graph structure. This graph similarity is established and analyzed by using an original graph similarity measure (which turned out to be useful in other contexts, too). Yet, practical approaches to measuring similarity of formulae are based on their numerical features due to the computational complexity issues. Two simple methods for algorithm selection, based on k nearest neighbors, were proposed. The first technique, `ArgoSmArT` is based on classification of instance in one of the predefined families for which the efficient algorithms are known. The instance is solved by algorithm corresponding to the family to which the instance was classified. The second technique, `ArgoSmArT k-NN` is based on finding several similar instances in the training set for which the solving times by all considered algorithms are known. The instance is solved by the algorithm that behaves the best on those instances. `ArgoSmArT` technique is better suited for configuration selection of a SAT solver, and `ArgoSmArT k-NN` for SAT solver selection. `ArgoSmArT k-NN` technique showed to be more efficient than the most important and very complex system for SAT solver selection — `SATzilla` system. Apart from CNF SAT solver selection, the problem of non-CNF SAT solver selection is considered. The focus was not on solver selection techniques, since the proposed techniques are directly applicable, but on the attributes that can be used to describe non-CNF SAT instances, which have not been proposed earlier. The results in this domain are positive, but still limited. The main reason for that is the lack of greater number of non-CNF SAT solver of different behaviour, which is not surprising, having in mind that this kind of solvers is in its early stage of development.

Apart from construction of efficient SAT solver selection system, the methodology of SAT solver comparison, based on statistical hypothesis testing is proposed. The need for such a methodology comes from great run time variations of single instance solving by a solver, which can result in different SAT solver orderings when one tries to compare their performance or rank them, as experimentally demonstrated. The proposed methodology gives the estimate of statistical significance of the performed test and the estimate of the effect size, for instance the probability of a

solver being faster than another.

The second part of the thesis is concerned with generalizing CDCL search system to fragments of first order logic. The proposed system can be used as a basis for efficient proving in some fragment if the rules of resolution and factoring are specified for that fragment. These rules are defined for an extension of coherent logic. The soundness and completeness of the system are proved. The system has several distinguishing features which are a consequence of previously performed analysis of challenges in coherent logic theorem proving. The system enables first order reasoning, instead of ground one characteristic for all existing coherent logic provers. Moreover, it introduces backjumps and lemma learning. The special attention in system design was paid to the possibility of generating readable proofs by the prover implementing the system. This possibility is one of the greatest qualities of coherent logic, but it is not easy to achieve if CDCL search system is used. One of the properties of the system that came from the need for generation of readable proofs is preservation of quantifiers in proving process which is rather unusual for existing CDCL systems. Another advantage of the proposed CDCL system is the possibility of transfer of heuristics which are already successfully employed in SAT solving to other domains.

Based on the proposed system, the proof procedure `Calypso` for extended coherent logic was defined which can also be used in standard coherent logic. The extension of Rete algorithm which enables detection of conflicts and literals to be propagated or decided is proposed. Procedure `Claypso` is implemented in C++. It was evaluated on a representative coherent logic problems and it showed superior to other coherent logic provers and also the prover `Vampire`, the most efficient prover for first order logic.

Based on the results presented in this thesis, it can be concluded that the main hypothesis of this work, that the search system used in CDCL SAT solvers can be significantly improved by simple guiding and that it can be successfully formulated for fragments of first order logic such as coherent logic, was confirmed and that the concrete answers on how to do that were provided.

Keywords: automated theorem proving, search, SAT solvers, coherent logic, data mining

Research area: computer science

Research subarea: automated theorem proving

UDK number: 004.83:007.5(043.3)

Sadržaj

Glava 1

Uvod

U prethodne dve decenije automatski dokazivači teorema su doživeli velika unapređenja na planu algoritama, heuristika i implementacionih tehnika. Povećanje efikasnosti kojim su ova unapređenja rezultovala omogućilo je primenu dokazivača teorema na probleme sa stotinama hiljada aksioma. Ovo je učinilo mogućim automatsko rešavanje mnogih praktičnih problema poput automatske verifikacije i dizajna hardvera i softvera, automatskog planiranja i raspoređivanja, dokazivanja matematičkih teorema i drugih.

Dve ključne odluke u dizajnu automatskog dokazivača teorema za konkretnu teoriju su izbor skupa pravila čijom se primenom dokazivanje vrši i izbor tehnika kojima se usmerava primena ovih pravila. Interakcija između ova dva izbora je jedna od najvažnijih tema u razvoju automatskih dokazivača teorema i obično najvažniji razlog njihove uspešnosti ili neuspešnosti u primenama. U slučaju mnogih dokazivača je teško postaviti strogu granicu između pravila izvođenja i pravila koja služe za navođenje, pa se diskusija u nastavku može smatrati specifičnom za neke familije dokazivača. Pre svega, misli se na rezolucijske dokazivače za logiku prvog reda, dokazivače zasnovane na rezonovanju unapred, SAT rešavače i na SMT rešavače. Zajednička crta ovih dokazivača je eksplicitna ili implicitna povezanost sa rezolucijom i mogućnost pružanja rezolucijskih dokaza.

Pravila izvođenja. Pravila izvođenja koja dokazivač koristi opisuju transformacije kojima se od postojećih formula dobijaju nove formule. Primeri ovakvih pravila su pravilo rezolucije i pravila prirodne dedukcije [37]. Trenutno najefikasnji dokazivači teorema za logiku prvog reda kao pravilo izvođenja koriste upravo pravilo rezolucije. Jedan od rezolucijskih dokazivača — `Vampire`, već godinama ubedljivo dominira takmičenjima automatskih dokazivača teorema za logiku prvog reda. U dokazivanju teorema iskazne logike najefikasniji dokazivači, zasnovani na DPLL pro-

ceduri [23, 22], koriste pravilo rezolucije u procesu *analize konflikta* i *učenja*, kojima najviše duguju za svoju efikasnost. Primena pravila izvođenja ne mora uvek biti eksplicitna. Primera radi, u originalnoj DPLL proceduri nema eksplicitnih pravila izvođenja, već su data samo pravila pretrage. Međutim, svakom stablu pretrage indukovanom izvršavanjem DPLL procedure na nezadovoljivoj iskaznoj formuli do njenog zaustavljanja, odgovara neko rezolucijsko pobijanje te iskazne formule, koje se može izvesti na osnovu stabla pretrage. U modernoj DPLL proceduri [13], primena pravila rezolucije je eksplicitna i dobijene rezolvente se mogu dodati u skup klauza. Izbor rezolventi koje će biti dodate u skup klauza je vođen nekom heuristikom. Pravilo rezolucije je korisno i u slučaju zadovoljivih formula — u izgradnji modela.

Usmeravanje primene pravila izvođenja. Pravila izvođenja se često mogu primeniti na različite formule iz skupa postojećih formula. Na primer, pravilo rezolucije se često može primeniti na veći broj klauza. U slučaju nekih sistema može postojati više različitih primenljivih pravila. Stoga, proces dokazivanja nije pravolinijski, već se može razmatrati kao proces pretrage. Vreme dokazivanja teoreme može drastično varirati u zavisnosti od načina na koji se pravila izvođenja primenjuju. Primera radi, ukoliko se dokazivanje vrši *rezonovanjem unapred*, odnosno primenom aksioma čije su pretpostavke zadovoljenje, tvrđenje $\forall x\ (p(x) \Rightarrow t(x))$ se iz aksioma

$$A1:\ \forall x\ (p(x) \Rightarrow q(x) \vee r(x)) \quad A2:\ \forall x\ (p(x) \Rightarrow s(x)) \quad A3:\ \forall x\ (s(x) \Rightarrow t(x))$$

može dokazati na dva načina neformalno prikazana na slici 1.1. Drugi dokaz je očigledno efikasniji u smislu veličine dokaza, ali automatski dokazivač teorema može proizvesti i prvi dokaz.

Važna tema u analizi sistema pravila na kojima se automatski dokazivači zasnivaju je složenost dokaza (eng. proof complexity) koje dopuštaju. Na primer, u slučaju dokazivanja nezadovoljivosti iskaznih formula, moderna DPLL procedura dopušta dokaze eksponencijalno manje veličine u odnosu na dokaze koji se mogu dobiti korišćenjem originalne DPLL procedure [81]. Pitanje koje se prirodno nameće je kako iskoristiti takvu mogućnost, odnosno kako primenjivati pravila sistema tako da se u što manjem broju koraka dođe do što kraćih dokaza. Ispostavlja se da je u slučaju sistema koji dopuštaju kraće dokaze, problem optimalnog ili makar kvalitetnog izbora pravila koja se primenjuju puno teži nego u slučaju sistema sa većom složenošću dokaza (ovo zapažanje je potvrđeno i empirijski i teorijski [1]). Ovim tehnike za usmeravanje primene pravila, odnosno pretrage za dokazom, dodatno dobijaju na važnosti.

Dokaz 1:

- Neka važi $p(a)$ za neko a
- na osnovu aksiome A1 i $p(a)$, važi $q(a) \vee r(a)$
- Neka važi $q(a)$
  - Na osnovu aksiome A2 i $p(a)$, važi $s(a)$
  - Na osnovu aksiome A3 i $s(a)$, važi $t(a)$
  - Stoga, važi $p(a) \Rightarrow t(a)$
  - Kako je a proizvoljno, važi $\forall x\ (p(x) \Rightarrow t(x))$
- Neka važi $r(a)$
  - Na osnovu aksiome A2 i $p(a)$, važi $s(a)$
  - Na osnovu aksiome A3 i $s(a)$, važi $t(a)$
  - Stoga, važi $p(a) \Rightarrow t(a)$
  - Kako je a proizvoljno, važi $\forall x\ (p(x) \Rightarrow t(x))$

Dokaz 2:

- Neka važi $p(a)$ za neko a
- Na osnovu aksiome A2 i $p(a)$, važi $s(a)$
- Na osnovu aksiome A3 i $s(a)$, važi $t(a)$
- Stoga, važi $p(a) \Rightarrow t(a)$
- Kako je a proizvoljno, važi $\forall x\ (p(x) \Rightarrow t(x))$

Slika 1.1: Dva dokaza tvrđenja $\forall x\ (p(x) \Rightarrow t(x))$.

Tehnike usmeravanja pretrage u automatskom dokazivanju teorema su brojne i mogu se odnositi na različite aspekte pretrage. Neke od njih su algoritamskog karaktera, poput povratnih skokova [89] koji omogućavaju eliminaciju grananja u dokazu, ako ta grananja nisu relevantna za zatvaranje grane dokaza. Takvo je grananje u prvom dokazu na slici 1.1. Ovakve tehnike omogućavaju odsecanje velikih delova prostora pretrage. Druga vrsta tehnika je heuristička i fokusira se na inteligentan izbor pravila koje treba primeniti ili načina na koji pravilo treba primeniti. U primeru na slici 1.1, heuristika koja daje prioritet primeni aksioma koje ne dovode do grananja u dokazu bi bila dovoljna da dokazivač proizvede željeni dokaz. U slučaju problema zadovoljivosti iskaznih formula, heuristike se bave izborom promenljive po čijoj vrednosti se vrši grananje u pretrazi, izbor vrednosti te promenljive, odnosno grane kojom će se pretraga nastaviti, izbor trenutka za otpočinjanje pretrage iznova i drugim sličnim aspektima pretrage. U daljem tekstu će u različitim kontekstima biti razmatrane obe vrste tehnika za usmeravanje pretrage.

Usmeravanje pretrage izborom ili podešavanjem algoritma. Moderni dokazivači teorema često omogućavaju podešavanje različitih aspekata svog rada. Neki

dokazivači implementiraju različite tehnike za usmeravanje pretrage i omogućavaju korisniku izbor između njih. Dodatno, svaka od tehnika može imati parametre koji preciznije definišu njeno ponašanje, koji se takođe mogu birati prilikom pokretanja dokazivača. Ponašanje različitih konfiguracija istog dokazivača na istim problemima može biti drastično različito i naizgled vrlo nepredvidljivo. Tim pre, razumljivo je i da ponašanje različitih dokazivača na istim problemima može biti vrlo različito čak i ako su oni zasnovani na istim ili sličnim algoritmima. Jedan način usmeravanja pretrage predstavlja izbor dokazivača ili njegove konfiguracije za konkretan problem koji je potrebno rešiti. Ovaj izbor se vrši jednom, pre početka dokazivanja i ne podrazumeva promene u toku rada dokazivača (bilo bi moguće razmatrati i problem konfigurisanja u toku rada dokazivača, ali se taj problem u ovom radu ne razmatra). Problem izbora dobrog dokazivača, tehnika pretrage koje će on koristiti i vrednosti njihovih parametara je netrivijalan i zahteva puno ljudskog vremena. U tabeli 1.1 prikazan je broj teorema iz unapred datog skupa koji svaki od dokazivača može da dokaže u određenom vremenu, kao i maksimalan broj teorema koje bi mogle biti dokazane ukoliko bi se znalo koji je dokazivač najbolje primeniti. Konkretan izbor dokazivača i problema o kojima je reč u ovom trenutku nije relevantan. Kao što

Tabela 1.1: Broj dokazanih teorema iz nekog skupa za 6 dokazivača i maksimalan broj teorema koje bi se mogle dokazati izborom najboljeg dokazivača za svaku teoremu.

Dokazivač	1	2	3	4	5	6	Maks.
Broj teorema	282	252	244	238	259	243	444

se može primetiti, uz poznavanje najboljeg dokazivača, broj dokazanih teorema se može značajno povećati u odnosu na bilo koji pojedinačni dokazivač. Stoga je problem automatskog izbora dokazivača ili njegovog automatskog podešavanja, kako bi pretraga za dokazom bila što kraća, od velikog značaja. Taj problem je prvi problem kojim se ovaj rad bavi.

CDCL sistem pretrage. Tehnike za usmeravanje pretrage se nekad formulišu za vrlo specifične aspekte sistema pravila koji se koristi i mogu biti neprimenljive u drugim kontekstima. Međutim, neke tehnike pretrage se mogu prenositi iz jednog konteksta u drugi. Identifikovanje takvih tehnika koje su uspešne u više domena i njihova eksploatacija su od posebnog značaja. Jedan sistem pretrage, zasnovan na *konfliktima vođenom učenju klauza* (eng. *conflict driven clause learning*), skraćeno CDCL, se pokazao kao najuspešniji u ispitivanju iskazne zadovoljivosti i primenjen je u mnogim SAT rešavačima, ali i u SMT rešavačima koji vrše ispitivanje zadovoljivosti formula logike prvog reda. Dodatno, formulisane su i vrlo uspešne heuristike za

specifične aspekte primene pravila ovog sistema. Zbog toga, ovaj sistem pretrage je zanimljiv kandidat za prenošenje u bogatije logike.

Koherentna logika. Jedan fragment logike provog reda, koherentna ili geometrijska logika, je od posebnog značaja u primenama automatskog dokazivanja teorema. Formule koherentne logike su sledećeg oblika:

$$\forall \vec{v}\ (p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v}) \Rightarrow \exists \vec{y}\ Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v}, \vec{y})),$$

gde $p_1, \ldots, p_n$ označavaju atomičke formule, a $Q_1, \ldots, Q_m$ konjunkcije atomičkih formula. Različiti problemi iz oblasti poput geometrije i algebre se mogu prirodno predstaviti u ovoj logici bez ikakvih ili sa vrlo malim transformacijama polaznih formula. Zahvaljujući prirodnom deduktivnom sistemu zasnovanom na rezonovanju unapred, koherentna logika je posebno pogodna za automatsko generisanje čitljivih dokaza matematičkih teorema. Dokazivači koji se trenutno koriste u koherentnoj logici uglavnom koriste vrlo jednostavne i neefikasne sisteme pretrage. Zbog te činjenice i svog značaja, koherentna logika je prirodan domen za formulisanje tehnika pretrage analognih onima koje se već uspešno koriste u drugim domenima. Prenošenje tehnika CDCL pretrage u domen koherentne logike je drugi problem kojim se ovaj rad bavi.

Osnovna teza ovog rada je da se *sistem pretrage koji se koristi u CDCL SAT rešavačima može značajno unaprediti jednostavnim usmeravnjem, a da se takođe može uspešno formulisati i za fragmente logike prvog reda kao što je koherentna logika.*

Doprinosi ovog rada su sledeći:

- Uočena je i empirijski analizirana veza između sintaksne sličnosti iskaznih formula i sličnosti algoritama koji su najpogodniji za njihovo rešavanje. Ova veza je uočena korišćenjem mere sličnosti grafova koja je formulisana u ovu svrhu. Rad o tome je objavljen u časopisu *Intelligent Data Analysis* [75] (potpoglavlje 3.1.2).

- Na osnovu uočene sličnosti, predložena su dva algoritma za izbor i konfigurisanje SAT rešavača zasnovanih na CDCL sistemu pretrage. Predloženi algoritmi su mnogo jednostavniji od postojećih, a postižu značajno bolje rezultate od njih u empirijskoj evaluaciji. Jedan rad o tim algoritmima je prikazan na konferenciji *Theory and Applications of Satisfiability Testing* [73], a drugi ob-

javljen u časopisu *Artificial Intelligence Review* [77] (poglavlja 3.2, 2.4, 3.4 i 3.5).

- Predložen je metod poređenja SAT rešavača koji je zbog velikog variranja vremena rešavanja iskaznih formula zasnovan na statističkom testiranju hipoteza i omogućava procenu statističke značajnosti poređenja. Rad o ovom metodu je prikazan na konferenciji *Theory and Applications of Satisfiability Testing* [74] (poglavlje 3.6).

- Predloženo je uopštenje CDCL sistema pretrage koje se može instancirati kako za koherentnu logiku i jedno njeno proširenje tako i za iskaznu logiku, ali i za druge logike u kojima bi se na pogodan način definisala pravila rezolucije i faktorisanja. Svojstva ovog sistema su teorijski analizirana i dokazana je njegova saglasnost i potpunost. Rad o ovom sistemu pravila prikazan je na konferenciji *Conference On Intelligent Computer Mathematics u sekciji Claculemus* [76] (poglavlje 4.4).

- Implementiran je dokazivač teorema za koherentnu logiku koji odgovara predloženom CDCL sistemu i proizvodi dokaze u sistemu koherentne rezolucije. Empirijskom evaluacijom je pokazano da je novi dokazivač efikasniji od postojećih dokazivača za koherentnu logiku i najefikasnijeg opšteg dokazivača za logiku prvog reda primenjenog na ovom fragmentu. Dokazivač je uspešno primenjen na probleme koji sadrže i po 30000 aksioma (poglavlje 4.5).

Ostatak rada je organizovan na sledeći način:

- U glavi 2 opisane su tehnike pretrage korišćene u modernim SAT rešavačima, sa naglaskom na CDCL sistemu pretrage. CDCL sistem je formulisan u obliku apstraktnog sistema pravila koji ga precizno definiše. Biće prikazani i osnovni rezultati vezani za ispravnost SAT rešavača.

- Glava 3 opisuje tehnike za prilagodljivo rešavanje SAT problema koje na pretragu utiču kroz izbor algoritma, heuristika koje on koristi ili vrednosti njihovih parametara. Ove tehnike su primenjene u slučaju kako klasičnih, KNF, tako i novih, ne-KNF SAT rešavača. U ovoj glavi se diskutuje i poređenje SAT rešavača.

- U glavi 4 je formulisan sistem pretrage za dokazivanje teorema u koherentnoj logici inspirisan CDCL sistemom korišćenim u iskaznoj logici. Dokazana su njegova osnovna svojstva i opisan je dokazivač `Calypso` koji ga implementira. Opisani su i drugi sistemi za dokazivanje u koherentnoj logici.

- Glava 5 daje zaključke ovog rada i opisuje dalje pravce istraživanja koji iz njega proističu.

Glava 2

SAT problem i SAT rešavači

Iskazna logika se bavi istinitošću iskaza, izgrađenih nad iskaznim promenljivim pomoću iskaznih veznika. Iskazne promenljive predstavljaju elementarne iskaze. Način izgradnje složenijih iskaza iz njih je definisan *sintaksom* iskazne logike. Istinitosnu vrednost ispravnih iskaza opisuje *semantika* iskazne logike. Osnovni problem iskazne logike — problem SAT je problem ispitivanja da li je iskazna formula zadovoljiva, to jest, da li postoji dodela vrednosti promenljivim za koju je iskazna formula tačna. Ovaj problem je jedan od fundamentalnih matematičkih problema i ima centralno mesto u teoriji složenosti izračunavanja. Dokazano je da je problem SAT NP-kompletan [19]. Pošto ima široke primene, posvećuje se velika pažnja algoritmima koji bi što efikasnije rešavali instance ovog problema u praksi. Pod *rešavanjem* iskazne formule, podrazumeva se ispitivanje njene zadovoljivosti. Do sada je razvijen veliki broj *SAT rešavača*, sistema koji implementiraju ovakve algoritme.

SAT rešavači se mogu svrstati u dve grupe — potpune i stohastičke. Za datu iskaznu formulu, potpuni SAT rešavači sigurno nalaze zadovoljavajuću valuaciju ili utvrđuju da ona ne postoji, dok stohastički rešavači ne mogu dokazati nezadovoljivost formule iako mogu pokazati njenu zadovoljivost. Njihova prednost je u tome što za neke klase formula, mogu utvrditi zadovoljivost brže od potpunih rešavača. U ovom radu biće razmatrani samo potpuni SAT rešavači.

Moderni potpuni SAT rešavači su najčešće zasnovani na proceduri DPLL [23, 22], opisanoj ranih šezdesetih godina. SAT rešavači su postigli veliki napredak na prelazu prošlog u ovaj vek, kada su se pojavili poznati rešavači kao `SATO`, `CHAFF`, `MiniSAT` i drugi. Ovaj napredak je bio posledica kako konceptualnih, tako i implementacionih unapređenja procedure DPLL koja su vodila do takozvane CDCL varijante DPLL procedure. U daljem tekstu će se pod SAT rešavačima podrazumevati potpuni SAT rešavači.

U ovoj glavi će biti razmotreni osnovni pojmovi iskazne logike, tehnike pretrage

koje se koriste u rešavanju SAT problema, apstraktni sistem pravila koji opisuje CDCL sistem pretrage, heuristike koje se koriste za upravljanje različitim aspektima rada modernih SAT rešavača, kao i rezultati vezani za ispravnost CDCL sistema.

2.1 Osnovni pojmovi iskazne logike

U tekstu se koriste naredne definicije osnovnih pojmova iskazne logike [46, 64].

Definicija 1 Skup iskaznih formula $\mathcal{IF}$ *nad prebrojivim skupom iskaznih slova* P *je skup za koji važi:*

- *iskazna slova (iz skupa* P*) i logičke konstante* $\top$ *i* $\bot$ *su iskazne formule;*
- *ako su* A *i* B *iskazne formule, onda su i* $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ *i* $(A \Leftrightarrow B)$ *iskazne formule.*

U zapisu iskaznih formula, zagrade mogu biti izostavljene i u tom slučaju se podrazumeva prioritet operatora u sledećem poretku, od najvišeg ka najnižem: $\neg$, $\wedge$, $\vee$.

Iskazna slova nazivamo i *iskaznim promenljivim*, a iskazna slova i logičke konstante nazivamo *atomičkim iskaznim formulama*. Atomičke iskazne formule i njihove negacije nazivamo *literalima*. Disjunkcije literala nazivamo *klauzama*.

Ako je p iskazno slovo i l literal nad tim iskaznim slovom, $\bar{l}$ označava literal $\neg p$ ukoliko je $l = p$, a literal p ukoliko je $l = \neg p$.

Definicija 2 *Skup* potformula *formule* A *je najmanji skup formula koje zadovoljavaju sledeće uslove:*

- A *je potformula sama sebi;*
- *Ukoliko je* A *jednako* $\neg B$*, potformule formule* B *su potformule formule* A*; Ukoliko je* A *jednako* $B \wedge C$*,* $B \vee C$*,* $B \Rightarrow C$ *ili* $B \Leftrightarrow C$*, onda je svaka potformula formule* B *i svaka potformula formule* C*, takođe potformula formule* A*.*

Definicija 3 Pozicija *je niz* $a_1.a_2.\ldots.a_n$*, pri čemu je* $n \geq 0$ *i* $a_i \in \{1, 2\}$ *za svako* i $(1 \leq i \leq n)$*. U slučaju kad je* $n = 0$*, pozicija se naziva* praznom *i zapisuje* ϵ*.* Polaritet *može biti bilo koja od vrednosti -1,0,1. Za formulu* A*,* potformula na poziciji π*,* $A|\pi$ *i* polaritet potformule na poziciji π*,* $pol(A, \pi)$*, definišu se na sledeći način:*

- $A|_\epsilon = A$ *i* $pol(A, \epsilon) = 1$*;*
- *ako je* A_π *jedanko* $\neg A_1$*, onda je* $A|_{\pi.1} = A_1$ *i* $pol(A, \pi.1) = -pol(A, \pi)$*;*

- *ako je* $A|_\pi$ *jednako* $A_1 \wedge A_2$ *ili* $A_1 \vee A_2$*, onda je* $A|_{\pi.i} = A_i$ *i* $pol(A, \pi.i) = pol(A, \pi)$ *za* $i = 1, 2$*;*
- *ako je* $A|_\pi$ *jednako* $A_1 \Rightarrow A_2$*, onda je* $A|_{\pi.i} = A_i$ *za* $i = 1, 2$ *i* $pol(A, \pi.1) = -pol(A, A|\pi)$ *i* $pol(A, \pi.2) = pol(A, A|\pi)$*;*
- *ako je* $A|_\pi$ *jednako* $A_1 \Leftrightarrow A_2$*, onda je* $A|_{\pi.i} = A_i$ *i* $pol(A, \pi.i) = 0$ *za* $i = 1, 2$*.*

Definicija 4 *Funkcije koje preslikavaju skup iskaznih slova u skup* $\{0, 1\}$ *se nazivaju* valuacijama*. Polazeći od valuacije* v *konstruiše se funkcija* I_v *koja preslikava skup iskaznih formula u skup* $\{0, 1\}$*. Nju nazivamo* funkcijom istinitosne vrednosti *za valuaciju* v *i definišemo na sledeći način:*

- $I_v(p) = v(p)$*, za svaki element* p *skupa* P*;*
- $I_v(\top) = 1$ *i* $I_v(\bot) = 0$*;*
- $I_v(\neg A) = 1$ *ako je* $I_v(A) = 0$ *i* $I_v(\neg A) = 0$ *ako je* $I_v(A) = 1$*;*
- $I_v(A \wedge B) = 1$ *ako je* $I_v(A) = 1$ *i* $I_v(B) = 1$*;* $I_v(A \wedge B) = 0$ *inače;*
- $I_v(A \vee B) = 0$ *ako je* $I_v(A) = 0$ *i* $I_v(B) = 0$ *;* $I_v(A \vee B) = 1$ *inače;*

Definicija 5 *Iskazna formula* F *je* zadovoljiva *ako postoji valuacija* v *takva da je* $I_v(F) = 1$*. Takva valuacija se naziva* modelom *formule* F*, što se zapisuje* $v \models F$*. U suprotnom,* F *je* nezadovoljiva*. Iskazna formula* F *je* valjana *(*tautologija*) ako za svaku valuaciju* v *važi* $I_v(F) = 1$*. U suprotnom,* F *je* poreciva*. Formula* F *je logička posledica skupa formula* Γ*, što se zapisuje* $\Gamma \models F$*, ukoliko je svaka valuacija koja je model za sve formule iz skupa* Γ*, takođe model i za formulu* F*.*

Definicija 6 *Iskazna formula je u* konjunktivnoj normalnoj formi *(KNF) ako je oblika*

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

pri čemu je svaka od formula A_i $(1 \leq i \leq n)$ *klauza.*

Definicija 7 Supstitucija *(ili* zamena*) formule* C *formulom* D *u formuli* A *je funkcija* $A[C \mapsto D] : \mathcal{IF} \times \mathcal{IF} \times \mathcal{IF} \to \mathcal{IF}$ *definisana na sledeći način:*

- *ako je* A *jednako* C*, onda je* $A[C \mapsto D]$ *jednako* D*;*
- *ako* A *nije* C *i* A *je iskazno slovo, onda je* $A[C \mapsto D]$ *jednako* A*;*
- *ako* A *nije* C *i važi* $A = (\neg B)$ *za neku iskaznu formulu* B*, onda je* $A[C \mapsto D]$ *jednako* $\neg(B[C \mapsto D])$*;*
- *ako* A *nije* C *i važi* $A = B_1 \wedge B_2$*, odnosno* $A = B_1 \vee B_2$*, za neke iskazne formule* B_1 *i* B_2*, tada je* $A[C \mapsto D]$ *jednako* $B_1[C \mapsto D] \wedge B_2[C \mapsto D]$*, odnosno* $B_1[C \mapsto D] \vee B_2[C \mapsto D]$*.*

```
function dpll (F: KNF formula) : bool
begin
    if F = ∅ then return true
    else if ∅ ∈ F then return false
    else if {l} ∈ F then return dpll(F[[l ↦ ⊤]])
    else if pureLiteral(l,F) then return dpll(F[[l ↦ ⊤]])
    else begin
        select l such that l ∈ C and C ∈ F
        if dpll(F[[l ↦ ⊤]]) = true then return true
        else return dpll(F[[l ↦ ⊥]])
    end
end
```

Slika 2.1: Osnovna DPLL procedura.

Primene supstitucija će uvek biti pisane postfiksno.

Definicija 8 *Ako je A iskazna formula, a l literal, onda*

- $A[[l \mapsto \top]]$ *označava formulu* $A[l \mapsto \top][\bar{l} \mapsto \bot]$ *iz koje su uklonjene sve klauze koje sadrže konstantu* $\top$ *i sva pojavljivanja konstante* $\bot$*;*
- $A[[l \mapsto \bot]]$ *označava formulu* $A[[\bar{l} \mapsto \top]]$*;*

2.2 Osnovne procedure pretrage

U ovom poglavlju će biti prikazane osnovne procedure pretrage koje se koriste u rešavanje SAT problema. Prvo će biti opisana osnovna DPLL procedura [22], a potom i njena moderna varijanta definisana koja implementira CDCL sistem pretrage.

2.2.1 Osnovna DPLL procedura

Osnovni algoritam za proveru zadovoljivosti iskaznih formula je Dejvis-Patnam-Logman-Lovelandova procedura (DPLL procedura) [22]. Formula čija se zadovoljivost ispituje mora biti u konjunktivnoj normalnoj formi. Formula se predstavlja kao skup klauza, a klauza kao skup literala. DPLL procedura je data na slici 2.1 [64]. Funkcija `pureLiteral` ima vrednost `true` ukoliko je l literal koji se javlja u formuli F, pri čemu se $\bar{l}$ ne javlja u formuli F.

DPLL algoritam se može koristiti i za proveru da li je formula valjana, tako što se krene od negacije polazne formule i ako je ona nezadovoljiva, zaključuje se da je polazna formula valjana.

```
function dpll (F: KNF formula) : bool
begin
    M := []
    repeat begin
        if v_M ⊨ F then return true
        else if v_M ⊨ ¬F then
            if M = M'| l M'' and | ∉ M'' then M := M' l̄
            else return false
        else if {l} ∪ C ∈ F and v_M ⊨ ¬C then M := M l
        else begin
            select l such that l ∈ C and C ∈ F and l ∉ M and l̄ ∉ M
            M := M| l
        end
    end
end
```

Slika 2.2: Iterativna DPLL procedura.

2.2.2 Moderna DPLL procedura

Većina modernih SAT rešavača se zasniva na DPLL proceduri, ali često sa kompleksnim izmenama. Modifikovanje formule i njeno prosleđivanje u rekurziju je neefikasno u praksi. Takođe, nalaženje literala l za koji važi `pureLiteral(l,F)` se upraksi ispostavlja kao neefikasno. Stoga se DPLL procedura implemenitra iterativno, bez modifikovanja formule. Umesto toga, gradi se lista literala M koji se smatraju tačnim i ona se menja dok valuacija v_M koja joj odgovara ne zadovolji formulu ili dok se ne ustanovi da takva valuacija ne postoji. Iterativna procedura data je na slici 2.2. Ova procedura predstavlja prvi korak ka modernoj DPLL proceduri.

Izbor literala koji se dodaje u listu M se naziva *odlukom* (eng. decision). Literalu koji je *odlučen* u listi M prethodi oznaka $|$. Prilikom odluke, najčešće se odovojeno bira promenljiva čija će vrednost biti zadata, kao i njen inicijalni polaritet, tj. određuje se da li pri odluci promenljiva u valuaciji v_M prvo dobija vrednost 0 ili 1. Kada se ispostavi da neka promenljiva mora imati određenu vrednost (u slučaju kad važi $\{l\} \cup C \in F$ `and` $v_M \models \neg C$), radi se o *implikaciji* (eng. implication). Situacija u kojoj važi $v_M \models \neg F$, naziva se *konfilkt* (eng. conflict). Kada se desi konflikt, potrebno je izvršiti promenu prethodne odluke kako bi se pokušala naći neka valuacija koja zadovoljava formulu.

Kako bi se procedura dalje poboljšala, moguće je tražiti razloge zbog kojih je do konflikta uopšte došlo. Oni se formulišu u obliku *naučenih klauza* (eng. learnt clauses) koje se dodaju u početni skup klauza, a konstruišu se nekim mehanizmom rezonovanja (na primer iskaznom rezolucijom), na osnovu informacija koje su raspoložive o konfliktu. Ako je, na primer, klauza $\{x, y, z\}$ postala netačna u toku

rešavanja, onda $\{\neg x, \neg y, \neg z\}$ nazivamo *razlogom konflikta* (eng. reason set of the conflict) [28]. Vrednost 0 je implicirana za promenljivu x, na primer, tako što je prisutna klauza $\{\neg u, \neg v, \neg x\}$, a promenljive u i v imaju vrednost 1 u tekuć oj parcijalnoj valuaciji. Odatle sledi da i skup $\{u, v, \neg y, \neg z\}$ takođe mora voditi ka konfliktu. Taj konflikt se može sprečiti dodavanjem klauze $\{\neg u, \neg v, y, z\}$ za koju kažemo da je *naučena* (eng. learnt). Kako bi se pretraga nastavila, potrebno je otkloniti konflikt i da bi se to učinilo, preduzima se *povratni skok* pri čemu se parcijalna valuacija koja se gradi skraćuje tako da naučena lema više nije netačna u njoj. Kako broj naučenih klauza obično raste, njihovo analiziranje opterećuje rešavač, pa stoga ove klauze u nekom trenutku treba brisati (one su ionako uvek izvedene iz polaznih klauza). Ovaj postupak se naziva *zaboravljanje*. Kako je drvo pretrage koju DPLL procedura sprovodi veliko, rešavač može veliku količinu vremena potrošiti pretražujući grane koje ne sadrže rešenje. Da bi se to sprečilo, proces rešavanja s vremena na vreme *otpočinje iznova* (eng. restart). Da otpočinjanje iznova ne bi ugrozilo potpunost pretrage, ono se obično vrši sve ređe i ređe u toku rešavanja.

Tehnike povratnih skokova i učenja lema, pored nekih implementacionih tehnika, su najzaslužnije za efikasnost modernih SAT rešavača i čine jezgro takozvanog CDCL sistema pretrage. Povratni skokovi omogućavaju da se posle konflikta ne ispituju alternative odluka koje nisu bile zaslužne za dolaženje do konflikta i time se vrši veliko odsecanje prostora pretrage. Ovo odsecanje nazivamo *odsecanjem unazad*. Uloga učenja lema je da se u budućnosti pretrage skrati izvođenje konflikta koji bi nastao iz istih razloga kao konflikt na osnovu kojeg je lema izvedena. Dakle, umesto da se do istog konflikta ponovo dođe pretragom, on biva otkriven zahvaljujući prisustvu leme. Ovakvo odsecanje delova prostora pretrage naziva se *odsecanjem unapred*. Značaj učenja lema u modernim SAT rešavačima je inspirisao novi pogled na modernu DPLL proceduru — umesto posmatranja DPLL procedure kao procedure pretrage, ona se može smatrati mehanizmom usmeravanja rezolucije u cilju izvođenja prazne klauze i formiranja rezolucijskog dokaza nezadovoljivosti formule.

2.3 Apstraktni sistem pravila za CDCL pretragu

Moderni SAT rešavači, zasnovani na CDCL (eng. conflict driven clause learning) sistemu pretrage, su često među sobom vrlo slični. Mnogi od njih su nastali manjim ili većim modifikovanjem rešavača `MiniSAT` [28]. Često se razlikuju pre svega u heuristikama koje upravljaju nekim aspektima njegovog rada. Dodatno, implementacije rešavača nisu jednostavne ni lake za razumeti. Zarad boljeg razumevanja SAT rešavača korisno je razdvojiti različite komponente njihovog dizajna i uočiti u

čemu se sastoji osnovni mehanizam pretrage, koje komponente čine implementaciona poboljšanja, a šta su heuristike koje upravljaju nekim aspektima rada rešavača. U tom cilju predložena su dva apstraktna sistema pravila koji opisuju samo osnovni mehanizam pretrage i apstrahuju sve druge aspekte modernih SAT rešavača [72, 53]. Ovakvi sistemi omogućavaju analizu teorijskih svojstava SAT rešavača poput saglasnosti, zaustavljanja i potpunosti bez opterećivanja implementacionim detaljima ili heuristikama koje ne utiču na ove aspekte, već samo na efikasnost izvršavanja. Apstraktni sistemi pravila su i osnova za formalnu analizu i dokazivanje svojstava SAT rešavača u sistemima za formalno dokazivanje teorema [66] o čemu će više reči biti u poglavlju 2.5. U nastavku je analiziran apstraktni sistem Krstića i Goela [53].

Apstraktni sistem pravila za SAT problem opisuje proces provere zadovoljivosti iskazne formule kao lanac stanja. Stanje je uređena trojka (F, M, C) gde je F skup klauza, M lista literala, a C može biti klauza ili poseban simbol no_cflct. Relacija prelaska $\rightarrow$ nad stanjima je definisana tako da su dva stanja S_1 i S_2 u relaciji $S_1 \rightarrow S_2$ ukoliko se stanje S_2 može dobiti iz stanja S_1 primenom nekog od pravila sa slike 2.3, čiji opis sledi. Stanje S je završno ukoliko ne postoji stanje S' takvo da $S \rightarrow S'$. Završno stanje je *prihvatajuće* ukoliko važi $C = no\_cflct$, a *odbacujuće* u suprotnom. Svaki lanac stanja oblika $S \rightarrow \ldots S_i \rightarrow S_{i+1} \rightarrow \ldots$ je konačan, odnosno primena pravila ovog sistema se zaustavlja. Takođe, sistem je saglasan i potpun, odnosno odgovor koji daje na pitanje zadovoljivosti bilo koje iskazne formule je tačan, i za svaku iskaznu formulu je u stanju da dâ odgovor. U prihvatajućem završnom stanju, lista M indukuje valuaciju iskaznih promenljivih v_M takvu da je $v_M(v) = 1$ ukoliko $v \in M$, a $v_M(v) = 0$ ukoliko $\neg v \in M$ i koja zadovoljava polaznu formulu F. Valuacija v je dobro definisana zahvaljujući tome što je lista M u svakom stanju neprotivrečna.

Pravila na slici 2.3 opisuju načine na koje se mogu menjati elementi stanja. U nastavku su data objašnjenja ovih pravila.

Decide opisuje pravljenje pretpostavke o tačnosti nekog literala. Ukoliko ni literal ni njegova negacija nisu u listi M, onda se on može dodati u nju.

Unit propagation opisuje korak jednostavnog zaključivanja. Ukoliko su svi literali neke klauze, osim jednog (takvog da ni on ni njegova negacija nisu u M), netačni u valuaciji v_M koju indukuje lista M, onda taj literal mora biti tačan i može se dodati u listu M.

Conflict opisuje ustanovljavanje postojanja konflikta. Ukoliko su svi literali neke klauze netačni u valuaciji v_M, konfliktna klauza se pamti u elementu stanja C kako

Decide:
$$\frac{l \in L \qquad l, \bar{l} \notin M}{M := M\ |l}$$
Unit propagation:
$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$
Conflict:
$$\frac{C = no\_cflct \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \in M}{C := \{l_1, \ldots, l_k\}}$$
Explain:
$$\frac{l \in C \qquad l \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \prec l}{C := C \cup \{l_1, \ldots, l_k\} \setminus \{l\}}$$
Learn:
$$\frac{C = \{l_1, \ldots, l_k\} \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \notin F}{F := F \cup \{\bar{l}_1 \vee \ldots \vee \bar{l}_k\}}$$
Backjump:
$$\frac{C = \{l, l_1, \ldots, l_k\} \qquad \bar{l} \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad \text{level } l > m \geq \text{level } l_i}{C := no\_cflct \qquad M := M^m\ \bar{l}}$$
Forget:
$$\frac{C = no\_cflct \qquad c \in F \qquad F \setminus c \models c}{F := F \setminus c}$$
Restart:
$$\frac{C = no\_cflct}{M := M^{[0]}}$$

Slika 2.3: Apstraktni sistem pravila za SAT predložen od strane Krstića and Goela (L je skup svih literala nad datim skupom iskaznih promenljivih, $l_i \prec l_j$ označava da literal l_i prethodi literalu l_j u listi M, $\bar{l}$ označava literal suprotnog polariteta od literala l, | označava da je sledeći literal odlučen, level l označava broj odluka zaključno sa literalom l, a M^m označava prefiks liste M pre odluke $m + 1$)

bi se daljim objašnjavanjem konflikta mogli naći njegovi uzroci. Proces kojim se dolazi do ovih uzroka se naziva *analiza konflikta*.

Explain opisuje korak analize konflikta pri kojem se uočava jedan literal iz konfliktne klauze C i nalazi se klauza iz F koja ga je mogla propagirati. Objašnjavanje se vrši rezolviranjem te klauze sa trenutnom konfliktnom klauzom i tako nastaje nova konfliktna klauza.

Learn omogućava da se trenutna konfliktna klauza doda u skup klauza ukoliko već nije u njemu. To je moguće jer je ona posledica polaznog skupa klauza jer je nastala njihovim rezolviranjem.

Backjump omogućava vraćanje u pretrazi. Pri tome se preskaču odluke koje nisu bile relevantne za dolazak do konflikta i njihove alternative se ne ispituju. Povratni skok se vrši skraćivanjem liste M tako da se ukloni poslednji literal l iz nje koji

klauzu C čini konfliktnom. Uslovi primene ovog pravila garantuju da posle njegove primene, ova klauza može da propagira literal $\bar{l}$. Pri povratnom skoku se može iz liste M ukloniti veliki broj odluka čije se alternative ne ispituju.

Forget omogućava uklanjanje impliciranih (izvedenih) klauza iz skupa klauza. Razlog za zaboravljanje je pre svega praktične prirode — u slučaju prevelikog broja klauza, SAT rešavači mogu raditi vrlo neefikasno.

Restart se realizuje uklanjanjem svih literala iz liste M osim onih koji prethode prvoj odluci. Ovo omogućava izlazak iz delova prostora pretrage u kojima pretraga predugo traje sa nadom da će se do rešenja brže doći u nekom drugom delu pretrage. Otpočinjanjem iznova se ne poništava prethodni rad rešavača. Jedan od razloga za to je što se zadržavaju naučene klauze koje će eliminisati delove prostora pretrage koji su već obiđeni.

2.4 Heuristike za usmeravanje pretrage u CDCL sistemu

Mnogi aspekti ponašanja SAT rešavača se mogu preciznije definisati i to se postiže uvođenjem različitih heuristika. Za neke od tih aspekata će biti detaljnije opisane heuristike koje se najčešće koriste.

Heuristike izbora promenljive

Heuristike izbora promenljive se odnose na izbor promenljive u vezi sa čijom vrednošću će biti doneta odluka. U nastavku će biti opisane neke od njih.

Jednostavna heuristika je heuristika pseudoslučajnog izbora jedne od promenljivih čija vrednost u datom trenutku nije definisana nekom prethodnom odlukom niti implicirana. Ovu heuristiku ćemo u daljem tekstu označavati VS_{random}. Ova heuristika se može uopštiti tako što bi se dozvolilo da se u određenom broju biranja izbor promenljive vrši u skladu sa heuristikom VS_{random}, a u ostalim, prema alternativnoj ponuđenoj heuristici. Za zadatu verovatnoću izbora p ($0 \leq p \leq 1$) u skladu sa slučajnom heuristikom, ovakvu heuristiku ćemo dalje označavati sa $\mathsf{VS}^{p}_{random} \circ \mathsf{VS}_{x}$. Iako zanimljiva u kontekstu teorijske analize rada SAT rešavača, heuristika slučajnog izbora se u praksi pokazuje vrlo loše i stoga se ne koristi. S druge strane, korišćenje heuristike $\mathsf{VS}^{p}_{random} \circ \mathsf{VS}_{x}$ za vrlo malo p može biti uspešnije od korišćenja same alternativne politike.

Nešto složenija i, ujedno, najpopularnija heuristika izbora promenljive je heuristika VSIDS. Ona koristi dinamičko rangiranje promenljivih prema njihovom učešću u skorašnjim konfliktima. Za svaku promenljivu se čuva *faktor aktivnosti*. Prilikom svakog konflikta, svim promenljivama koje učestvuju u njemu povećava se faktor aktivnosti za neku vrednost `bumpAmount`. Prilikom konflikta, `bumpAmount` se množi koeficijentom `decayFactor` koji je veći od 1. Na taj način, promenljive koje su učestvovale u skorijim konfliktima imaju veće faktore aktivnosti. S vremena na vreme, svi faktori aktivnosti se skaliraju kako ne bi došlo do prekoračenja. Problem sa heuristikom VSIDS je što na početku rešavanja ni jedna promenljiva nema prednost nad nekom drugom. Jednostavna modifikacija koja se u praksi pokazala kao korisna je da se, na početku rešavanja, faktor aktivnosti svake promenljive inicijalizuje na broj njenih pojavljivanja u formuli. Politika je opisana u pseudokodu koji sledi, a u nastavku teksta ćemo je označavati sa $\mathsf{VS}_{VSIDS}^{b,d,init}$ gde je b skraćenica za `bumpAmount`, d za `decayFactor`, a *init* označava da li se vrši inicijalizacija polaznih aktivnosti.

```
function selectVariable () : variable
begin
    select v such that activity[v] = max(activity)
    return v
end

function onConflict (c: clause) : void
begin
    foreach v in c
        bumpVariableActivity(v)
end

function varDecayActivity () : void
begin
    bumpAmount := bumpAmount * decayFactor
end

function bumpVariableActivity (v : variable) : void
begin
    activity[v] := activity[v] + bumpAmount
    if activity[v] > MAX_ACTIVITY then
        rescaleVariableActivities()
end
```

```
function rescaleVariableActivities () : void
begin
    foreach v
        activity[v] := activity[v] / MAX_ACTIVITY
    bumpAmount := bumpAmount / MAX_ACTIVITY
end
```

Heuristike izbora polariteta promenljive

Kada je izabrana promenljiva u vezi sa čijom vrednošću će biti doneta odluka, potrebno je izabrati i njen polaritet, tj. odlučiti da li će joj prilikom odluke biti prvo dodeljivana vrednost 0 ili 1. Najjednostavnije su heuristike konstantnog izbora. Pod $\mathsf{PS}_{positive}$ će se podrazumevati heuristika koja uvek bira pozitivan polaritet za promenljivu, dok će $\mathsf{PS}_{negative}$ označavati heuristiku koja uvek bira negativan polaritet. Heuristika koja se bazira na slučajnom izboru, tako da se sa verovatnoćom p odlučuje za pozitivan polaritet, a inače za negativan, označavaće se sa PS^{p}_{random}.

Nešto složenija i često korišćenja heuristika *čuvanja polariteta*, $\mathsf{PS}^{init}_{polarity\_caching}$, koja se u praksi pokazala vrlo uspešnom, bira polaritet promenljive koji je za nju bio poslednji put izabran u odluci ili impliciran. Za polazni polaritet se u izvornoj verziji politike bira negativan. Kako na ovaj način postoji mogućnost da se negativan polaritet predugo zadrži za veliki broj promenljivih i da tako ova politika bude previše slična politici $\mathsf{PS}_{negative}$, ona se često modifikuje tako da se početni polaritet promenljive inicijalizuje na pozitivan ako ona u formuli češće učestvuje bez negacije, a na negativan u suprotnom. Ova politika će se označavati sa $\mathsf{PS}^{init}_{polarity\_caching}$.

Heuristike otpočinjanja iznova

Najjednostavnija heuristika otpočinjanja iznova je da se ono ne koristi. Ona će se označavati sa $\mathsf{R}_{no\_restart}$. Druge heuristike koje će biti opisane se zasnivaju na brojanju konflikata do sledećeg otpočinjanja iznova.

Heuristika koju koristi `MiniSAT` zahteva da se kao parametar zada polazni broj konflikata do otpočinjanja iznova `numConflictsForFirstRestart`. Posle svakog otpočinjanja iznova, broj potrebnih konflikata se povećava množenjem sa drugim parametrom `restartInc` koji treba da bude veći od 1. Za ovu politiku buće korišćena oznaka $\mathsf{R}^{c_0,q}_{minisat}$ gde je c_0 skraćenica za `numConflictsForFirstRestart`, a q za `restartInc`. Ova heuristika je opisana pseudokodom u nastavku.

```
function init () : void
begin
    numConflictsForNextRestart := numConflictsForFirstRestart
```

```
end

function shouldRestart () : bool
begin
    if numConflicts >= numConflictsForNextRestart then return true
    else return false
end

function onRestart () : void
begin
    numConflicts := 0
    numConflictsForNextRestart = numConflictsForNextRestart * restartInc
end

function onConflict () : void
begin
    numConflicts = numConflicts + 1
end
```

Zbog eksponencijalnog rasta broja potrebnih konflikata posle kojeg dolazi do otpočinjanja iznova pri ovoj politici, ono posle određenog vremena praktično nestaje. Zbog toga su formulisane dve heuristike brzog otpočinjanja iznova. Lubijeva heuristika [61], vrši i-to otpočinjanje iznova posle $t_i \cdot$ `sizeFactor` konflikata od prethodnog. Pri tome je `sizeFactor` parametar politike, dok je niz t_i definisan na sledeći način:

$$t_i = \begin{cases} 2^{k-1} & ako\ je\ i = 2^k - 1 \\ t_{i-2^{k-1}+1} & ako\ je\ 2^{k-1} \leq i \leq 2^k - 1 \end{cases}$$

Prvih nekoliko elemenata niza t_i su:

$$1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1, \ldots$$

Za ovu heuristiku će biti korišćena oznaka R^m_{luby} gde m skraćenica za `sizeFactor`.

Heuristika korišćena u rešavaču `PicoSAT` [48], kao parametre traži početni broj konflikata do otpočinjanja iznova `numConflictsForFirstRestart` i faktor kojim se ovaj broj množi — `restartInc`. Množenje se vrši posle svakog otpočinjanja iznova, sve dok broj potrebnih konflikata ne pređe određeno ograničenje. U tom trenutku se

to ograničenje množi sa `restartInc`, a broj potrebnih konflikata se ponovo postavlja na `numConflictsForFirstRestart`. Ova politika će biti označena sa $\mathsf{R}_{picosat}^{c_0,q}$ gde je c_0 skraćenica za `numConflictsForFirstRestart`, a q za `restartInc`. U nastavku je dat odgovarajući pseudokod.

```
function init () : void
begin
    inner := numConflictsForFirstRestart
    outer := numConflictsForFirstRestart
end

function shouldRestart () : bool
begin
    if numConflicts >= numConflictsForNextRestart then return true
    else return false
end

function onRestart () : void
begin
    numConflicts := 0;
    if inner >= outer then
    begin
        outer := outer * restartInc
        inner := numConflictsForFirstRestart
    end
    else inner = inner * restartInc
    numConflictsForNextRestart := inner
end

function onConflict () : void
begin
    numConflicts := numConflicts + 1
end
```

Heuristika izbora klauza za zaboravljanje

U trenutku kada treba vršiti zaboravljanje, potrebno je imati način izbora klauza koje će biti zaboravljene. Kod `MiniSAT` rešavača, klauze se sortiraju prema učestalosti njihove upotrebe u analizama konflikta. Zaboravlja se određeni procenat, `percentToForget`, ukupnog broja klauza koje smeju biti zaboravljene. Klauze

koje ne smeju biti zaboravljene su one koje su uzrokovale dodelu vrednosti trenutno impliciranih promenljivih. Njih nazivamo *zaključanim* (eng. locked).

Aktivnost klauza se računa na isti način kao aktivnost promenljivih kod politike $\mathsf{VS}^{b,d,init}_{VSIDS}$. Stoga u pseudokodu koji sledi računanje aktivnosti klauza nije prikazano.

```
function onForget (learntClauses : clause[]) : void
begin
    unlockedClauses := removeLockedClauses(learntClauses)
    sortAscendingOnActivity(unlockedClauses)
    firstToForget := length(ulockedClauses) * (1 - percentToForget)
    for i := firstToForget to length(unlockedClauses)
        delete(unlockedClauses[i])
end
```

Heuristika izbora trenutka zaboravljanja

`MiniSAT` koristi heuristiku kod koje se zaboravljanje vrši pošto se nauči određeni broj klauza. Ovaj broj se povećava za faktor `forgetInc` posle svakog otpočinjanja iznova. U nastavku je dat odgovarajući pseudokod.

```
function init () : void
begin
    numClausesForNextForget = numInitialClauses / 3
end

function shouldForget () : bool
begin
    if numLearntClauses >= numClausesForNextForget then return true
    else return false
end

function onConflict () : void
begin
    numClausesForNextForget = numClausesForNextForget * forgetInc
end

function onForget () : void
begin
    numLearntClauses := 0
end
```

2.5 Ispravnost SAT rešavača

Moderni SAT rešavači predstavljaju složene softverske sisteme. Njihove implementacije zasnovane na CDCL sistemu pretrage uključuju netrivijalne algoritme i tehnike na implementacionom nivou. Zbog toga je teško ili čak nemoguće uveriti se da ne sadrže greške bez njihove formalne analize. Opisi novih procedura se obično diskutuju vezujući se za konkretne implementacije i njihove tehničke detalje, dok su dokazi njihovih svojstava poput ispravnosti obično dati u vrlo neformalnom obliku. S obzirom na njihovu primenu u aplikacijama poput verifikacije softvera i hardvera u kojima je ispravnost od kritičnog značaja, dokazivanje njihvove ispravnosti predstavlja važan problem u naučnoj oblasti rešavanja SAT problema. Prve korake u teorijskoj analizi SAT rešavača predstavlja formulisanje apstraktnih sistema pravila koji ih formalno opisuju. Jedan takav sistem je prikazan u potpoglavlju 2.3. Na osnovu tih apstraktnih sistema, urađeni su prvi dokazi ispravnosti SAT rešavača [72, 53]. Kako su ovi dokazi bili standardni dokazi ,,na papiru", sadržali su nepreciznosti i nedorečenosti. Dodatno, nisu sadržali dokaze ispravnosti algoritama i složenih struktura podataka koje se koriste na konkretnom nivou — u implementacijama SAT rešavača. Alternativa standardnim dokazima izvedenim ,,na papiru" predstavljaju formalni dokazi proverivi od strane specijalizovanog softvera — interaktivnih dokazivača (eng. proof assistant) čiji su glavni predstavnici sistemi Isabelle [78] i Coq [25].

Formalizacija modernih SAT rešavača je urađena u sistemu Isabelle [65, 66, 67, 68]. Formalizacija je urađena na nekoliko nivoa. Dokazana je korektnost osnovne DPLL procedure [67], potom korektnost apstraktnog sistema pravila za SAT [68], zatim korektnost funkcionalne implementacije SAT rešavača [66] i korektnost imperativne implementacije u okviru horove logike [65]. Formalizacija se oslanja na apstraktni sistem pravila Krstića i Goela [53]. Dokaz saglasnosti SAT rešavača oslanja se na nekoliko uočenih invarijanti apstraktnih pravila sistema. Dokaz zaustavljanja se zasniva na zapažanju da je među listama literala M koje indukuju parcijalne valuacije za ulaznu formulu, moguće definisati parcijalno uređenje $\succ$ takvo da važi $M_1 \succ M_2$ ukoliko je M_2 lista literala u stanju koje je dobijeno primenom pravila na stanje kom pripada lista M_1. Dokaz zaustavljanja se zasniva na svojstvima ovog uređenja i činjenici da postoji konačan broj listi literala koje zadovoljavaju invarijante pravila. Težina dokaza zaustavljanja potiče pre svega iz prisustva pravila povratnog skoka. Takođe se dokazuje da sva stanja u kojima se pravila ne mogu primeniti, moraju biti ili prihvatajuća ili odbacujuća. Ova tri rezultata čine ispravnost SAT rešavača.

Ideja uvođenja uređenja nad listama literala biće klučna i u ovom radu u po-

glavlju 4.4 prilikom dokazivanja svojstava CDCL sistema za koherentnu logiku. Takođe, ispostaviće se da su i neka od drugih svojstava CDCL sistema za SAT rešavače koja figurišu u dokazu zaustavljanja, takođe korisne i u dokazima svojstava CDCL sistema za koherentnu logiku.

Glava 3

Prilagodljivo rešavanje SAT problema

Ova glava se bavi pristupima usmeravanja pretrage u rešavanju SAT problema zasnovanim na izboru rešavača, heuristika koje oni koriste i njihovih parametara, a na osnovu svojstava konkretne instance problema koju je potrebno rešiti. Ovaj izbor se vrši jednom — pre početka pretrage. Umesto razdvajanja i posebnog razmatranja izbora SAT rešavača, heuristika ili njihovih parametara, može se govoriti opšte, u terminima izbora algoritma. Ovo je opravdano utoliko pre što se za sve pomenute zadatke mogu koristiti iste ili vrlo slične tehnike.

Izvor potrebe za prilagodljivim rešavanjem SAT problema je varijacija vremena koju algoritmi pokazuju na različitim SAT instancama, zahvaljujući čemu nijedan algoritam nije pogodan za rešavanje svih SAT instanci. Sistemi koji implementiraju tehnike za ovakvo prilagodljivo rešavanje problema nazivaju se *algoritamskim portfolijima* i vrlo su korisni u rešavanju SAT problema. Iskustvo sa takmičenja u rešavanju SAT problema (eng. SAT Competitions[1]), na kojem učestvuju najbolji sistemi za rešavanje SAT problema, pokazuje da algoritamski portfoliji za SAT ne predstavljaju više samo korisnu nadogradnju SAT rešavača, već su sazreli da budu primarni izbor u praktičnom rešavanju SAT problema. Metode izbora algoritma koje će biti predložene u ovoj glavi su značajno jednostavnije od ranije korišćenih metoda, a u praksi se pokazuju superiornim u odnosu na njih. Ove metode počivaju na sintaksnoj sličnosti instanci kojima odgovaraju isti algoritmi. Analiza sintaksne sličnosti instanci urađena je pomoću dve predložene metode, od kojih jedna počiva na sličnosti grafova, a druga na rastojanjima između vektora svojstava koja opisuju instance.

Pored varijacije vremena rešavanja koju na istom problemu pokazuju različiti

[1] `http://www.satcompetition.org/`

algoritmi, postoji i značajna varijacija vremena pri ponavljanju rešavanja jedne instance istim algoritmom pri malim promenama u zapisu problema, poput zamene redosleda klauza ili literala u klauzi. Zbog ovog fenomena, za temeljno poređenje SAT rešavača potrebne su naprednije tehnike od prostog merenja vremena rešavanja, opisane u narednim poglavljima.

3.1 Osnove prilagodljivog rešavanja SAT problema

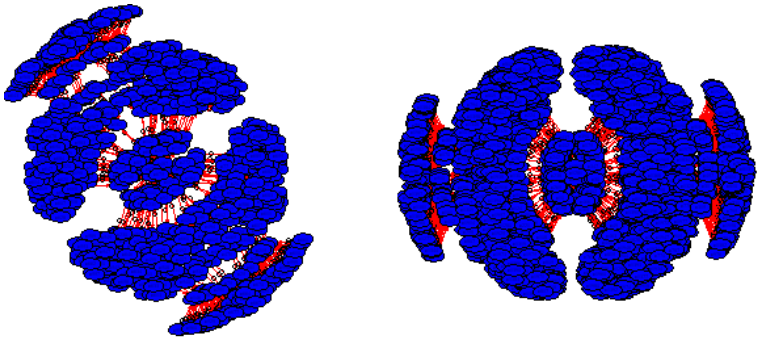
Pristup prilagodljivom rešavanju SAT problema koji će biti prikazan, suštinski se temelji na pojmu sličnosti SAT instanci. Sličnost koja je od značaja je prvo sintaksna sličnost, a potom i sličnost algoritama koji su adekvatni za rešavanje datih instanci.

Iskazne formule čiju je zadovoljivost potrebno proveriti često dolaze iz primena u kojima se iskazna logika koristi za modelovanje stvarnih problema, uređaja i slično. Zahvaljujući tome, one se mogu podeliti u *familije* formula istog porekla. Opravdano je pretpostaviti da sličnosti u njihovom poreklu impliciraju i sličnost algoritama koje treba primeniti prilikom njihovog rešavanja kako bi se dobilo na brzini. Stoga bi za SAT rešavanje bilo korisno da za različite familije formula budu poznati algoritmi koji se ponašaju u proseku najbolje na njima, a da se algoritam koji treba primeniti na nepoznatoj instanci bira na osnovu sličnosti sa već poznatim instancama.

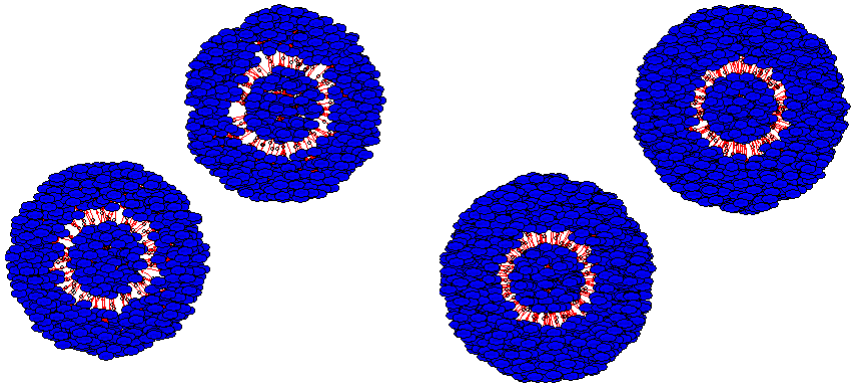
Formule koje dolaze iz primena, a pripadaju različitim familijama imaju različite i, u pogodnoj grafovskoj reprezentaciji, vizuelno prepoznatljive strukture. Postoji veći broj načina reprezentovanja formula grafovima. Takođe, postoji i veći broj načina vizualizovanja grafova. Stoga ovo zapažanje predstavlja samo neformalnu motivaciju za dalje razmatranje, a ne formalan argument. Ipak, vizualizacija grafovske strukture formule je tema kojoj je u literaturi već poklonjena pažnja [91]. Pretpostavka na kojoj počivaju metode koje će biti izložene je da se ta struktura može na neki način automatski prepoznati i da se na osnovu nje može postići visok nivo preciznosti klasifikacije. Kao ilustraciju prepoznatljivosti prikaza grafovskih struktura koje odgovaraju formulama, navodimo nekoliko primera. Razmatrane formule pripadaju korpusu sastavljenom za takmičenje SAT rešavača, SAT Competition 2002, a odgovarajući grafovi su vizuelizovani pomoću paketa `GRAPHVIZ`[2]. Na slikama 3.1, 3.2 i 3.3 su prikazane grafovske reprezentacije po dve formule iz tri familije. Neformalno govoreći, očigledno je da prikazi grafova koji predstavljaju formule koje pripadaju istoj familiji imaju sličan oblik, iako imaju različit broj čvorova.

Postojanje sličnosti između formula iz iste familije se može demonstrirati metodama istraživanja podataka — tako što bi se izvršila automatska klasifikacija formula

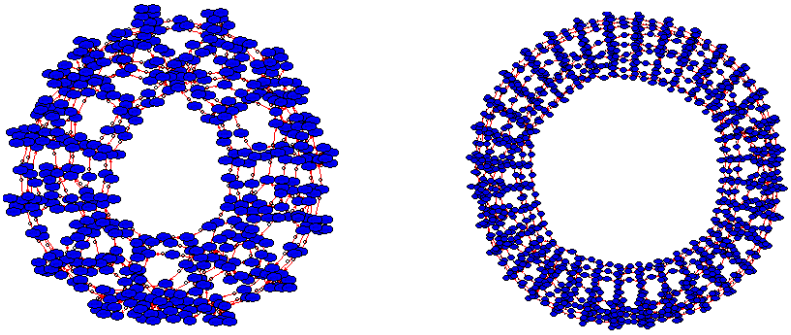
[2] `http://www.graphviz.org/`



Slika 3.1: Prikaz formula bart10.shuffled.cnf sa 144 promenljive i 560 klauza i bart20.shuffled.cnf sa 270 promenljivih i 1476 klauza. Obe formule potiču iz problema verifikacije hardvera.



Slika 3.2: Prikaz formula homer06.shuffled.cnf sa 180 promenljivih i 830 klauza i homer09.shuffled.cnf sa 270 promenljivih i 1920 klauza. Obe formule potiču iz problema verifikacije hardvera.



Slika 3.3: Prikaz formula rope_0003.shuffled.cnf sa 108 promenljivih i 252 klauze i rope_0008.shuffled.cnf sa 288 promenljivih i 672 klauze. Obe formule potiču iz problema bojenja grafova.

u familije i proverio njen kvalitet.

3.1.1 Problem klasifikacije i sličnost SAT instanci

Klasifikacija je jedan od najčešćih problema u istraživanju podataka. Neka postoji konačan broj unapred određenih disjunktnih klasa $\mathcal{C}_i$ čija je unija ceo skup instanci. Problem se sastoji u tome da se za novu, nepoznatu instancu pronađe jedna klasa kojoj ova instanca pripada. Formalno, ako je X skup svih instanci i ako je dat skup $\{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\}$ gde je $x_i \in X$, a $c_i \in \mathbb{N}$ predstavlja redni broj klase kojoj pripada i-ta instanca, onda je potrebno naći funkciju $h : X \to \mathbb{N}$ koja preslikava skup instanci X u skup rednih brojeva klasa kojima te instance pripadaju.

Problem klasifikacije se može javiti u različitim praktičnim situacijama: prepoznavanje relevantnih priloga na Internet grupama [56], prepoznavanje rukom pisanog teksta [59], prepoznavanje autora dokumenata [51] i slično.

Problem klasifikacije se često rešava tehnikama mašinskog učenja. Učenje se vrši na osnovu podataka koje nazivamo *podacima za trening*, a njihov skup *trening skupom*. Rezultat procesa učenja nazivamo *modelom*. Naučeni model se kasnije na neki način primenjuje na nepoznate podatke kako bi se došlo do zaključaka o njima. Pre upotrebe uobičajeno je da se model testira i evaluira. U tu svrhu se koristi odvojen skup *podataka za testiranje — test skup* na kojima se ocenjuje unapred izabrana mera kvaliteta kao što je procenat pravilno klasifikovanih instanci.

Ako se odredi neka funkcija rastojanja (ili sličnosti) nad instancama koje treba klasifikovati, za rešavanje problema klasifikacije može se koristiti algoritam *k najbližih suseda* koji novu instancu dodeljuje klasi koja se najčešće javlja među k najbližih instanci za trening u smislu usvojene distance (ili sličnosti) [27].

Standardna mera kvaliteta klasifikacije je preciznost klasifikacije — broj tačno klasifikovanih instanci podeljen ukupnim brojem instanci.

Kako bi se ispitala teza o sličnosti iskaznih formula iz iste familije potrebno je proveriti preciznost klasifikacije iskaznih formula u familije. Za to je potrebno definisanje mera sličnosti nad iskaznih formulama. U nastavku su predložena dva načina merenja sličnosti.

- Prvi pristup merenju sličnost SAT instanci se zasniva na meri sličnosti nad grafovima, pošto je prva indikacija sličnosti upravo vizuelno ustanovljena sličnost grafova.

- Drugi pristup se zasniva na predstavljanju iskaznih formula vektorima numeričkih atributa i korišćenju mera rastojanja nad tim vektorima.

3.1.2 Sličnost grafova i klasifikacija zasnovana na njoj

Sličnost grafova je od centralnog značaja za razne tehnike analize podataka i postoji veći broj takvih metoda [52, 38, 14, 101]. Ove metode su uspešno primenjene u više domena poput rangiranja rezultata upita prilikom pretrage Interneta [52], izdvajanja sinonima [14], uparivanja struktura baza podataka [70], konstrukcije filogenetskih stabala [38], analize socijalnih mreža [58] i slično. Ove tehnike se često zasnivaju na iterativnom računanju sličnosti čvorova grafova i izvođenju ukupne mere sličnosti za cele grafove iz sličnosti čvorova. Međutim, poznatim metodama nedostaju određena poželjna svojstva koja su potrebna za analizu sličnosti iskaznih formula. Stoga je potrebno profinjenje pojma sličnosti grafova koje vodi do nove metode za merenje sličnosti grafova i njihovih čvorova [75]. Za ovu novu metodu, dokazana je konvergencija ali i sledeća poželjna svojstva, od kojih svakoj postojećoj meri sličnosti nedostaje bar jedno:

- Pri računanju sličnosti grafa sa samim sobom, svaki čvor bi trebalo da bude najsličniji samom sebi.
- Vrednosti mere sličnosti čvorova treba da budu iz fiksiranog intervala, pri čemu sličnost čvora sa samim sobom treba uvek da bude maksimalna vrednost iz intervala.
- Vrednosti mere sličnosti čvorova ne treba samo da odražavaju relativnu sličnost parova čvorova (u smislu da je jedan par sličniji od drugog), već da odražavaju sličnost dva čvora i nezavisno od sličnosti drugih čvorova (ovo svojstvo je posebno važno za konstrukciju mere sličnosti celih grafova).
- Sličnost čvorova treba prepoznati i u slučaju nedostatka grana koje vode ka tom čvoru ili od njega.

Sličnost čvorova grafova. U postojećim metodama, računanje sličnosti x_{ij} čvora i iz grafa A i čvora j iz grafa B se bazira na sabiranju ili uprosečavanju sličnosti svih parova čvorova koji su susedni čvoru i u grafu A i čvoru j u grafu B. Predložena metoda, koja će biti nazvana *metoda uparivanja suseda*, se bazira na principu koji se može ilustrovati sledećim primerom. Dve ljudske ruke deluju slično ne zato što su svi prsti jedne ruke slični svim prstima druge ruke, već zato što svakom prstu jedne ruke odgovara jedan prst druge koji mu je vrlo sličan. Analogno, koncept sličnosti se može korigovati — *dva čvora i i j se mogu smatrati sličnim ukoliko se susedi čvora i mogu upariti sa susedima čvora j koji su im vrlo slični.*

Ako je (i, j) grana u grafu, čvor i se naziva *ulazni sused* čvora j, a čvor j se naziva *izlazni sused* čvora i. *Ulazni stepen* $us(i)$ čvora i je broj ulaznih suseda čvora i, a *izlazni stepen* $is(i)$ čvora i je broj izlaznih suseda čvora i. Ako su A i B dva konačna skupa, *uparivanje* elemenata ovih skupova je skup parova $M = \{(i, j)|i \in A, j \in B\}$ takvih da ni jedan element jednog skupa nije uparen sa više od jednog elementa drugog skupa. Za uparivanje M, *funkcije nabrajanja* $f : \{1, 2, \ldots k\} \to A$ i $g : \{1, 2, \ldots k\} \to B$ se definišu tako da važi $M = \{(f(l), g(l)) \mid l = 1, 2, \ldots, k\}$ gde važi $k = |M|$. Ako je $w(a, b)$ funkcija koja dodeljuje težine parovima elemenata $a \in A$ i $b \in B$, *težina uparivanja* je suma težina dodeljenih parovima elemenata tog uparivanja. *Problem dodele* je problem nalaženja uparivanja elemenata skupova A i B najveće težine, koje nazivamo *optimalnim*. Problem dodele se obično rešava poznatim Mađarskim algoritmom složenosti $O(mn^2)$ gde je $m = \max(|A|, |B|)$ i $n = \min(|A|, |B|)$ [54].

Računanje sličnosti x_{ij} čvorova i i j se zasniva na iterativnoj proceduri datoje sledećim jednačinama:

$$x_{ij}^{k+1} \leftarrow \frac{s_{ul}^{k+1}(i, j) + s_{iz}^{k+1}(i, j)}{2}$$

gde je

$$s_{ul}^{k+1}(i, j) \leftarrow \frac{1}{m_{ul}} \sum_{l=1}^{n_{ul}} x_{f_{ij}^{ul}(l) g_{ij}^{ul}(l)}^{k} \qquad s_{iz}^{k+1}(i, j) \leftarrow \frac{1}{m_{iz}} \sum_{l=1}^{n_{iz}} x_{f_{ij}^{iz}(l) g_{ij}^{iz}(l)}^{k} \tag{3.1}$$

$$m_{ul} = \max(us(i), us(j)) \qquad m_{iz} = \max(is(i), is(j))$$

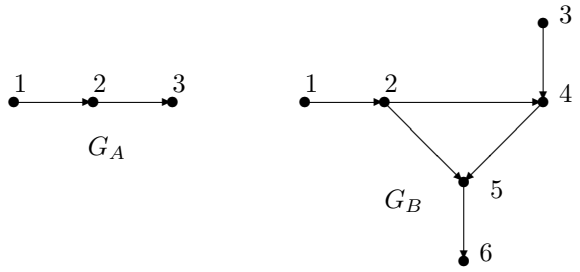
$$n_{ul} = \min(us(i), us(j)) \qquad n_{iz} = \min(is(i), is(j))$$

pri čemu su funkcije f_{ij}^{ul} i g_{ij}^{ul} funkcije nabrajanja za optimalno uparivanje ulaznih suseda čvorova i i j u odnosu na funkciju težine $w(a, b) = x_{ab}^{k}$, i analogno za f_{ij}^{iz} i g_{ij}^{iz}. U jednačinama (3.1), $\frac{0}{0}$ se definiše kao 1. Polazne vrednosti sličnosti x_{ij}^{0} su 1 za svako i i j. Kriterijum zaustavljanja je $\max_{ij} |x_{ij}^{k} - x_{ij}^{k-1}| < \varepsilon$ za neku odabranu preciznost ε.

Primer 1 *Zarad ilustracije metoda, metod je primenjen na dva grafa (prikazana na slici 3.4). Vrednosti mere sličnosti njihovih čvorova su date u tabeli 3.1. U datoj tabeli može se primetiti značajna sličnost čvorova* 1_A *sa* 1_B *i* 3_B *pošto oni imaju samo po jednu izlaznu granu* 3_A *sa* 6_B *pošto oni imaju samo po jednu ulaznu granu. Čvor* 2_A *je sličan ,,unutrašnjim" čvorovima* 2_B,4_B *i* 5_B.

Predloženi metod konvergira, kao što tvrdi sledeća teorema.

Teorema 1 *Za bilo koja dva grafa* A *i* B*, i za bilo koji par čvorova* $i \in A$ *i* $j \in B$*, postoji granična vrednost* $x_{ij} = \lim_{k \to \infty} x_{ij}^{k}$ *sa vrednošću u intervalu* $[0, 1]$.



Slika 3.4: Primer dva grafa.

Tabela 3.1: Sličnosti čvorova grafova datih na slici 3.4, izračunate korišćenjem metode uparivanja suseda za $\varepsilon = 10^{-4}$.

	1_B	2_B	3_B	4_B	5_B	6_B
1_A	0.682	0.100	0.597	0.200	0.000	0.000
2_A	0.000	0.364	0.045	0.195	0.400	0.000
3_A	0.000	0.000	0.000	0.091	0.091	0.700

Dokaz. Za svako $i \in A$ i $j \in B$, niz $(x_{ij}^k)_{k=0}^{\infty}$ je neopadajući. To će biti dokazano indukcijom po k.

Polazna vrednost sličnosti x_{ij}^0 je jednaka 1, po definiciji. U prvoj iteraciji, težina optimalnog uparivanja ulaznih, odnosno izlaznih čvorova je jednaka n_{ul}, odnosno n_{iz}, pošto je težina uparivanja bilo koja dva čvora 1. Pošto važi $m_{ul} \geq n_{ul}$ i $m_{iz} \geq n_{iz}$, važi $s_{ul}^1(i,j) = \frac{n_{ul}}{m_{ul}} \leq 1$ i $s_{iz}^1(i,j) = \frac{n_{iz}}{m_{iz}} \leq 1$. Isto važi i za prosek ove dve vrednosti koji je po definiciji x_{ij}^1. Ovim je dokazano da u prvom koraku vrednost sličnosti ne može da poraste što je baza indukcije.

Pretpostavimo da je do iteracije k niz vrednosti x_{ij}^k neopadajući, što znači da važi $x_{ij}^k \leq x_{ij}^{k-1}$. Ovo znači da težine uparivanja (što su upravo x_{ij}^k) bilo koja dva čvora prilikom računanja s_{ul}^{k+1} i s_{iz}^{k+1} nisu veće nego težine (x_{ij}^k) prilikom računanja s_{ul}^k i s_{iz}^k, i stoga važi $s_{ul}^{k+1} \leq s_{ul}^k$ i $s_{iz}^{k+1} \leq s_{iz}^k$. Ovo će biti dokazano u slučaju sličnosti s_{ul}^{k+1} ulaznih čvorova, a rezonovanje za sličnost izlaznih čvorova s_{iz}^{k+1} je analogno. Neka su f_{ij}^k i g_{ij}^k funkcije nabrajanja optimalnog uparivanja ulaznih suseda čvorova $i \in A$ i $j \in B$ u iteraciji k. Tada važi

$$\sum_{l=1}^{n_{ul}} x^{k+1}_{f_{ij}^{k+1}(l) g_{ij}^{k+1}(l)} \leq \sum_{l=1}^{n_{ul}} x^{k}_{f_{ij}^{k+1}(l) g_{ij}^{k+1}(l)} \leq \sum_{l=1}^{n_{ul}} x^{k}_{f_{ij}^{k}(l) g_{ij}^{k}(l)}$$

Prva nejednakost važi na osnovu induktivne pretpostavke, a druga iz optimalnosti uparivanja definisanog funkcijama f_{ij}^k i g_{ij}^k, u iteraciji k. Deljenjem svih izraza sa m_{ul}, zaključuje se da važi $s_{ul}^{k+1}(i,j) \leq s_{ul}^k(i,j)$. Isto važi i za sličnosti izlaznih čvorova. Stoga važi $x_{ij}^{k+1} \leq x_{ij}^k$. Ovim je dokazan korak indukcije. Stoga je niz vrednosti sličnosti $(x_{ij}^k)_{k=0}^{\infty}$ neopadajući.

Indukcijom po k će biti dokazano da su u svim iteracijama vrednosti sličnosti nenegativne. U prvoj iteraciji vrednosti sličnosti su nenegativne po definciji. U svakoj sledećoj iteraciji, vrednosti se ažuriraju uprosečavanjem vrednosti iz prethodne iteracije. Uprosečavanjem nenegativnih vrednosti dobijaju se nenegativne vrednosti. Stoga su vrednosti sličnosti ograničene odozdo nulom. Nerastući niz ograničen odozdo mora imati graničnu vrednost, tako da $x_{ij} = \lim_{k\to\infty} x_{ij}^k$ postoji. Pošto je niz nerastući i važi $x_{ij}^0 = 1$, granična vrednost ne može biti veći od 1. Takođe, pošto su svi elementi nenegativni, granična vrednost mora biti nenegativna. Ovim je teorema dokazana. □

Jednostavnim primerima se može pokazati da je ova mera sličnosti može uzeti i maksimalnu i minimalnu vrednost iz intervala $[0, 1]$.

Važno svojstvo sličnosti za izomorfne grafove, dato je sledećom teoremom.

Teorema 2 *Neka su A i B dva izomorfna grafa. Neka je $f : A \to B$ izomorfizam ova dva grafa. Za svaki čvor $i \in A$, važi $x_{i,f(i)} = 1$.*

Dokaz. Indukcijom po k se može dokazati da važi $x_{i,f(i)}^k = 1$ za sve $i \in A$ i sve $k \geq 0$.

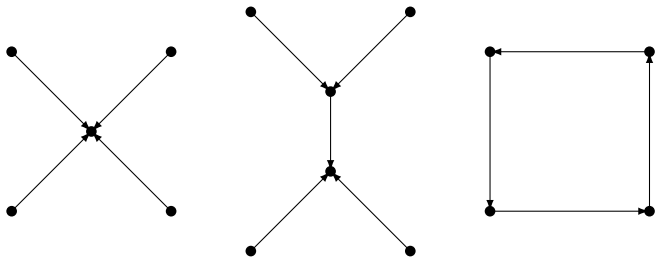
Po definiciji, polazna vrednost $x_{i,f(i)}^0$ je jednaka 1 za sve $i \in A$. Ovim je dokazana baza indukcije. Neka je $k > 0$. Neka važi $x_{i,f(i)}^k = 1$ za sve $i \in A$. Pošto je f izomorfizam dva grafa, čvorovi i i $f(i)$ moraju imati isti broj ulaznih i izlaznih suseda. Stoga važi $m_{ul} = n_{ul}$ i $m_{iz} = n_{iz}$. Stoga je dovoljno dokazati da su težine optimalnih uparivanja za ulazne i izlazne sličnosti jednake n_{ul} i n_{iz}. Prvo će biti razmotrena sličnost ulaznih čvorova. Pošto je f izomorfizam, f preslikava sve ulazne susede čvora i u ulazne susede čvora $f(i)$. Težine $x_{af(a)}$ uparivanja svakog ulaznog suseda a čvora i sa ulaznim susedom $f(a)$ čvora $f(i)$ su jednake 1 na osnovu induktivne hipoteze, što je pri tome maksimalna vrednost. Stoga, uparivanje ulaznog suseda a čvora i sa ulaznim suedom $f(a)$ čvora $f(i)$ je optimalno. Pošto postoji n_{ul} ulaznih suseda, težina optimalnog uparivanja ulaznih suseda je n_{in}. Analognim zaključivanjem se može pokazati da je težina optimalnog uparivanja izlaznih suseda jednaka n_{out}. Stoga, i ulazna i izlazna sličnost čvora i sa čvorom $f(i)$ u koraku $k+1$ je 1 za sve $i \in A$, pa je stoga i njihov prosek $x_{i,f(i)}^{k+1}$ takođe jednak 1 za sve $i \in A$.

Pošto važi $x_{i,f(i)}^k = 1$ za sve $k \geq 0$, i $i \in A$, limes $x_{i,f(i)}$ je takođe jednak 1 za sve $i \in A$. □

U slučaju gde je $A = B$ i f trivijalni automorfizam $f(i) = i$, za sve $i \in A$, ova teorema ima jednostavnu posledicu.

Posledica 1 *Za svaki graf A i svaki čvor $i \in A$, važi $x_{ii} = 1$.*

Prema navedenim tvrđenjima, metoda uparivanja suseda poseduje prve dve od nabrojanih poželjnih osobina.



Slika 3.5: Primer tri grafa.

Kako se matrica sličnosti $[x_{ij}]$ ne normira na kraju iteracije kao kod drugih metoda, lako je naći primere grafova čiji svi čvorovi imaju sličnost 0 ili 1. U slučaju čvorova kojima nedostaju ulazne ili izlazne grane, njihova ulazna, odnosno izlazna, sličnost je 1. Može se zaključiti da metoda uparivanja suseda poseduje sva navedena poželjna svojstva.

Sličnost grafova. Sličnosti x_{ij} izračunate metodom uparivanja suseda mogu se upotrebiti za definisanje mere sličnosti dva grafa. Neka su f i g funkcije nabrajanja optimalnog uparivanja čvorova grafova A i B u odnosu na funkciju težine x_{ij} i neka važi $n = \min(|A|, |B|)$. Tada se sličnost grafova A i B može definisati na sledeći način

$$s(A, B) = \frac{1}{n} \sum_{l=1}^{n} x_{f(l)g(l)}. \tag{3.2}$$

Na osnovu teoreme 1, vrednost mere sličnosti s je u intervalu $[0, 1]$. Posledica teoreme 2 je da u slučaju da su A i B izomorfni grafovi, važi $s(A, B) = 1$. Naravno, i drugačije definicije sličnosti dva grafa su moguće.

Primer 2 *Na slici 3.5, prikazana su tri grafa. Važi* $s(A, B) = 0.74$, $s(A, C) \approx 0$ *i* $s(B, C) \approx 0$. *Razlog za ovakav odnos sličnosti parova grafova bi se mogao ilustrovati sledećom neformalnim razmatranjem. U prva dva grafa, protok duž grana se iz spoljnih čvorova koncentriše u centralni deo grafa, dok kod trećeg grafa postoji cikličan protok.*

Metoda uparivanja suseda se u evaluaciji na nekoliko problema pokazala superiornom u odnosu na druge metode [75].

Sličnost iskaznih formula. Iskazne formula se mogu predstaviti grafom čiji su čvorovi klauze i literali i u kojem postoji grana između klauze i literala ukoliko literal učestvuje u klauzi. To je takozvani graf *literala i klauza*. Formula se može predstaviti i grafom čiji su čvorovi promenljive i u kojem postoji grana između dve

promenljive ukoliko se one javljaju u istoj klauzi u iskaznoj formuli. Ovo je graf *promenljivih*.

Kako bi se proverila teza o sličnosti iskaznih formula iz iste familije, urađena je klasifikacija iskaznih formula u familije, korišćenjem metoda k najbližih suseda. Za merenje sličnosti korišćena je metoda uparivanja suseda. Korišćeno je 149 instanci iz 9 familija iz korpusa sa takmičenja SAT 2002.[3] Odabrane instance su one sa manjim brojem čvorova u grafu literala i klauza, kako bi računanje sličnosti bilo vremenski efikasno. Grafovi literala i klauza za većinu formula su imali do 1000 čvorova, ali su grafovi 25 formula bili veći (do 5280 čvorova). Klasifikacija je izvršena korišćenjem algoritma k najbližih suseda sa unakrsnom validacijom [27] — za svaku formulu F, algoritmom k najbližih suseda odabrana je klasa na osnovu preostalih formula.

Eksperimenti su sprovedeni na klaster računaru sa 32 Intel Xeon procesora radnog takta od 2GHz sa dva jezgra i sa 2GB RAM memorije po procesoru. Ukupno vreme eksperimenta, koje je uključivalo 11026 izračunavanja sličnosti, je 102 sata, a prosečno vreme računanja sličnosti je 33 sekunde. Najbolja preciznost klasifikacije je bila 93% za $k = 7$. Ovim je potvrđeno da iskazne formule iz iste familije imaju strukturu koja se može uspešno automatski prepoznati, ali je računanje sličnosti grafova koji odgovaraju formulama previše računski zahtevno da bi se koristilo u praksi.

3.1.3 Numerički atributi iskaznih formula i klasifikacija zasnovana na njima

`SATzilla`, algoritamski portfolio koji dominira prethodnim takmičenjima SAT rešavača[4], je najvažniji i najuspešniji algoritamski portfolio za SAT problem, sa zadivljujućim performansama [99, 98]. Među brojnim doprinosima sistema `SATzilla`, verovatno je najveći uticaj na dalji razvoj oblasti učinilo predlaganje skupa atributa kojima se mogu okarakterisati instace SAT problema. Ovaj skup atributa intenzivno je korišćen u mnogim drugim sistemima uključujući i one koji su predloženi u ovom radu. Ovi atributi su prvi put predloženi od strane Nudelmana i saradnika [79], ali su u kasnijim verzijama proširivani [98].

Na slici 3.6 su prikazani atributi SAT instanci koje koriste sve verzije sistema `SATzilla` [99]. One se mogu podeliti u dve kategorije. Prvu grupu čine čisto sintaksni atributi (1-33) koji mogu biti izračunati jednostavnim pregledom instance. Drugu grupu čine atributi koji se računaju na osnovu kratkih pokretanja algoritama pretrage nad ovim instancama (34-48). Ovi atributi su korišćeni od strane više al-

[3] Instance su dostupne na adresi `http://www.satcompetition.org`.

[4] `http://www.satcompetition.org/`

goritamskih portfolija za SAT problem [99, 73, 49, 62, 77]. U svim eksperimentima opisanim u nastavku će biti korišćeni samo sintaksni atributi.

Atributi vezani za veličinu instance:

1-3. *Broj klauza c*, *Broj promenljivih v*, *Odnos v/c*

Atributi vezani za graf literala i klauza:

4-8. *Statistike stepena čvorova literala*: prosek, koeficijent varijacije, minimum, maksimum i entropija.
9-13. *Statistike stepena čvorova klauza*: prosek, koeficijent varijacije, minimum, maksimum i entropija.

Atributi grafa promenljivih:

14-17. *Statistike stepena čvorova*: prosek, koeficijent varijacije, minimum i maksimum.

Atributi vezani za odnose:

18-20. *Odnos broja pozitivnih i negativnih literala u klauzama*: prosek, koeficijent varijacije i entropija.
21-25. *Odnos broja pozitivnih i negatvinih pojavljivanja promenljivih*: prosek, koeficijent varijacije, minimum, maksimum i entropija.
26-27. *Udeo klauza dužine dva i tri.*

Blizina Hornovoj formuli:

28. *Udeo Hornovih klauza*
29-33. *Broj pojavljivanja svake promenljive u Hornovim klauzama*: prosek, koeficijent varijacije, minimum, maksimum i entropija.

Atributi pretrage DPLL procedurom:

34-38. *Broj jediničnih propagacija*: računat na dubini pretrage 1,4,16,64 i 256.
39-40. *Ocena veličine prostora pretrage*: prosečna dubina konflikta, ocena logiratma broja čvorova u stablu pretrage.

Atributi lokalne pretrage:

41-44. *Broj koraka do najboljeg lokalnog minimuma po pokretanju*: prosek, medijana, deseti i devedeseti percentil za lokalni rešavač `SAPS`.
45. *Prosečno poboljšanje kvaliteta tekućeg rešenja u odnosu na najbolje rešenje po pokretanju rešavača* `SAPS`.
46-47. *Udeo poboljšanja dobijenog na osnovu prvog lokalnog minimuma*: prosek za rešavače `SAPS` i `GSAT`.
48. *Koeficijent varijacije broja nezadovoljenih klauza u svakom lokalnom minimumu pri pokretanju rešavača* `SAPS`.

Slika 3.6: Atributi korišćeni od strane sistema `SATzilla`.

Kako bi se vektori numeričkih atributa koristili za klasifikaciju algoritmom k

najbližih suseda, potrebno je definisati funkciju rastojanja nad njima. U ovom radu biće korišćeno nekoliko funkcija rastojanja koje su se pokazale uspešnim u drugim domenima [94].

$$d_1(x,y) = \sqrt{\sum_{x \in atributi} (x_i - y_i)^2} \tag{3.3}$$

$$d_2(x,y) = \sum_{x \in atributi} \left(\frac{x_i - y_i}{\sqrt{|x_i y_i|} + 1} \right)^2 \tag{3.4}$$

$$d_3(x,y) \sum_{x \in atributi} \frac{|x_i - y_i|}{\sqrt{|x_i y_i|} + 1} \tag{3.5}$$

$$d_4(x,y) = \sum_{x \in atributi} \left(\frac{x_i - y_i}{\sqrt{|x_i y_i|} + 10} \right)^2 \tag{3.6}$$

Za klasifikaciju je korišćen ceo korpus sa takmičenja SAT 2002, od 1964 formule. Eksperimenti su sprovedeni na klaster računaru sa 32 Intel Xeon procesora radnog takta od 2GHz sa dva jezgra i sa 2GB RAM memorije po procesoru. Prosečno vreme računanja atributa je bilo 0.39 sekundi, a najbolja preciznost izračunata unakrsnom validacijom postignuta je za $k = 1$ sa merom rastojanja d_3 i iznosila je 98,6%. Pristup klasifikovanju instanci pomoću numeričkih atributa daje izrazito visoku preciznost, a vreme računanja atributa i rastojanja je za praktične potrebe zanemarljivo i stoga se može koristiti kao osnova prilagodljivog rešavanja SAT problema.

3.2 Metode prilagodljivog rešavanje SAT problema

Kao što je rečeno, najefikasniji i najznačajniji algoritamski portfolio za SAT problem je `SATzilla` [99]. `SATzilla` je vrlo uspešan sistem, ali vrlo složen i nije ga lako razumeti, reimplementirati ili modifikovati. Ovo inspiriše na potragu za jednostavnijim principom za izgradnju algoritamskog portfolija. U ovom poglavlju biće opisana dva pristupa prilagodljivom izboru algoritma za rešavanje zadate SAT instance. Oba se zasnivaju na metodi k-najbližih suseda. Pri tome, oba su značajno jednostavniji od sistema `SATzilla`, a u eksperimentalnoj evaluaciji pokazuju bolje rezultate. Kako je `SATzilla` najefikasniji postojeći portfolio sistem, zaključuje se da su predstavljene tehnike bolje od postojećih pristupa.

Tehnika zasnovana na klasifikaciji. Iskazne formule koje nastaju zapisivanjem praktičnih problema (na primer, raspoređivanje časova, verifikacija različitih vrsta

```
function ArgoSmArT (
    T : (atributeVector,family)[],
    f : function (F : family) : algorithm,
    k : integer,
    i : instance
) : algorithm
begin
    v := computeAtributeVector(i)
    T' := nearestNeighbors(k,v,T,distance)
    F := mostFrequentFamily(T')
    return f(F)
end
```

Slika 3.7: Procedura za izbor algoritma zasnovana na klasifikaciji instanci u familije.

hardvera i slično) se mogu podeliti u familije na osnovu svog porekla, odnosno problema koji predstavljaju. Kao što je pokazano u potpoglavljima 3.1.3 i 3.1.2, instance koje pripadaju jednoj familiji su među sobom obično sintaksno sličnije nego instance koje pripadaju različitim familijama. Ove sličnosti se mogu iskoristiti kao osnova za izgradnju algoritamskog portfolija zasnovanog na klasifikaciji algoritmom k najbližih suseda. Prva predložena tehnika, zasnovana na klasifikaciji nepoznate instance u neku od poznatih familija, podrazumeva da je skup instanci unapred particionisan na familije i da je za svaku od tih familija poznato koji algoritam, iz nekog skupa unapred datih algoritma, u proseku najbrže rešava instance te familije. Nepoznata instanca koju je potrebno rešiti, se klasifikuje algoritmom k najbližih suseda u jednu od familija i rešava se algoritmom koji odgovara toj familiji. Precizniji opis je dat na slici 3.7. Na osnovu rezultata prikazanih u potpoglavlju 3.1.3, najbolja vrednost za parametar k je 1 i ta vrednost će biti korišćena u svim narednim eksperimenitma, tako da u njima k neće figurisati kao parametar tehnike. Izbori funkcija `computeAtributeVector` i `distance` mogu biti različiti i nisu definisani opisom ove opšte tehnike. Takođe način na koji će za svaku familiju biti odabran pogodan algoritam nije jedinstveno određen, ali postoji nekoliko metoda pomoću kojih je to moguće uraditi [43, 3]. Ova tehnika će se zvati `ArgoSmArT`.

Tehnika zasnovana na sličnosti pojedinačnih instanci. Druga tehnika je inspirisana zapažanjem da postojeći portfolio sistemi za SAT, pa i tehnika `ArgoSmArT`, grade svoje modele (regresioni modeli, grupisanje instanci, itd.) unapred bez obzira na karakteristike ulazne instance. Stoga je poželjno isprobati tehniku koja bi se oslanjala na lokalne modele specifične za instancu koju je potrebno rešiti. To je takođe moguće postići razmatranjem njenih suseda u prostoru zadatih atributa.

```
function ArgoSmArTkNN (
    A : algorithm[],
    T : (atributeVector,time)[],
    k : integer,
    i : instance
) : algorithm
begin
    v := computeAtributeVector(i)
    T' := nearestNeighbors(k,v,T,distance)
    foreach a in A
        c'[a] := cost(a,T')
    select A'[] such that a ∈ A' iff a ∈ A and c'[a] = min(c')
    foreach a in A'
        c''[a] := cost(a,T)
    select A''[] such that a ∈ A'' iff a ∈ A' and c''[a] = min(c'')
    select a such that a ∈ A''
    return a
end
```

Slika 3.8: Procedura zasnovana na lokalnom izboru algoritma.

Sledeća tehnika podrazumeva da je dat skup instanci za koje je poznato vreme izvršavanja svih algoritama (u okviru unapred određenog maksimalnog vremena za rešavanje) u algoritamskom portfoliju. Na osnovu ovih vremena, za svaki solver se može izračunati *cena* pokretanja na svakoj instaci (što je veće vreme pokretanja, veća je cena). Za ulaznu instancu se prvo nalazi k najbližih poznatih instanci, a onda se instanca rešava algoritmom koji na nađenim instanciama ima najmanju cenu. U slučaju da više algoritama ima jednaku cenu na tim instancama, onda se bira bilo koji od njih sa najmanjom cenom na celom skupu poznatih instanci. Ova procedura je preciznije opisana na slici 3.8. Funkcije `computeAtributeVector`, `distance` i `cost` se mogu izabrati na različite načine i nisu definisane opisom ove opšte tehnike. Ova tehnika će se zvati `ArgoSmArT k-NN`.

Neke karakteristike ovih tehnika se mogu uočiti i bez empirijske evaluacije. Tehnika `ArgoSmArT k-NN` zahteva poznavanje vremena rešavanja k susednih instanci pomoću svih algoritama. U slučaju velikog broja algoritama, kao u slučaju problema izbora heuristika i njihovih parametara gde broj mogućih kombinacija može da se meri hiljadama i desetinama hiljada, primena ove tehnike je prakticno nemoguća osim u slučaju $k = 1$ kada je za svaku instancu dovoljno poznavati makar jedan algoritam koji je efikasno rešava. S druge strane, `ArgoSmArT k-NN` ne zahteva nikakvo grupisanje instanci koje može predstavljati ograničenje u slučaju nedostatka adekvatnog korpusa.

U nastavku su prikazani rezultati primene datih tehnika na sledeće probleme:

1. problem izbora heuristika SAT rešavača pogodnih za rešavanje ulazne KNF instance,
2. problem izbora SAT rešavača pogodnog za rešavanje ulazne KNF instance i
3. problem izbora SAT rešavača za rešavanje zadate ne-KNF instance.

Drugi relevantni sistemi za rešavanje tih problema su diskutovani na kraju svakog poglavlja.

3.3 Izbor heuristika SAT rešavača

U ovom poglavlju će opšte tehnike prikazane u poglavlju 3.2 biti primenjene na problem izbora heuristika SAT rešavača `ArgoSAT` [64]. `ArgoSAT` predstavlja visoko podesivi CDCL SAT rešavač koji nudi više različitih heuristika za razne aspekte svog funkcionisanja.

Konfiguracije rešavača. U tabeli 3.2, prikazane su heuristike koje će biti korišćene sa svojim parametrima u skladu sa oznakama iz poglavlja 2.4. Jedan izbor heuristika za svaki od nabrojanih aspekata rada SAT rešavača nazivaće se *konfiguracijom*. Ukupan broj konfiguracija je 60 ($3 \cdot 5 \cdot 4$).

Tabela 3.2: Pregled heuristika korišćenih u eksperimentu.

Izbor promenljive	VS_{random}, $\mathsf{VS}_{VSIDS}^{1.0,\,1.0/0.95,\,\text{FREQ}}$ $\mathsf{VS}_{random}^{0.05} \circ \mathsf{VS}_{VSIDS}^{1.0,\,1.0/0.95,\,\text{FREQ}}$
Izbor polariteta promenljive	PS_{POS}, PS_{NEG}, $\mathsf{PS}_{random}^{0.5}$ $\mathsf{PS}_{polarity\_caching}^{\text{NEG}}$, $\mathsf{PS}_{polarity\_caching}^{\text{FREQ}}$
Otpočinjanje iznova	$\mathsf{R}_{no\_restart}$, $\mathsf{R}_{minisat}^{100,1.5}$, R_{luby}^{512}, $\mathsf{R}_{picosat}^{100,1.5}$

Korpus. Predložene tehnike biće primenjene i evaluirane na korpusu sa takmičenja SAT rešavača SAT 2002.[5] Iako postoje korpusi iz skorijih godina, prednost ovog korpusa je što se sastoji iz velikog broja familija iskaznih formula sličnog porekla. Ovo svojstvo ga čini zanimljivim za primenu tehnike `ArgoSmArT`. Ukupan broj instanci u korpusu je 1964, a broj familija (skupova instanci koji potiču od istog problema) je 38.

[5] `http://www.satcompetition.org/`

Empirijska evaluacija. Sve instance iz korpusa su rešene za svih 60 konfiguracija koje se mogu dobiti kombinovanjem heuristika iz tabele 3.2. Dozvoljeno vreme rešavanja za jednu instancu i jednu konfiguraciju je bilo 600s. Ukupan broj pokretanja SAT rešavača je bio 117840 ($= 1964 \cdot 60$). Eksperimenti su sprovedeni na klaster računaru sa 32 Intel Xeon procesora radnog takta od 2GHz sa dva jezgra i sa 2GB RAM memorije po procesoru. Ukupno procesorsko vreme potrošeno za eksperimente je oko 505 dana. U svrhe evaluacije, iz korpusa je slučajnim izborom izdvojena jedna trećina instanci koje će biti korišćene isključivo za testiranje.

Za funkciju rastojanja korišćena je funkcija d_3 koja se pokazala kao najbolja u eksperimentima opisanim u potpoglavlju 3.1.3. Kao i u tim eksperimentima, i u ovom su bila korišćena prva 33 atributa sistema `SATzilla`. Atributi nisu skalirani pošto korišćena funkcija d_3 uključuje normiranje razlika vrednosti atributa. Za funkciju cene koristi se *PAR10 skor* koji je jednak vremenu rešavanja instance ako je instanca rešena, a desetostrukoj vrednosti vremenskog ograničenja ukoliko instanca nije rešena [43]. PAR10 skor rešavača na skupu instanci jednak je sumi PAR10 skorova na pojedinačnim instancama. U slučaju tehnike `ArgoSmArT`, na osnovu rezultata prikazanih u potpoglavlju 3.1.3, najbolja vrednost za parametar k je 1. Za tehniku `ArgoSmArT k-NN`, eksperimentalno je utvrđeno da je najbolja vrednost $k = 5$. Pored ove vrednosti, biće prikazani rezultati i za `ArgoSmArT 1-NN` zbog posebnog značaja ove varijante prokomentarisanog u prethodnom poglavlju. U tabeli 3.3, prikazano je poređenje broja rešenih formula i medijana (središnjih vrednosti) vremena rešavanja koje su na odvojenom test skupu postigli `ArgoSmArT`, `ArgoSmArT 1-NN`, `ArgoSmArT 5-NN`, najbolja pojedinačna konfiguracija na trening skupu[6], portfolio sistem `ISAC` [49] i *virtualna najbolja konfiguracija* — teorijski najbolji prortfolio koji za svaku instancu bira najbolju od raspoloživih konfiguracija.

Tabela 3.3: Poređenje algoritamskih portfolija za izbor heuristika SAT rešavača.

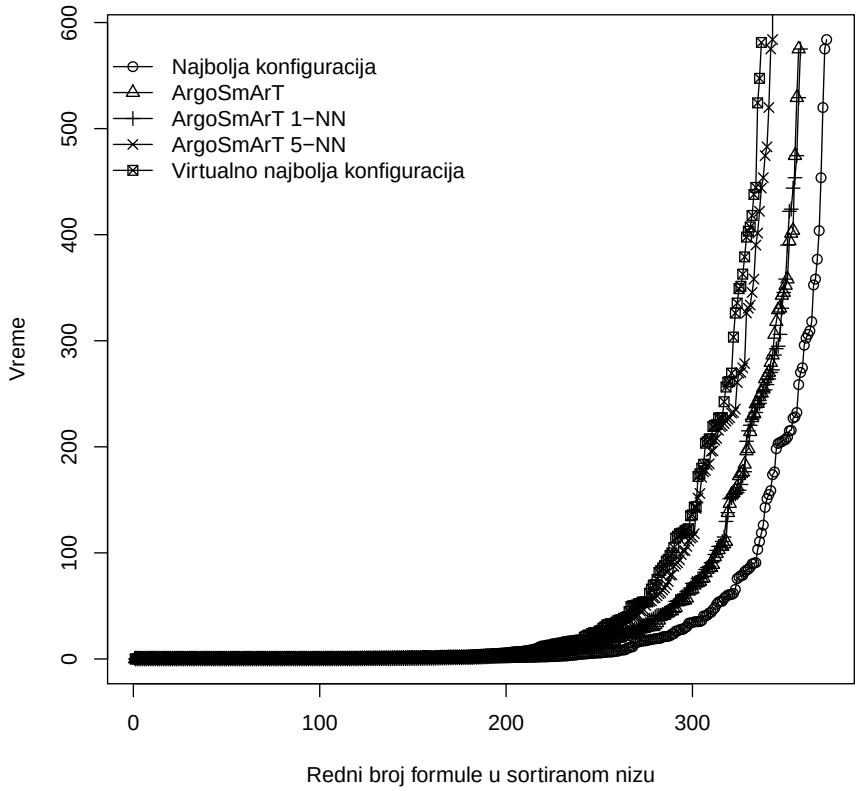
Metoda	Broj rešenih instanci	Medijana vremena rešavanja
Najbolja pojed. konf.	337	256.4s
`ArgoSmArT 1-NN`	343	224.2s
`ISAC`	357	118.9s
`ArgoSmArT`	357	110.7s
`ArgoSmArT 5-NN`	358	129.6s
Virt. najbolja konf.	372	58.7s

Kako bi se shvatio pun značaj prikazanih rezultata, treba imati u vidu vrlo brz rast niza sortiranih vremena rešavanja za sve tehnike prikazan na slici 3.9. Sa slike se

[6] U ovom eksperimentu to je bila konfiguracija ($\mathsf{VS}^{1.0,\,1.0/0.95,\,\text{FREQ}}_{VSIDS}$, $\mathsf{PS}^{\text{NEG}}_{polarity\_caching}$, $\mathsf{R}^{100,1.5}_{minisat}$).

uočava da je rešavanje svake sledeće instance mnogo teže nego rešavanje prethodnih instanci. Dodatno, svi metodi rešavaju 40-50% instanci iz test skupa za manje od 1s. Ovo znači da bi relativne razlike između brojeva rešenih instanci izgledale mnogo veće kad bi se lake instance uklonile.

Može se primetiti da algoritamski portfoliji značajno nadmašuju najbolju pojedinačnu konfiguraciju. Rezultat ostvaren od strane tehnike `ArgoSmArT 1-NN` je značajno lošiji od ostalih varijanti predloženih tehnika. Najverovatniji razlog za to je veliki broj vrlo lakih instanci u korpusu koje su uspešno rešene korišćenjem bilo koje konfiguracije, pa stoga jedna susedna instanca često ne daje dovoljno informacije za razlikovanje konfiguracija.



Slika 3.9: Sortirana vremena rešavanja za različite tehnike izbora konfiguracije.

Treba primetiti da iako `ArgoSmArT 5-NN` rešava najviše instanci, `ArgoSmArT` je ipak pogodniji za rešavanje ovog problema, s obzirom da rešava samo jednu formulu manje, a zahteva samo da se za svaku familiju zna jedna dobra konfiguracija Za razliku od njega, `ArgoSmArT k-NN` za $k > 1$ zahteva da se na svim instancama trening skupa zna vreme rešavanja svih mogućih konfiguracija. S obzirom na to da broj svih mogućih konfiguracija nekad može da se meri i desetinama hiljada, ovo je

ozbiljna mana tehnike ArgoSmArT k-NN.

Prikazani rezultati pokazuju da se inteligentnim, a ipak jednostavnim, izborom heuristika SAT rešavača može značajno povećati njegova efikasnost.

Relevantni sistemi. ISAC je sistem za konfigurisanje rešavača koji se može koristiti kao algoritamski portfolio [49]. Ovaj sistem je u literaturi opisan godinu dana posle tehnike ArgoSmArT [73] i koristi vrlo slične ideje. Za razliku od tehnike ArgoSmArT, ISAC automatski vrši grupisanje instanci u familije korišćenjem metoda klasterovanja. On uključuje GGA, sistem za nalaženje pogodnih konfiguracija za skupove instaci [3]. Takođse, instance se predstavljaju korišćenjem SATzilla atributa skaliranih u interval [-1,1]. Za ulaznu instancu, računaju se atributi, i nalazi se centar najbližeg klastera u trening skupu. Ukoliko je rastojanje do centra tog klastera manje od nekog praga, za rešavanje ulazne instance koristi se konfiguracija najbolja za taj klaster. U suprotnom se koristi konfiguracija koja je najbolja za ceo trening skup. S obzirom na sličnost ovog sistema sa predloženim tehnikama, sličnost dobijenih rezultata nije iznenađujuća.

SATzilla nije pogodan sistem za izbor konfiguracija SAT rešavača s obzirom na to da SATzilla mora da trenira po jedan predikcioni model za svaku konfiguraciju, a broj mogućih konfiguracija može biti izuzetno veliki. Kao što je rečeno, za primenu tehnika ArgoSmArT i ArgoSmArT 1-NN, dovoljno je poznavati po jednu dobru konfiguraciju za svaku familiju ili pojedinačnu instancu uz trening skupa, što je mnogo lakše obezbediti. Poređenje ovih tehnika sa sistemom SATzilla biće dato u sledećem poglavlju na drugom problemu.

Pored ovih, postoje i stariji pristupi koji nemaju dostupne implementacije niti učestvuju na takmičenjima rešavača [55], ili su orijentisani na rešavanje drugih, srodnih, problema [88].

3.4 Izbor KNF SAT rešavača

U ovom poglavlju će tehnike iz poglavlja 3.2 biti primenjene na problem izbora SAT rešavača koji kao svoj ulaz uzimaju formulu u KNF obliku, što je standardni ulaz za SAT rešavače. Kako je ovo isti zadatak koji rešava i sistem SATzilla eksperiment je dizajniran tako da se omogući direktno poređenje sa tim sistemom.

Komponentni rešavači. U eksperimentima je korišćeno 13 SAT rešavača koje koristi sistem SATzilla 2009 [98]. To su rešavači: Kcnfs04, March_dl2004, MiniSAT 2007, MiniSAT 2.0, Vallst, Zchaff rand, Tts, PicoSAT 846, MXC08, Adapt g2w

`SAT 0`, `Adaptg2wSAT Plus`, `Gnovelty Plus` i `SATenstein qcp`.

Korpus. Umesto rešavanja instanci iz trening skupa, kao podaci za primenu predloženih tehnika, korišćeni su podaci pomoću kojih je treninran sistem `SATzilla` 2009 [98] i koji su dostupni na sajtu posvećenom ovom sistemu.[7] `SATzilla` sistem je treniran korišćenjem 5883 instance sa SAT takmičenja od 2002 do 2005 i 2007 i sa SAT trka 2006[8] i 2008[9]. Podaci dostupni na sajtu uključuju informacije o rešavanju za 4701 instancu. Samo te instance su korišćenje u evaluaciji. Od dostupnih podataka korišćena su vremena rešavanja instanci pomoću različitih rešavača, dok su atributi izračunati iznova kako bi se izbeglo pretprocesiranje koje `SATzilla` vrši. Dozvoljeno vreme rešavanja instanci je bilo 1500s. Kada su odstranjene instance koje nisu rešene ni jednim rešavačem u ovom vremenu, kao i višestruka pojavljivanja nekih instanci, preostalo je 4276 instanci u trening skupu. Kao test skup korišćene su instance sa SAT takmičenja iz 2009. godine koji sadrži 1143 instance.

Empirijska evaluacija. Instance iz test skupa su rešene pomoću u nastavku navedenih sistema. Eksperimenti su izvršeni na klaster računaru sa 32 Intel Xeon procesora radnog takta od 2GHz sa dva jezgra i sa 2GB RAM memorije po procesoru.

`SATzilla` sistem ima tri specijalizovane verzije za tri različite vrste instanci — za slučajno generisane, za ručno dizajnirane i za industrijske [98]. Za predložene tehnike nisu pravljene specijalizovane varijante. Kako poređenje ne bi favorizovalo predložene tehnike, na svakoj od tri kategorije, u eksprimentima je korišćena odgovarajuća verzija `SATzilla` sistema. Za funkciju rastojanja u predloženim tehnikama korišćena je funkcija d_3 koja se pokazala kao najbolja u eksperimentima opisanim u potpoglavlju 3.1.3. Kao i u tim eksperimentima, i u ovom su bila korišćena prva 33 atributa sistema `SATzilla`. Za funkciju cene korišćen je PAR10 skor. U eksperimentalnu evaluaciju uključen je rešavač `MXC08`, kao najbolji pojedinačni rešavač na trening skupu, `SATzilla`, `ArgoSmArT`, `ArgoSmArT 1-NN`, `ArgoSmArT 9-NN` ($k = 9$ se pokazao kao najbolji izbor na trening skupu) i virtualno najbolji rešavač kao gornja granica mogućih performansi. Rezultati su dati u tabeli 3.4 i uključuju broj rešenih instanci i medijanu vremena rešavanja po kategorijama. Rezultati pokazuju da `ArgoSmArT k-NN` značajno nadmašuje ostale sisteme u svim kategorijama.

Uobičajena je praksa na takmičenjima SAT rešavača da se ponavljaju instance korišćene na prethodnim takmičenjima. Ovo rezultuje preklapanjem trening i test

[7] `http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/`
[8] `http://fmv.jku.at/sat-race-2006/`
[9] `http://baldur.iti.uka.de/sat-race-2008/`

Tabela 3.4: Poređenje algoritamskih portfolija za izbor SAT rešavača. Za svaki sistem dat je broj rešenih instanci i medijana vremena rešavanja na celom korpusu SAT 2009 i broj rešenih instanci po pojedinačnim kategorijama koje sadrže slučajno generisane, ručno dizajnirane i industrijske instance.

	T_{Sve}	N_{Sve}	N_{Slu}	N_{Ruc}	N_{Ind}
`MXC08`	>1500s	**355**	84	124	147
`ArgoSmArT`	874s	**609**	308	154	147
`SATzilla`	635s	**635**	375	128	132
`ArgoSmArT 1-NN`	390s	**685**	367	158	160
`ArgoSmArT 9-NN`	353s	**692**	390	149	153
VNR	115s	**816**	454	188	174

Tabela 3.5: Poređenje algoritamskih portfolija za izbor SAT rešavača. Za svaki sistem dat je broj rešenih instanci i medijana vremena rešavanja na korpusu SAT 2009 bez instanci koje su se pojavljivale na prethodnim takmičenjima i broj rešenih instanci po pojedinačnim kategorijama koje sadrže slučajno generisane, ručno dizajnirane i industrijske instance.

	T_{Sve}	N_{Sve}	N_{Slu}	N_{Ruc}	N_{Ind}
`MXC08`	>1500s	**243**	84	77	82
`ArgoSmarT`	895s	**475**	308	88	79
`SATzilla`	497s	**513**	375	68	70
`ArgoSmArT 1-NN`	343s	**533**	367	86	80
`ArgoSmArT 9-NN`	274s	**553**	390	83	80
VNR	76s	**659**	454	115	90

skupa. Zarad temeljnosti evaluacije, u tabeli 3.5 su prikazani rezultati na test skupu iz kojeg su odstranjene instance koje postoje u trening skupu. Tako su od 1143 preostale 894 instance.

Na osnovu rezultata iz tabele 3.5, može se primetiti da se na industrijskim instancama komponentni rešavač `MXC08` ponaša bolje od svih portfolio pristupa. Ovo verovatno znači da se industrijske instance u test skupu u značajnoj meri razlikuju od onih u trening skupu. Na celom test skupu `ArgoSmArT k-NN` značajno nadmašuju ostale sisteme. Mogući razlog za to je što `ArgoSmArT k-NN` za razliku od tehnike `ArgoSmArT` i sistema `SATzilla` ne koristi predefinisane grupe instanci niti prediktivne modele izgrađene bez znanja o instancu koju treba rešiti. Umesto toga, `ArgoSmArT k-NN` bira rešavač uzimajući u obzir samo susede instance koju treba rešiti u trening skupu. Na taj način se zanemaruje uticaj instanci koje nisu relevantne. Uistinu, sistem `SATzilla` popravlja svoje performanse upravo gradeći specijalizovane verzije za manje skupove instanci (slučajno generisane, ručno dizajnirane i industrijske) [99], što potkrepljuje ponuđeno objašnjenje.

Zanimljivo je da i `ArgoSmArT 1-NN` značajno nadmašuje sistem `SATzilla`. Pre-

dloženi pristup time pokazuje da budućnost portfolio sistema ne mora počivati na velikoj kompleksnosti.

Relevantni sistemi. Kako zbog svog značaja, tako i zbog jasnijeg kontrasta sa jednostavnošću predloženih tehnika, sistem `SATzilla` će biti detaljno opisan. `SATzilla` raspolaže skupom SAT rešavača za koje je temeljnom evaluacijom utvrđeno da se ponašaju različito jedni od drugih. Osnovni princip izbora rešavača je sledeći. `SATzilla` predstavlja SAT instance koje je potrebno rešiti pomoću određenog broja numeričkih atributa, i potom bira najbolji za rešavanje te instance na osnovu ovih atributa oslanjajući se na *empirijske modele težine problema* koji su dobijeni u fazi treniranja sistema. Empirijski modeli težine su statistički regresioni modeli čija je svrha predviđanje težine problema.

`SATzilla` je vrlo kompleksan sistem. Na datu instancu koju treba rešiti, `SATzilla` prvo, na kratko (na primer, po 5 sekundi), primenjuje dva *predrešavača* u nadi da će trivijalne instance biti vrlo brzo rešene bez primene sofisticiranih tehnika. Ukoliko instanca ne bude rešena predrešavačem, potrebno je izračunati atribute koji joj odgovaraju. Pošto računanje atributa može dugo da potraje, pre računanja atributa predviđa se vreme koje je potrebno za to računanje pomoću regresionog modela treniranog u ovu svrhu. Ukoliko je procenjeno vreme računanja preko dva minuta, računanje se ne vrši, već se primenjuje *rezervni SAT rešavač*. Ukoliko je predviđeno vreme manje od dva minuta, računaju se atributi i primenom empirijskih modela težine, predviđa se vreme rešavanja. Za rešavanje problema koristi se rešavač za koji je predviđeno vreme najmanje. Ukoliko ovaj rešavač nepravilno završi izvršavanje (na primer zbog manjka memorije ili greške u programiranju) pokreće se drugorangirani rešavač.

Podaci na kojima se treniraju empirijski modeli težine se dobijaju merenjem vremena rešavanja svih problema iz nekog izabranog korpusa instanci za trening pomoću svih SAT rešavača koje `SATzilla` koristi (uz korišćenje unapred određenog vremena nakon kojeg se izvršavanje prekida). Za svaku kategoriju SAT instanci koja se koristi na takmičenjima SAT rešavača — slučajno generisane, ručno pravljene i industrijske instance, konstruiše se poseban `SATzilla` sistem. Za svaki od tih sistema, trenira se poseban empirijski model težine. Predviđanje koje on daje se dobija kombinovanjem predviđanja uslovnih modela koji se treniraju posebno za zadovoljive i nezadovoljive instance. Kako bi ovakva kombinacija bila moguća, `SATzilla` koristi statističku ocenu verovatnoće zadovoljivosti instance koju rešava. Svaki od uslovnih modela se dobija na sledeći način. Prvo, polazeći od skupa polaznih atributa, vrši se izbor njegovog podskupa kako bi se našli atributi pri čijem korišćenju se najviše

smanjuje greška predviđanja na podacima za treniranje. Potom se u skup odabranih atributa dodaju proizvodi njihovih parova i ponovo se vrši izbor podskupa atributa. Potom, se trenira regresioni model za predviđanje vremena na osnovu izabranih atributa. Iz skupa rešavača koji su evaluirani na trening podacima, automatski se bira podskup rešavača korišćenjem randomizovane iterativne procedure. Predrešavači i rezervni rešavač se takođe automatski biraju.

Pristup *modeliranja latentnih klasa* se zasniva na statističkim modelima ponašanja rešavača koji se koriste za izbor rešavača [90]. Predloženi model pokušava da opiše zavisnosti između rešavača, problema i trajanja rešavanja. Prilikom treniranja, svaka instanca mora biti rešavana puno puta od strane svakog rešavača. Model se trenira na osnovu dobijenih vremena korišćenjem poznatog algoritma maksimizacije očekivanja. U toku faze testiranja, model se ažurira na osnovu novih podataka. Ovaj pristup omogućava i izbor rešavača i izbor vremena rešavanja. Autori prijavljuju da je njihov sistem grubo uporediv sa sistemom `SATzilla`.

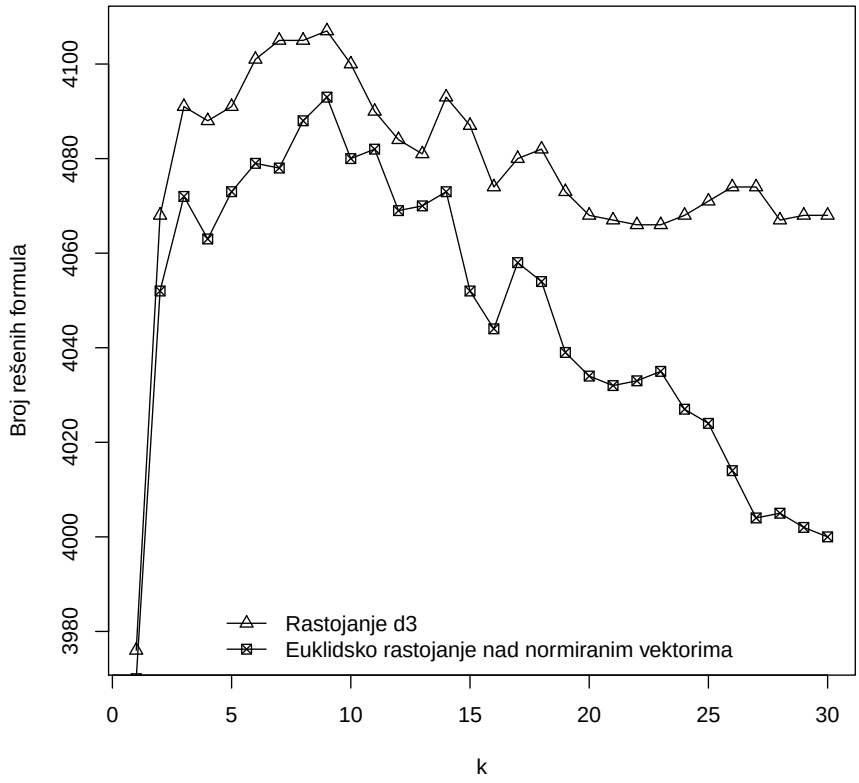
Pristup zasnovan na metodi k najbližih suseda je nezavisno razvijen i paralelno predložen od strane Malitskog i drugih [62]. Određene razlike u odnosu na predložene tehnike ipak postoje. Pomenuti pristup koristi euklidsko rastojanje i skaliranje atributa u interval [0,1]. Takođe, koristi svih 48 atributa sistema `SATzilla`. Implementacija ovog pristupa nije javno dostupna, ali je elemente koji se razlikuju bilo moguće implementirati u okviru implementacije tehnike `ArgoSmArT k-NN`. Broj rešenih instanci na trening skupu za oba sistema, za različito k, je prikazan na slici 3.10. Vidi se da je `ArgoSmArT k-NN` uniformno bolji od alternativnog pristupa.

Pored broja rešenih instanci, značajno je razmotriti i vreme računanja atributa. Vremena računanja za 33 atributa i za 48 atributa su data u tabeli 3.11. Vreme računanja za 48 atributa je značajno veće. Može se zaključiti da je uprkos sličnostima, predloženi pristup bolji.

Pored ovih, postoje i stariji pristupi koji nemaju dostupne implementacije niti učestvuju na takmičenjima rešavača [60, 100, 35, 41].

3.5 Izbor ne-KNF SAT rešavača

Problem zadovoljivosti iskaznih formula, SAT, po definiciji podrazumeva da se radi o formulama u konjunktivnoj normalnoj formi (KNF). Međutim, kad se formula transformiše u KNF oblik, strukturalna informacija u formuli, potencijalno važna i upotrebljiva u procesu rešavanja, se gubi. Kako bi se ovaj problem izbegao, razvijaju se algoritmi, takozvani ne-KNF SAT rešavači, koji proveravaju zadovoljivost formula koje nisu u KNF obliku [93, 44, 45]. Nažalost, i dalje nema puno takvih rešavača, a



Slika 3.10: Broj rešenih instanci iz trening skupa za `ArgoSmArT k-NN` kada se koristi rastojanje d_3 i euklidsko rastojanje nad normianim vektorima za k od 1 do 30.

Broj atributa/Vreme	Minimalno	Prosečno	Maksimalno
33	0,0017s	0,45s	51,2s
48	0.002s	19.2s	6257s

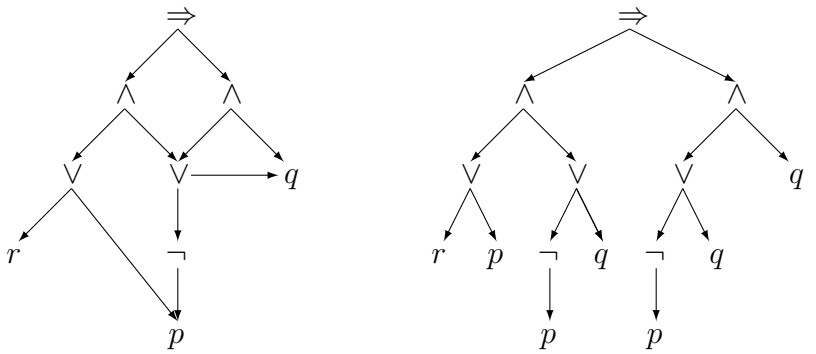
Slika 3.11: Vremena računanja `SATzilla` atributa.

postojeći rešavači nisu lako dostupni, ali se može očekivati da će se sa napretkom u oblasti to promeniti.

Kao što je primećeno u slučaju KNF SAT rešavača, algoritamski portfoliji su u stanju da značajno nadmaše performanse pojedinačnih rešavača. Slično ponašanje se može očekivati i u slučaju ne-KNF SAT rešavača, što motiviše izgradnju algoritamskog portfolija za tu vrstu rešavača. Portfoliji za KNF SAT rešavače se uglavnom oslanjaju na atribute sistema `SATzilla`. Sličnih atributa za formule koje nisu u KNF obliku, nema. Stoga, u ovom poglavlju se prvo predlažu atributi, inspirisani atributima sistema `SATzilla`, za formule u ne-KNF obliku.

Atributi za ne-KNF SAT instance. Prirodan način predstavljanja ne-KNF instanci je usmereni aciklični graf (UAG). Prednost predstavljanja ne-KNF instance pomoću UAG-a je u tome da se ponovljena pojavljivanja potformule u formuli zapisuju samo jednom.

Primer 3 *Na slici 3.12 prikazanje UAG i stablo koji odgovaraju formuli* $((r \vee p) \wedge (\neg p \vee q)) \Rightarrow ((\neg p \vee q) \wedge q)$. *Usled deljenja zapisa potformule* $\neg p \vee q$, *reprezentacija u obliku UAG-a je manja nego reprezentacija u obliku stabla.*



Slika 3.12: UAG (levo) i stablo (desno) za formulu $((r \vee p) \wedge (\neg p \vee q)) \Rightarrow ((\neg p \vee q) \wedge q)$

Stoga je prirodno formulisati atribute oslanjajući se na tu strukturu. Ne-KNF instance mogu uključivati veznike negacije, konjunkcije, disjunkcije, implikacije i ekvivalencije, što treba uzeti u obzir pri predlaganju atributa (proširenje na veći broj veznika je trivijalno). Predloženi atributi, podeljeni na tri grupe su dati na slici 3.13. Prva grupa se sastoji od jednostavnih atributa vezanih za veličinu problema. Druga grupa se sastoji od atributa vezanih za raspodelu pojavljivanja veznika, promenljivih i različitih polariteta u instanci. Treća grupa su atributi vezani za grafovsku strukturu. U početku je razmatran i širi skup atributa, ali su zadržani samo atributi koji se mogu izračunati u linearnom vremenu u odnosu na veličinu UAG-a. Atributi se računaju[10] kako za UAG koji odgovara instanci, tako i za stablo koje odgovara instanci. Pritom, pošto bi konstrukcija stabla bila memorijski i vremenski vrlo zahtevna, atributi vezani za stablo se računaju bez njegove eksplicitne konstrukcije — obilaskom UAG-a. Postoje jasne sličnosti sa atributima sistema `SATzilla`. Ti atributi uključuju neke atribute vezane za veličinu formule, atribute vezane za grafovsku strukturu i za polaritet promenljivih. Međutim, ne mogu se svi atributi sistema `SATzilla` formulisati za ne-KNF instance, poput atributa koji se direktno odnose na svojstva klauza (atributi 1,3,26-33 na slici 3.6). Za neke od njih, poput atributa koji se odnose na svojstva grafova i literala (atributi 2,4-25 na slici 3.6) mogu se definisati analogni atributi. Povrh toga, predloženi su i neki atributi bez

[10]Računanje atributa su implementirali Aleksandar Zeljić i Milan Todorović, studenti doktorskih studija na Matematičkom fakultetu.

analogona u skupu atributa sistema `SATzilla`. To su atributi koji se odnose pojavljivanja veznika, svojstva UAG-a i stabla i odnose tih njihovih svojstava (atributi 2-23,40-43,54-57 na slici 3.13). Pošto ne-KNF SAT rešavači nisu otvorenog koda, nije se moglo eksperimentisati sa atributima zasnovanim na algoritmima pretrage.

Atributi vezani za veličinu instance:

1. *Broj promenljivih*
2. *Broj pojavljivanja iskaznih veznika u stablu formule* l_f
3. *Broj pojavljivanja iskaznih veznika u UAG-u* l_d
4. *Veličina stabla formule* s_t
5. *Veličina UAG-a* s_d

6-7. *Faktor kompresije i njegova recipročna vrednost* s_t/s_d, s_d/s_t

8. *Broj pojavljivanja promenljivih u stablu formule* v_f

9-10. *Odnos broja pojavljivanja iskaznih veznika i promenljivih u stablu formule i njihova recipročna vrednost* l_f/v_f, v_f/l_f

Atributi vezani za veznike:

11-16. *Raspodela frekvencija* (C_1) *pojavljivanja veznika u stablu formule*: raspodela i entropija.

17-22. *Raspodela frekvencija* (C_2) *pojavljivanja veznika u UAG-u*: raspodela i netropija.

23. *Pirsonov koeficijent korelacije* $\rho(C_1, C_2)$

24-27. *Raspodela frekvencija* (V_1) *promenljivih u stablu formule*: prosek, minimum, maksimum i entropija.

28-30. *Udeo efektivno pozitivnih pojavljivanja promenljivih* $p/(p+n)$: prosek, koeficijent varijacije i entropija.

31-33. *Balansiranost polariteta promenljivih* $\min\{p, n\}/(p + n)$: prosek, koeficijent varijacije i entropija.

34-36. *Udeo efektivno pozitivnih pojavljivanja svake potformule* $p_f/(p_f + n_f)$: prosek, koeficijent varijacije i entropija.

37-39. *Balansiranost polariteta potformula* $\min\{p_f, n_f\}/(p_f + n_f)$: prosek, koeficijent varijacije i entropija.

Atributi vezani za grafovsku strukturu instance:

40-43. *Dužina putanja od korena do lista u stablu formule*: prosek, minimum, maksimum i koeficijent varijacije.

44-48. *Izlazni stepen čvorova UAG-a (arnost veznika)*: prosek, minimum, maksimum, koeficijent varijacije i entropija.

49-53. *Ulazni stepen čvorova UAG-a*: prosek, minimum, maksimum, koeficijent varijacije i entropija.

54-55. *Odnos dubine UAG-a i broja promenljivih i njegova recipročna vrednost.*

56-57. *Odnos broja grana i čvorova u UAG-u i njegova recipročna vrednost.*

Slika 3.13: Predloženi atributi za ne-KNF SAT instance.

Komponentni rešavači. Kao komponentni rešavači za porfolio, korišćena su tri rešavača — Nflsat [45], Lingeling [12] i kw_aig [2]. Dok je Nflsat pravi ne-KNF rešavač, Lingeling i kw_aig su u suštini KNF rešavači koji pre rešavanja vrše Cajtinovu transformaciju [95] kako bi dobili instancu u KNF obliku. Korišćeni su Nflsat i kw_aig pošto su oni učestvovali na SAT trci u okviru koje se vrši poređenje ne-KNF SAT rešavača i dostupni su. Lingeling je korišćen kao efikasniji naslednik rešavača PicoSAT koji je takođe učestvovao na SAT trci. Drugi dostupni ne-KNF rešavači nisu korišćeni pošto su u preliminarnim testovima pokazali dosta lošije rezultate od ovih. Zbog manjka rešavača, korišćeno je nekoliko konfiguracija svakog od dostupnih rešavača (9 konfiguracija za Nflsat, 8 za Lingeling i 1 za kw_aig) pri čemu je svaka od njih tretirana kao poseban rešavač.

Korpus. Polazni korpus se sastojao od 3923 ne-KNF instance. Korišćene su instance sa SAT trke 2008[11], SMT problemi instituta FMV[12] i strukturalne SAT instance FMV instituta[13]. Sve instance su zapisane u takozvanom AIG formatu u kojem se formule zapisuju pomoću veznika konjunkcije i negacije[14]. Svaka instanca je rešena svakim rešavačem i vreme rešavanja je zabeleženo. Maksimalno dozvoljeno vreme je bilo 900s. Ako instanca nije rešena ni jednim rešavačem, odstranjena je iz trening skupa (83 instance su odstranjene na ovaj način). Dodatno, instanca je odstranjena ako se može smatrati lakom, to jest ako je rešena od strane svih rešavača (3094 instance su uklonjene na ovaj način)[15]. Dodatno, odstranjene su i formule koje se pojavljuju u test skupu (31 instanca je uklonjena na ovaj način). Finalni trening skup se sastojao od 746 instanci. Kao test skup, korišćeno je svih 100 instanci sa SAT trke 2010[16].

Izbor atributa. Pre evaluacije na test skupu, izvršen je *izbor atributa unazad* na trening skupu. Izbor atributa unazad je uobičajena iterativna procedura za izbor atributa [99]. U svakom koraku procedure, za svaki atribut iz tekućeg skupa atributa, radi se sledeće. Atribut se izdvaja iz skupa, računa se PAR10 skor za portfolio na trening skupu bez tog atributa i atribut se vraća u skup. Najlošiji atribut, odnosno onaj čije uklanjanje rezultuje najvećim PAR10 skorom se beleži i uklanja. Ovo se radi dok skup atributa ne postane jednočlan. Na kraju se bira

[11] `http://baldur.iti.uka.de/sat-race-2008/downloads/AIG-SAT-Race-TS.tgz`, `http://baldur.iti.uka.de/sat-race-2008/downloads/SAT-Race-2008-AIG.tar`

[12] `http://fmv.jku.at/aiger/smtqfbv-aigs.7z`

[13] `http://fmv.jku.at/aiger/tip-k-ind-aigs-o1234g.zip`

[14] `http://fmv.jku.at/aiger/`

[15] Uklanjanje ovihinstanci ne utiče na rezultate. Njihovo prisustvo samo donekle usporava porfolio u fazi korišćenja.

[16] `http://baldur.iti.uka.de/sat-race-2010/`

najbolji evaluirani skup atributa — onaj sa najmanjim PAR10 skorom. U našem slučaju, 19 atributa je izabrano ovom procedurom: 1, 2, 4, 9, 10, 11, 16, 27, 28, 33, 35, 37, 39, 49, 50, 52, 55, 56, i 57. Samo ovi atributi su korišćeni u daljoj evaluaciji. U svakom koraku procedure, najbolji izbor za parametar k je bila vrednost 3, tako da je ova vrednost korišćena i u daljoj evaluaciji.

Empirijska evaluacija. Eksperimenti su sprovedeni na klaster računaru sa 32 Intel Xeon procesora radnog takta od 2GHz sa dva jezgra i sa 2GB RAM memorije po procesoru.

Atributi su izračunati za sve instance u trening skupu (najveća instanca je imala više od 4 miliona konjunkcija). Minimalno, prosečno i maksimalno vreme računanja atributa bilo je 0.008s, 0.89s i 21.9s, što pokazuje da je računanje atributa dovoljno efikasno za upotrebu u algoritamskim portfolijima.

Za izbor ne-KNF SAT rešavača korišćena je tehnika `ArgoSmArT k-NN`, pošto se na problemu izbora rešavača u prethodnom poglavlju pokazala superiornom. Takođe, za rastojanje je korišćena funkcija d_3, a za funkciju cene PAR10 skor. Ovaj pristup je upoređen sa najboljim pojedinačnim rešavačem na trening skupu, koji predstavlja minimalno očekivanje od portfolija i sa virtualnim najboljim rešavačem kao teorijskim maksimumom performansi portfolija. Tabela 3.8 prikazuje broj rešenih instanci iz test skupa i medijanu vremena rešavanja za svaki od pomenitih sistema.

Tabela 3.6: Broj rešenih formula i ukupno vreme rešavanja na test skupu. Vrsta *dobitak* označava odnos ostvarenog poboljšanja (u poređenju sa najboljim pojedinačnim rešavačem) u odnosu na moguće poboljšanje.

	Broj rešenih	Vreme
Najbolji pojedinačni rešavač	62	229
`ArgoSmArT 3-NN`	66	223
VNR	68	132
Dobitak	67%	6%

Tabela 3.8 pokazuje da je razlika između rezultata najboljeg fiksiranog rešavača i virtualnog najboljeg rešavača prilično mala. U rezultatima SAT trke za 2010,[17] razlika između prvog i poslednjeg rešavača je bila 5 formula. U tom svetlu, postignuti rezultati su pozitivni. Ipak, mogla se očekivati veća razlika sa 18 korišćenih rešavača. Kako bi portfolio postigao dobre rezultate, potrebno je da komponentni rešavači budu nezavisni u smislu da se vremena rešavanja među rešavačima na istim instancama značajno razlikuju. Da bi se razmotrilo ovo svojstvo, izračunata je

[17] `http://baldur.iti.uka.de/sat-race-2010/results.html`

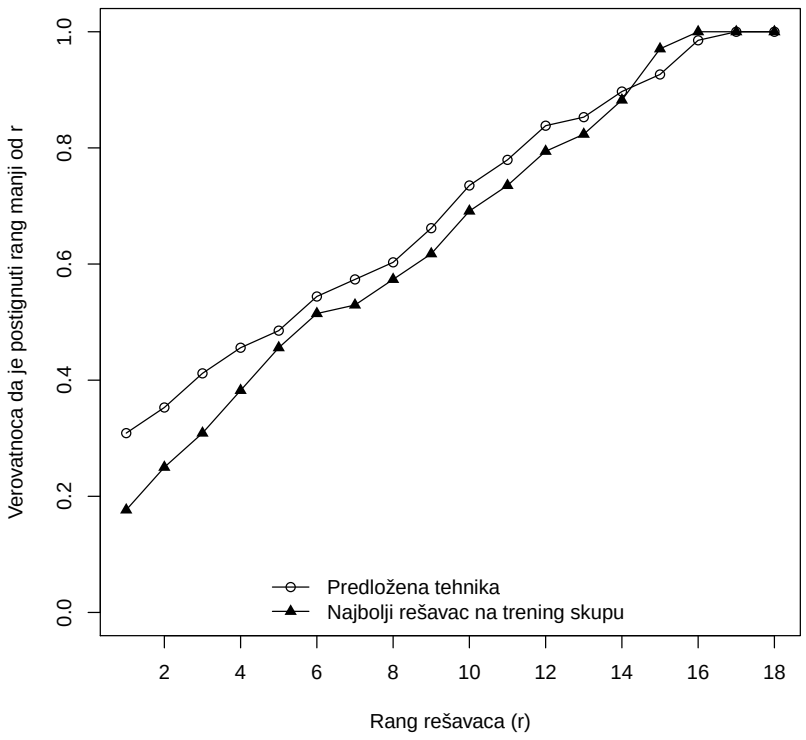
korelacija vremena rešavanja komponentnih rešavača na test skupu. Minimalni koeficijent korelacije između vremena rešavanja dva rešavača je bio 0.55, a prosečni 0.8. Ovi koeficijenti su vrlo visoki, što znači da komponentni rešavači nisu bili dovoljno nezavisni. Razlog za to je najpre korišćenje više konfiguracija svega 3 različita rešavača.

Performanse portfolija zavise od nezavisnosti njegovih komponentnih rešavača i od sposobnosti portfolija da detektuje bolje rešavače za ulazne instance. Za svaku instancu, rešavači se mogu rangirati prema vremenu za koje rešavaju tu instancu. Dobar portfolio će birati bolje rangirane rešavače. Verovatnoća izbora rešavača određenog ranga se može oceniti na test skupu. Kumulativna raspodela verovatnoće pokazuje verovatnoću izbora solvera zadatog ili boljeg ranga. Slika 3.14 prikazuje kumulativnu raspodelu verovatnoće za predloženi pristup i najbolji rešavač na trening skupu. Frekvencija izbora prvorangiranog rešavača je skoro duplo veća prilikom korišćenja portfolija, nego prilikom korišćenja rešavača koji se ponaša najbolje na trening skupu. Ovo pokazuje da predloženi atributi omogućavaju izgranju portfolija koji donosi bolje izbore nego što je bilo koji konstantan izbor na osnovu trening skupa.

Relevantni sistemi. Drugih portfolija za ne-KNF SAT instance nema, ali može se skrenuti pažnja na upotrebu atributa sistema `SATzilla` ili njihovih proširenja u različitim domenima. Na osnovu predviđanja vremena izvršavanja na osnovu atributa koje koristi `SATzilla`, uspešno je konstruisan algoritamski portfolio za problem zadovoljenja najvećeg broja klauza u SAT instanci (MAX-SAT problem) [69]. Portfolio rešavač za problem ispitivanja zadovoljivosti kvantifikovanih iskaznih formula je konstruisan na osnovu atributa inspirisanih atributima koje koristi `SATzilla`, ali je ukupan broj atributa bio veći zbog specifičnosti ovog domena [84]. Oslanjajući se na atribute sistema `SATzilla`, postignuti su dobri rezultati u predviđanju zadovoljivosti iskaznih formula pomoću metoda mašinskog učenja [26]. U oblasti zadovoljavanja ograničenja, razmatran je problem automatskog podešavanja algoritma pretrage pri čemu su korišćeni atributi inspirisani atributima sistema `SATzilla` [4].

3.6 Poređenje SAT rešavača

U ovom poglavlju će biti predložena metodologija za poređenje SAT rešavača zasnovana na statističkom testiranju hipoteza. Dodatno, biće prikazana i njena empirijska evaluacija u poređenju nekoliko SAT rešavača na dva korpusa. Razlog za predlaganje ovakve metodologije je velika varijacija vremena rešavanja jedne instance



Slika 3.14: Kumulativna raspodela verovatnoće ranga izabranog rešavača.

od strane jednog SAT rešavača, što dovodi u pitanje pouzdanost poređenja bez ocene statističke značajnosti.

3.6.1 Motivacija za poređenje SAT rešavača

U prethodnih 15 do 20 godina, SAT rešavači su doživeli mnogobrojna unapređenja. Odlučivanje o tome koji je rešavač ili koja je verzija rešavača bolja, se u okviru zajednice koja se bavi rešavanjem SAT problema obično vrši na osnovu empirijske evaluacije koja uključuje poređenje rešavača ili njihovih verzija. Takva poređenja omogućavaju identifikovanje i afirmaciju dobrih ideja. Upravo to je jedna od glavnih svrha SAT takmičenja i SAT trka koje su u značajnoj meri pomogle u usmeravanju razvoja SAT rešavača. I pored postojanja ovakvih takmičenja, glavna odgovornost za evaluaciju novih ideja je ipak na samim istraživačima.

Mnogi SAT rešavači nastaju modifikovanjem postojećih SAT rešavača, poput rešavača MiniSAT [28]. Kako bi se procenio značaj predložene modifikacije, obično se osnovna i modifikovana varijanta SAT rešavača pokreću na nekom skupu SAT instanci. Rešavač koji reši više instanci se smatra boljim. Isti postupak se koristi

Tabela 3.7: Broj rešanih instanci za ,,srećnu" i ,,nesrećnu" verziju svakog od rešavača.

	Industrijske		Bojenje grafova	
Rešavač	Srećan	Nesrećan	Srećan	Nesrećan
`MiniSAT 09z`	161	111	180	157
`minisat cumr r`	156	107	190	180
`MiniSAT`	141	93	200	183
`MiniSat2hack`	144	93	200	183

i u slučaju poređenja dva proizvoljna rešavača. Razlika u performansama koja je potrebna da bi se rezultat smatrao pouzdanim nije određena i predstavlja stvar ličnog uverenja istraživača. Mana ovog pristupa je što vremena rešavanja instanci mogu značajno da variraju u zavisnosti od trivijalnih svojstava formule ili rešavača, poput poretka klauza i literala u njima, imenovanja promenljivih i polazne vrednosti generatora pseudoslučajnih brojeva. Razlike u ovim svojstvima mogu dovesti do različitih eksperimentalnih rezultata, iako su, u matematičkom smislu, ova svojstva potpuno nebitna.

Kako bi se ova tvrdnja potkrepila, sproveden je sledeći eksperiment. Izabrana su četiri rešavača koja su učestvovala u takmičenju SAT 2009, u posebnom odeljku za modifikacije rešavača `MiniSAT`. Izabrani su prvorangirani i poslednji rešavač, polazni rešavač `MiniSAT` i rešavač sa srednjim rezultatima na osnovu rezultata takmičenja[18]. Korišćena su dva skupa instanci. Prvi se sastojao od 292 industrijske instance korišćene u ovom takmičenju, a drugi od 300 instanci koje potiču od problema bojenja grafova sa takmičenja SAT 2002. Svaki rešavač je pokretan na 50 *izmešanih* varijanti svake instance. Izmešane varijante se dobijaju slučajnim preraspoređivanjem klauza, literala u njima i preimenovanjem promenljivih. Rešavanje je trajalo najviše 1200 sekundi po instanci.

Prvo je analizirano koliko broj rešenih formula može da varira. Rešavač se smatra *srećnim* ukoliko za svaku formulu koju treba da reši, dobija njenu izmešanu varijantu koju rešava za najmanje vremena. Rešavač se smatra *nesrećnim* ukoliko za svaku formulu koju treba da reši, dobija njenu izmešanu varijantu koju rešava za najviše vremena (ili je ne rešava ukoliko takva varijanta postoji). U tabeli 3.7 dat je broj rešenih formula za svaki rešavač i svaki skup instanci. Uočava se da varijacija u broju rešenih formula može biti drastična.

Ispitan je i efekat primećene varijacije vremena na poređenje rešavača. Potvrđeno je da za se svaka dva rešavača, na skupu industrijskih instanci može izabrati pogodna izmešana varijanta za svaku instancu, tako da je prvi rešavač bolji od drugog, ali i

[18] `http://www.cril.univ-artois.fr/SAT09/results/ranking.php?idev=25`

da se izmešane varijante mogu izabrati tako da je drugi rešavač bolji od prvog (oba rešavača se pokreću na istoj izmešanoj varijanti svake formule). Naravno, ovakav događaj može biti izuzetno redak i njegova verovatnoća mora biti uzeta u obzir. Za svaki par rešavača, izvršeno je 10000 simuliranih poređenja pri čenu je izmešana varijanta svake instance slučajno birana kako bi se procenila verovatnoća da se jedan rešavač ispostavi kao bolji od drugog. Za većinu parova, izmena ishoda poređenja se pokazala kao vrlo malo verovatna. Međutim, ima izuzetaka. Pri poređenju rešavača `MiniSAT 09z` i `minisat cumr r` na industrijskim instancama, šanse za pobedu u poređenju su 92% prema 8%, pri poređenju rešavača `minisat2hack` i `MiniSAT` na industrijskim instancama, šanse su 94% prema 6%, i prilikom poređenja rešavača `MiniSAT` i `minisat2hack` na instancama vezanim za bojenje grafova, šanse su 74% prema 26%. Interesantno je primetiti da se ni dobijene šanse od 8% ili 6% za pobedu lošijeg rešavača ne bi mogle smatrati zanemarljivim u bilo kom eksperimentu koji uključuje proveru statističke značajnosti rezultata. Zanimljivost je da se na takmičenju SAT 2009 desio upravo manje verovatan raspored rešavača `MiniSAT` i `minisat2hack`.

Prikazani primeri se zasnivaju na mešanju instanci. Međutim, mešanje instanci nije od ključnog značaja za ovaj fenomen. Uloga mešanja je da učini da rešavač donese različite odluke u toku rešavanja u različitim pokretanjima i da se na taj način dobije informacija o raspodeli vremena rešavanja. Takav efekat se takođe mogao postići promenom polazne vrednosti generatora pseudoslučajnih brojeva[19]. Ilustracije radi, u tom slučaju, ,,srećna" varijanta rešavača MiniSAT rešava 144 instanci, a ,,nesrećna" 96, što je slično varijaciji dobijenoj prilikom mešanja. Ovi eksperimenti sugerišu da je korišćenje velikog broja formula u eksperimentima nedovoljno za donošenje pouzdanih zaključaka.

Pored prikazanog rezultata, u naučnoj praksi postoje i problemi vezani za način izvođenja zaključaka iz dostupnih eksperimentalnih rezultata. Nekada se rezultati prezentuju tabelama koje pokazuju da se jedan SAT rešavač ponaša bolje od drugog na jednom podskupu instanci, a lošije na drugom, bez jasnog zaključka o ukupnoj razlici. Dodatno, poređenja SAT rešavača se zaključuju bez diskusije o tome da li je dobijena razlika mogla biti dobijena slučajno ili je autentična.

U ovom poglavlju biće formulisana statistički zasnovana metodlogija poređenja SAT rešavača koja je u stanju da (i) ublažava efekat slučajnosti u poređenju, kao što je slučaj sa statističkim testovima u drugim oblastima, (ii) da odgovor postoji li ukupna pozitivna ili negativna razlika među rešavačima i (iii) da informaciju o statističkoj značajnosti razlike među rešavačima. Takva metodologija će omogućiti

[19]Upotreba slučajnih brojeva je vrlo važna u radu SAT rešavača.

lakše prepoznavanje dobrih i loših ideja, olakšavajući fokusiranje na one koje više obećavaju.

Postoji nekoliko problema na koje je potrebno obratiti pažnju prilikom formulisanja ovakve metodologije. Prvi je prisustvo *cenzurisanih* podataka. Ukoliko formula nije rešena u predviđenom vremenu, poznato je samo da je za njeno rešavanje potrebno više vremena, ali ne tačno koliko. Drugi problem je potreba za poređenjem raspodela vremena rešavanja umesto vremena rešavanja pri jednom pokretanju SAT rešavača, za koja je pokazano da su nepouzdana. Treći problem je nalaženje načina da se objedine zaključci dobijeni na pojedinačnim instancama iz korpusa kako bi se doneo ukupan zaključak.

Metodologija koja će biti predložena je zamišljena pre svega za detekciju poboljšanja nad nekim konkretnim rešavačem (npr. mnogi uspešni rešavači su dobijeni unapređivanjem rešavača MiniSAT), ali se može koristi bez ograničenja, za poređenje bilo koja dva rešavača. Dodatno, biće pokazano kako se ova metodologija može primeniti na rangiranje više rešavača. Pritom, metodlogija se ne bavi izborom instaci na kojima se rešavači porede. Pretpostavlja se da su korišćene instance reprezentativne za probleme od značaja.

3.6.2 Osnove poređenja SAT rešavača

U ovom potpoglavlju se opisuju koncepti i tehnike važni za razumevanje predložene metodologije i uvodi se notacija koja će biti korišćena.

Statističko testiranje hipoteza i pojam veličine efekta. Statističko testiranje hipoteza se bavi određivanjem da li je predložena hipoteza o nekoj populaciji (odnosno populacijama) tačna, na osnovu uzorka populacije (odnosno populacija). Test se obično obavlja tako što se pokušava odbaciti *nulta hipoteza* H_0. H_0 je obično tvrdnja o nedostatku efekta u razmatranoj populaciji (odnosno populacijama), poput nedostatka korelacije, razlike među prosecima i slično.

Kako bi se testiralo da li hipoteza H_0 važi, na uzorku se računa vrednost t neke test statistike T (koja zavisi od svrhe i formulacije testa), koja ima poznatu raspodelu verovatnoće. Verovatnoća dobijanja opažene ili ekstremnije vrednosti statistike, pod pretpostavkom da hipoteza H_0 važi, naziva se p vrednost. Ukoliko je p vrednost manja od nekog unapred određenog praga α (obično 0.05), opaženi događaj se smatra previše neverovatnim ukoliko hipoteza H_0 važi, pa se stoga hipoteza H_0 odbacuje. Takav rezultat se naziva *statistički značajnim na nivou statističkog testiranja* α. U suprotnom ($p > \alpha$), hipoteza H_0 se ne može odbaciti.

Što je p vrednost manja, veća je pouzdanost da opaženi efekat nije dobijen

slučajno. Međutim, mala p vrednost nije dovoljna da se zaključi da je efekat veliki u bilo kom praktično važnom smislu, pošto p vrednost zavisi kako od veličine efekta, tako i od veličine uzorka. Stoga, čak i ako je efekat statistički izuzento značajan, i dalje može biti previše mali da bi bio od bilo kakvog praktičnog značaja. Zbog toga su predložene mere *veličine efekta*. Postoji više standardnih mera veličine efekta [87, 36]. Jedna, često korišćena takva mera, je point-biserijska korelacija [87].

Point-biserijska korelacija. *Point-biserijska korelacija* ρ_{pb} između dve slučajne promenljive Z_1 i Z_{-1} je korelacija između vrednosti tih promenljivih i *indikatorske promenljive* koja uzima vrednost 1 za vrednosti promenljive Z_1, a -1 za vrednosti promenljive Z_{-1}. Uzoračka ocena ove veličine r_{pb} se računa po formuli:

$$r_{pb} = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i - \overline{Y})^2}}$$

gde su X_i vrednosti iz uzoraka promenljivih Z_1 i Z_{-1}, a Y_i su odgovarajuće vrednosti indikatorskih promenljivih. $\overline{X}$ i $\overline{Y}$ označavaju proseke vrednosti X_i i Y_i, a N je broj elemenata u celom uzorku.

Primer 4 *Neka je* $\{1, 2, 4\}$ *uzorak vrednosti promenljive* Z_1*, a* $\{3, 5, 6\}$ *uzorak vrednosti promenljive* Z_{-1}*. Vrednosti* X_1*,* X_2*,* X_3*,* X_4*,* X_5 *i* X_6 *mogu biti 1, 2, 3, 4, 5 i 6 (mogu biti i drugačije raspoređene). Vrednosti* Y_1*,* Y_2*,* Y_3*,* Y_4*,* Y_5 *i* Y_6 *su onda 1, 1, -1, 1, -1, -1. Uzoračka očena* r_{pb}*, veličine* ρ_{pb} *je -0.683.*

Veličine ρ_{pb} i r_{pb} uzimaju vrednosti u intervalu $[-1, 1]$. Apsolutna vrednost blizu 1 znači da su raspodele slučajnih promenljivih Z_1 i Z_{-1} bolje razdvojene, dok apsolutna vrednost blizu 0 znači da se te raspodele značajno preklapaju.

Ukoliko nema informacije o raspodeli podataka, podaci se pre računanja point-biserijske korelacije *rangiraju*, odnosno zamenjuju svojim indeksima, odnosno *rangovima* u sortiranom uzorku. Ukoliko su neke vrednosti u uzorku jednake, rang svake od njih je prosek indeksa tih vrednosti u sortiranom uzorku. Point-biserijska korelacija izračunata nad rangiranim podacima predstavlja instancu Spirmanovog koeficijenta korelacije [21, 20].

Ocena r_{pb} je asimptotski normalno raspodeljena sa prosekom ρ_{pb}. Varijansu vrednosti r_{pb} nije lako odrediti ukoliko se koristi rangiranje ili ukoliko raspodela podataka nije normalna, osim u slučaju $\rho_{pn} = 0$ [21, 20]. Ipak, ona može biti ocenjena *poduzorkovanjem* (eng. bootstrapping), tehnikom koja se zasniva na uzimanju novog uzorka sa ponavljanjem iz postojećeg uzorka i računanju varijanse na osnovu vrednosti statistike na poduzorcima [29, 30]. Varijansa ocene r_{pb} je jako zavisna od

vrednosti ρ_{pb}. Stoga se r_{pb} u statističkom testiranju obično koristi nakon primene Fišerove transformacije za stabilizaciju varijanse $z(x) = arctanh(x)$ [42]. Dodatno, transformisana promenljiva je bliže normalnoj raspodeli od polazne. Ona ima prosek $z(\rho_{pb})$, a varijansa se može oceniti izrazom $var(r_{pb})(1 - r_{pb}^2)^{-2}$ gde je $var(r_{pb})$ varijansa ocene r_{pb}.

U cilju interpretacije veličine vrednosti r_{pb}, mogu se pratiti Koenove praktične smernice — efekti sa vrednostima $|r_{pb}|$ u intervalima [0,0.1), [0.1,0.3), [0.3,0.5), i [0.5,1], se mogu smatrati, zanemarljivim, malim, srednjim i velikim. Treba imati u vidu da ove preporuke ne počivaju na objektivnoj teorijskoj analizi, već predstavljaju samo okvirne smernice.

Cenzurisani podaci. Pod *cenzurisanim podacima* podrazumevaju se podaci za koje se zna da su veći od nekog praga koji će biti nazvan *prag cenzurisanja*, ali se tačna vrednost ne zna. Jedan poznat test za poređenje uzoraka sa cenzurisanim podacima je Gehanov test [34]. Statistika koja se koristi u ovom testu se može formulisati na sledeći način [63]. *Objedinjeni uzorak* je uzorak koji uključuje elemente oba uzorka koji se porede. Pritom su moguća ponavljanja elemenata. Neka je U_i razlika u broju elemenata objedinjenog uzorka od kojih je i-ti element strogo veći minus broj elemenata od kojih je i-ti element strogo manji. U slučaju jedinstveno određenog praga cenzurisanja, cenzurisani elementi se tretiraju kao jednaki i veći od svih necenzurisanih elemenata[20]. Gehanova statistika se definiše kao

$$W_G = \frac{1}{|A_1||A_2|} \sum_{i \in A_1} U_i$$

gde je A_j skup indeksa u objedinjenom uzorku koji pripadaju elementima iz j-tog uzorka ($j = 1, 2$). Kako je pokazao Gehan [34], korišćenjem teorije U statistika [39, 57], Gehanova statistika je konzistentna ocena veličine $\omega = P(X > Y) - P(X < Y)$. Ona je asimptotski normalno raspodeljena i ima prosek ω. Varijansa statistike W_G se lako računa ukoliko važi $\omega = 0$. U drugim slučajevima može se koristiti poduzorkovanje [29, 30]. Kao u slučaju statistike r_{pb}, varijansa statistike W_G zavisi od vrednosti ω, smanjujući se kako se ω približava ekstremnim vrednostima -1 i 1.

3.6.3 Metodologija poređenja rešavača

Ideja predložene metodologije za poređenje dva rešavača je jednostavna. Neformalno, za svaku instancu iz nekog korpusa, računa se pogodno izabrana razlika u performansama dva rešavača na toj instanci. Ukoliko su performanse dva rešavača

[20] U slučaju varirajućeg vremena cenzurisanja, koriste se drugačije metode.

približno iste na datom korpusu, onda bi razlike izračunate na pojedinačnim instancama trebalo da se potiru pri sumiranju, pa prosek tih razlika ne bi trebalo da bude veliki. U ovoj metodologiji pojam raspodele vremena rešavanja je važan, ali mehanizam uzorkovanja je ostavljen nedefinisan i metodologija je primenljiva bez obzira na to koji se mehanizam koristi (mešanje instanci, menjanje polazne vrednosti generatora pseudoslučajnih brojeva, itd).

Skica metodologije. Neka promenljiva τ^j predstavlja vreme rešavanja instance F od strane rešavača S_j $(j = 1, 2)$ Kako vreme rešavanja može biti vrlo veliko, uvodi se maksimalno vreme rešavanja T. Razlika performansi dva SAT rešavača predstavlja se nekom funkcijom $\delta(\tau^1, \tau^2)$ koja meri pogodno izabranu razliku između raspodela ovih promenljivih. Pošto raspodele promenljivih nisu poznate, zaključivanje o njima će biti vršeno na osnovu njihovih uzoraka. Stoga, vrednost razlike δ između raspodela se aproksimira razlikom d između uzoraka. Razlike δ_i koje odgovaraju formulama F_i mogu se uprosečiti kako bi se dobila ukupna vrednost $\overline{\delta}$ koja meri razliku između rešavača na celom korpusu formula. Uzoračka ocena veličine $\overline{\delta}$ je prosek vrednosti d_i i biće označavana sa $\overline{d}$. Raspodela proseka $\overline{d}$ pri nultoj hipotezi $\overline{\delta} = 0$ biće označena sa Θ. p vrednost se može izračunati na osnovu ove raspodele i dobijenog proseka $\overline{d}$. Dobijeni rezultat se smatra statistički značajnim ukoliko je p vrednost manja od α za neki unapred određeni nivo značajnosti statističkog testiranja α. Procedura za poređenje rešavača je preciznije data na slici 3.15.

Očigledno, da bi se ova metodologija praktično primenila, potrebno je diskutovati njene različite aspekte. Najvažniji su izbor funkcije d i ocena raspodele Θ, odnosno utvrđivanje statističke značajnosti.

Izbor ocene razlike. Uloga funkcije d je da kvantifikuje razliku u performansama dva rešavača na jednoj instanci na osnovu uzoraka njihovih vremena rešavanja. U tu svrhu analiziramo tri različite mere veličine efekta.

Jedan od intuitivnih indikatora da se dva rešavača ponašaju jednako na nekoj instanci F bi bio da je verovatnoća da jedan rešavač reši instancu brže nego drugi jednaka verovatnoći da drugi rešavač reši tu instancu brže nego prvi, preciznije:

$$P(\tau^1 > \tau^2) = P(\tau^1 < \tau^2)$$

ili ekvivalentno

$$\omega = P(\tau^1 > \tau^2) - P(\tau^1 < \tau^2) = 0$$

Ove dve verovatnoće se ne moraju sabirati do 1 ukoliko su prisutni cenzurisani

```
function compareSolvers (
    S1 : solver,
    S2 : solver,
    N : integer,
    F : instance[],
    alpha: real
) : (different: bool, pValue : real, S : solver, difference: real)
begin
    foreach f in F
    begin
        for i := 1 to N
        begin
            T1[i] := sampleSolvingTime(S1,f)
            T2[i] := sampleSolvingTime(S2,f)
        end
        differenceOnInstance[f] := d(T1,T2)
    end
    difference := average(differenceOnInstance)
    Theta := estimate of distribution of difference if δ̄ = 0
    pValue := estimatePValue(difference,Theta)
    if pValue > alpha then
        return (false, pValue, NA, difference)
    if difference < 0 then
        return (true, pValue, S1, difference)
    else
        return (true, pValue, S2, difference)
end
```

Slika 3.15: Procedura za poređenje SAT rešavača.

podaci. Neka je

$$\pi = \frac{1-\omega}{2} = P(\tau^1 < \tau^2) + \frac{1}{2}P(\tau^1 = \tau^2)$$

Ova vrednost je intuitivna mera koja kombinuje jačinu empirijskog dokaza da se jedan rešavač ponaša bolje od drugog sa neizvesnošću koja proizilazi iz izjednačenosti rešavača na nekim instancama. Kako se u praksi svaka dva necenzurisana podatka razlikuju makar malo ukoliko se merenje vrši sa dovoljno preciznosti, slučaj izjednačenih vremena rešavanja se pre svega odnosi na situaciju kad oba rešavača nisu rešila neku instancu. Vrednost π je u literaturi korišćena kao mera veličine efekta [36]. Kako se veličina ω može oceniti statistikom W_G, veličina π se može oceniti statistikom $(1-W_G)/2$. Mana korišćenja veličina ω i π je nedostatak transformacije za stabilizaciju varijanse i približavanje normalnoj raspodeli poput one koja je poznata u slučaju point-biserijske korelacije (opisane u potpoglavlju 3.6.2), koja bi merenje statističke značajnosti učinila pouzdanijim. Iako je pogodnija za merenje statističke značajnosti zbog dostupnosti pomenute transformacije, za point-biserijsku korela-

ciju nije očigledno u kakvom je odnosu sa prethodnim pomenutim merama. Takođe, mera poput π je intuitivnija i lakša za razumevanje većem broju ljudi. Sledeća teorema ustanovljava odnos između uzoračke point-biserijske korelacije r_{pb} i statistike W_G. Za vrednosti X_i, S_X^2 označava $\sum(X_i - \overline{X})^2$, pri čemu $\overline{X}$ označava prosek vrednosti X_i.

Teorema 3 *Neka su T^1 i T^2 uzorci slučajnih promenljivih τ^1 i τ^2, veličine n_1 i n_2. Neka je X_i i-ti element u sortiranom objedinjenom uzorku, R_i njegov rang u uzorku, Y_i odgovarajuća vrednost indikatorske promenljive i r_{pb} uzoračka point-biserijska korelacija između R_i i Y_i. Tada, ukoliko nema jednakih vrednosti u objedinjenom uzorku i prag cenzurisanja je konstantan, važi sledeća relacija:*

$$W_G = r_{pb} S_R S_Y / n_1 n_2 \tag{3.7}$$

Dodatno, ukoliko $n_1/n_2 \to c$, gde je c konačna pozitivna konstanta, kada $n_1 \to \infty$, onda važi

$$\lim_{n_1 \to \infty} var(W_G) = \lim_{n_1 \to \infty} var(r_{pb}) S_R^2 S_Y^2 / n_1^2 n_2^2 \tag{3.8}$$

Dokaz. Neka je $N = n_1 + n_2$, neka su brojevi cenzurisanih podataka u uzorcima T^1 i T^2 redom c_1 i c_2 i neka je $C = c_1 + c_2$. Sa I^1 i I^2 biće označeni skupovi indeksa u objedinjenom uzorku necenzurisanih podataka iz uzoraka T^1 i T^2. Dodatno, neka je $I = I^1 \cup I^2$. Skup indeksa elementata prvog uzorka u objedinjenom uzorku svih podataka biće označen sa A_1.

Prvo će biti pokazano da važi relacija (3.7). Biće razmotreni izrazi $n_1 n_2 W_G$ i $S_R S_Y r_{pb}$ i biće pokazano da su jednaki. Biće korišćena Mantelova formulacija statistike W_G [63] i zapažanje da se ona može razložiti na zbir izraza vezanih za necenzurisane podatke, plus izraz za cenzurisane podatke.

$$\begin{aligned} n_1 n_2 W_G &= \sum_{i \in A_1} U_i = \sum_{i \in I^1} [(R_i - 1) - (N - R_i)] + c_1(N - C) \\ &= 2 \sum_{\in I^1} R_i - (n_1 - c_1)(N + 1) + c_1(N - C) \\ &= 2 \sum_{\in I^1} R_i - (n_1 - 2c_1)(N + 1) - c_1(C + 1) \end{aligned}$$

Razmotrimo $S_R S_Y r_{pb}$:

$$S_R S_Y r_{pb} = \sum_{i=1}^{N} (R_i - \overline{R})(Y_i - \overline{Y}) = \sum_{i=1}^{N} R_i Y_i - \sum_{i=1}^{N} R_i \overline{Y} - \sum_{i=1}^{N} \overline{R} Y_i + \sum_{i=1}^{N} \overline{RY}$$

pri čemu su $\overline{R}$ i $\overline{Y}$ proseci vrednosti R_i i Y_i. Kako su poslednje tri sume jednake, važi

$$S_R S_Y r_{pb} = \sum_{i=1}^{N} R_i Y_i - \sum_{i=1}^{N} R_i \overline{Y} = \sum_{i=1}^{N} R_i Y_i - E_1$$

pri čemu je $E_1 = (N+1)(n_1 - n_2)/2$ i dobija se korišćenjem činjenice da je suma rangova konstantna i jednaka $N(N+1)/2$ i da je $\overline{Y} = (n_1 - n_2)/N$. Razdvajanjem cenzurisanih i necenzurisanih podataka, dobija se

$$S_R S_Y r_{pb} = \sum_{i \in I} R_i Y_i + E_2 - E_1 = \sum_{i \in I^1} R_i - \sum_{i \in I^2} R_i + E_2 - E_1$$

pri čemu je $E_2 = (2N - C + 1)(c_1 - c_2)/2$, jer je $(2N - C + 1)/2$ prosečni rang cenzurisanih podataka. Pošto su svi necenzurisani podaci manji od cenzurisanih i pošto je suma njihovih rangova konstanta, druga suma se može izraziti preko prve sume, zahvaljujući čemu se dobija:

$$S_R S_Y r_{pb} = 2 \sum_{i \in I^1} R_i - (N - C)(N - C + 1)/2 + E_2 - E_1$$

Jednostavnim računom se dobija:

$$S_R S_Y r_{pb} = 2 \sum_{\in I^1} R_i - (n_1 - 2c_1)(N+1) - c_1(C+1)$$

čime je relacija (3.7) dokazana.

U daljem izlaganju, zbog tehničke pogodnosti, biće korišćene oznake $a_1 = c_1/n_1$ i $a_2 = c_1/n_2$. Jedina svojstva uzorka od kojih zavise veličine S_R i S_Y su upravo a_1 i a_2. Korišćenjem relacije (3.7) i osnovnih svojstava varijanse, za uslovnu varijansu statistike W_G dobija se $var(W_G|a_1, a_2) = \frac{S_R^2 S_Y^2}{n_1^2 n_2^2} var(r_{pb})$. Kako bi se dokazala relacija (3.8), dovoljno je pokazati da važi $var(W_G)/var(W_G|a_1, a_2) \to 1$ kada $n_1 \to \infty$ (s obzirom da važi $n_1/n_2 \to c$, dovoljno je razmatrati granične vrednosti kad $n_1 \to \infty$). Na osnovu zakona potpune varijanse, važi

$$var(W_G) = E_{a_1,a_2} var(W_G|a_1, a_2) + var_{a_1,a_2} E(W_G|a_1, a_2)$$

Na osnovu zakona velikih brojeva, a_1 i a_2 teže u verovatnoći svojim očekivanjima α_1 and α_2 kada $n_1 \to \infty$. Kako, na osnovu toga, verovatnoće vrednosti a_i takvih da $|a_i - \alpha_i| \geq \varepsilon$ teže nuli za svako $\varepsilon > 0$ kada $n_1 \to \infty$, važi

$$\frac{E_{a_1,a_2} var(W_G|a_1, a_2)}{var(W_G|a_1, a_2)} \to 1$$

kada $n_1 \to 1$ pošto i brojilac i imenilac u tom slučaju teže $var(W_G|\alpha_1, \alpha_2)$.

Što se tiče drugog izraza u formuli zakona potpune varijanse, po definiciji varijanse, važi

$$var_{a_1,a_2}E(W_G|a_1, a_2) = E_{a_1,a_2}E^2(W_G|a_1, a_2) - (E_{a_1,a_2}E(W_G|a_1, a_2))^2$$

Ova vrednost teži nuli kad $n_1 \to 1$ na osnovu istih argumenata kao u prošlom slučaju. Ovim je dokazana relacija (3.8). □

Uslovi teoreme se mogu smatrati ispunjenim u kontekstu rešavanja iskaznih formula. Kao što je već primećeno, pretpostavka da nema jednakih vrednosti u objedinjenom uzorku za necenzurisane podatke je realistična pošto se vremena rešavanja u praksi makar malo razlikuju. Takođe, pretpostavka o jedinstvenom pragu cenzurisanja je standardna u evaluaciji rešavača. Poslednju pretpostavku je trivijalno ispuniti pošto se veličina uzoraka može kontrolisati. Na primer, mogu se koristiti uzorci jednake veličine za oba rešavača.

Teorema pokazuje da prilikom poređenja SAT rešavača, korišćenje bilo koje od statistika W_G i r_{pb} vodi ka istim zaključcima u poređenju, makar u slučaju velikih uzoraka. Razlog za to je što p vrednost zavisi samo od količnika test statistike i korena njene varijanse, koji su prema prethodnoj teoremi približno jednaki kada su uzorci veliki. Stoga, ima smisla koristiti ove statistike naizmenično ili zajedno. Tehnički najpogodnija point-biserijska korelacija se može koristiti prilikom računanja p vrednosti, ali se u rezultatima poređenja dva SAT rešavača, kao intuitivnija, može prijaviti i ocena vrednosti π.

Utvrđivanje statističke značajnosti. Smatra se da dva rešavača imaju jednake performanse ukoliko važi $\rho_{pb} = 0$, odnosno ukoliko r_{pb} nije značajno različito od 0 u smislu statističkog testiranja. Statističko testiranje zasnovano na statistici r_{pb} se obično vrši posle Fišerove transformacije prikazane u potpoglavlju 3.6.2. Kako bi se proverila statistička značajnost razlike između dva rešavača na celom korpusu, za svako r_i računa se vrednost $z(r_i)$. Pošto su statistike $z(r_i)$ asimptotski normalno raspodeljene, na osnovu svojstava normalne raspodele sa prosekom $z(\rho_i)$ i varijansom $var(r_i)/(1 - r_i^2)^2$, važi da je i prosek $\overline{z}$ asimptotski normalno raspodeljen:

$$\overline{z} \sim \mathcal{N}\left(\frac{1}{M}\sum_{i=0}^{M} z(\rho_i), \frac{1}{M^2}\sum_{i=1}^{M}\frac{var(r_i)}{(1 - r_i^2)^2}\right)$$

Kako bi se ustanovilo da li je moguće odbaciti nultu hipotezu $\overline{\delta} = 0$, potrebno je proveriti da li dobijeni prosek $\overline{z}$ značajno odstupa od njegove očekivane vrednosti

0 pri nultoj hipotezi. p vrednost (dvostrana) se može izračunati na osnovu formule $2(1 - \Phi(\overline{z}/\sqrt{var(\overline{z})}))$, gde je Φ funkcija raspodele $\mathcal{N}(0, 1)$. Na kraju, p vrednost je potrebno uporediti sa nekim unapred izabranim pragom statističke značajnosti (obično 0.05 ili 0.01). Ukoliko je p vrednost manja od tog praga, nulta hipoteza se odbacuje i konstatuje se da su performanse rešavača značajno različite. Koji od dva rešavača je bolji odlučuje se na osnovu znaka vrednosti $\overline{d}$ ili $\overline{z}$, kao što je prikazano na slici 3.15.

Poređenje više rešavača. U slučaju poređenja više rešavača, pored poznavanja svih pojedinačnih poređenja, potrebno je raspolagati i metodom rangiranja. Važan problem vezan za primenu statističkih testova je njihova potencijalna netranzitivnost. Pri tome, takva mogućnost nije mana ni jednog testa, već prirodni probabilistički fenomen. Naime, postoje primeri slučajnih promenljivih A, B i C takvih da važi $P(A < B) > \frac{1}{2}$ i $P(B < C) > \frac{1}{2}$, ali da $P(A < C) > \frac{1}{2}$ ne važi. Efronove kockice predstavljaju popularnu ilustraciju ovog fenomena [17]. Kako bi se izbegli ovakvi problemi koji onemogućavaju naivno rangiranje na osnovu rezultata pojedinačnih poređenja, može se koristiti, na primer, metod Kendala i Veija [50] koji je dizajniran za situacije koje uključuju netranzitivne rezultate poređenja.

3.6.4 Empirijska evaluacija

U ovom potpoglavlju su prikazana dva eksperimenta. Prvi se tiče broja izmešanih varijanti instance koje su potrebne za primenu metodologije, a drugi prikazuje rezultate primene prikazane metodologije. U oba eksperimenta su korišćena ista četiri rešavača i dva korpusa kao u uvodu ovog poglavlja. Za nivo statističke značajnosti α uzet je uobičajeni nivo 0.05. Uzorak iz raspodele vremena rešavanja za svaku formulu je uzet tako što je rešeno 50 izmešanih varijanti te formule sa vremenskim ograničenjem od 1200 sekundi. Ukoliko su sve izmešane varijante neke formule rešene za manje od 0.1 sekunde[21] od strane oba rešavača ili ni jedna izmešana varijanta nije rešena od strane ni jednog rešavača, instance su odstranjivane kao neinformativne. Za funkciju d izabrana je ocena point-biserijske korelacije r_{pb}. Varijansa ocene r_{pb} je ocenjena poduzorkovanjem sa 100000 poduzoraka [29].[22]

Prvo važno pitanje vezano za primenu predložene metodologije je njena računska zahtevnost koja se odražava kroz broj izmešanih varijanti koje je potrebno rešavati kako bi se dobila pouzdana ocena veličine efekta i statističke značajnosti. Pored

[21] Najviše jedna industrijska instanca i 30 instanci vezanih za bojenje grafova su odbačeni na osnovu ovog kriterijuma u bilo kom poređenju.

[22] Izvorni kôd softvera koji je korišćen za sve statističke kalkulacije dostupan je na adresi `http://www.matf.bg.ac.rs/~nikolic/solvercomparison/sc.zip`

Tabela 3.8: Ocene korelacije ρ_{pb} pri poređenju različitih verzija `MiniSAT` rešavača. Umesto celih imena, date su samo oznake koje ih razlikuju.

	Industrijske				Bojenje grafova			
	`09z`	`cumr r`	`polazni`	`hack`	`09z`	`cumr r`	`polazni`	`hack`
`09z`	-	-0.097	-0.249	-0.229	-	0.206	0.453	0.461
`cumr r`	0.097	-	-0.241	-0.208	-0.206	-	0.327	0.333
`polazni`	0.249	0.241	-	0.072	-0.453	-0.327	-	-0.001
`hack`	0.229	0.208	-0.072	-	-0.461	-0.333	0.001	-

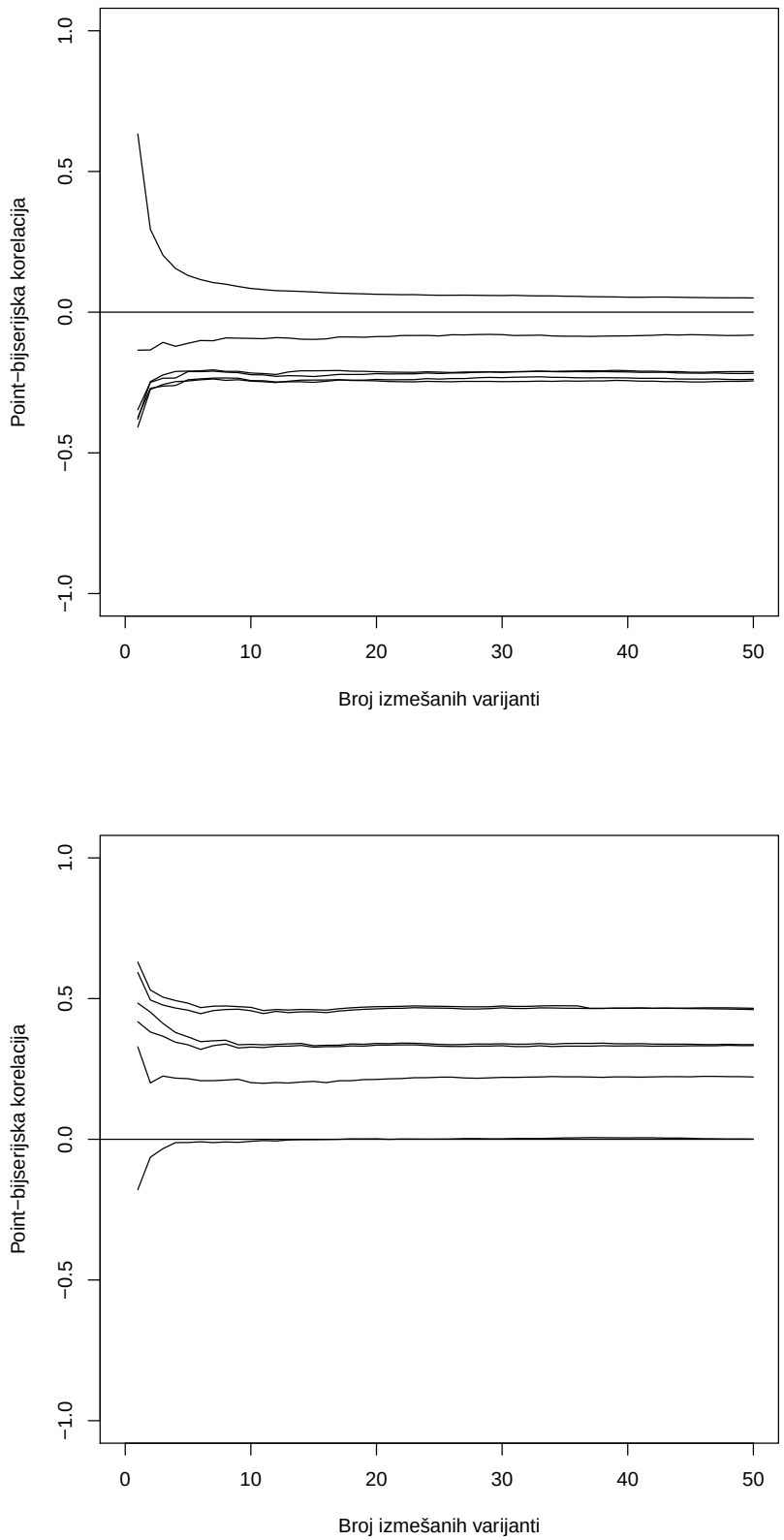
računske zahtevnosti, povećavanje broja izmešanih varijanti vodi ka smanjenju p vrednosti usled povećanja uzorka, iako bez povećanja veličine efekta. Praktični savet u ovakvim slučajevima je ne uvećavati uzorak pošto je ocena veličine efekta stabilizovana [18]. Kako bi se proverilo koji je broj izmešanih varijanti potreban, za svaka dva rešavača, na grafiku je prikazana vrednost r_{pb} za broj izmešanih varijanti od 1 do 50. Grafici za oba korpusa su dati na slici 3.16. Grafici sugerišu da je dovoljan broj izmešanih varijanti između 10 i 15.

U tabeli 3.8, prikazane su ocene statistike ρ_{pb} pri poređenju svaka dva rešavača koristeći 15 izmešanih varijanti svake formule. Dobijeni rezultati nisu iznenađujući s obzirom na informacije prikazane u tabeli 3.7. U svim poređenjima p vrednosti (dvostrane) su manje od 0.001, osim prilikom poređenja originalnog `MiniSAT` rešavača i njegove modifikacije `MiniSat2hack` na instancama poteklim iz bojenja grafova. U tom slučaju p vrednost je 0.945. I pored visoke statističke značajnosti, neke razlike se mogu smatrati zanemarljivim u svetlu preporuka za interpretaciju veličine r_{pb} datih u potpoglavlju 3.6.2. U ovom eksperimentu nema netranzitivnosti i rangiranje je lako izvesti iz tabele. U slučaju industrijskih instanci, rangiranje je $S_1 S_2 S_4 S_3$, a u slučaju instanci poteklih iz bojenja grafova, rangiranje je $S_3 S_4 S_2 S_1$, gde su oznake iste kao u tabeli 3.8.

3.6.5 Drugi pristupi

Postoji više radova na temu poređenja SAT rešavača. Le Ber i Simon su prvi uočili značaj ovog pitanja u oblasti rešavanja SAT problema [8]. Takođe, primećeno je da mešanje instanci može da dovede do zamene redosleda rešavača. Sugerisano je da se u korpuse mogu dodati izmešane varijante formula. S druge strane, njihov rad se bavi standardnim načinom poređenja rešavača. Odemar i Simon su dalje analizirali uticaj mešanja na broj rešenih formula i zaključili da on može biti značajan [5].

Ecioni i Ecioni su predložili statističke testove za cenzurisane podatke u evaluaciji metoda učenja zarad ubrzavanja (eng. speedup learning), ali poređenje vremenskih



Slika 3.16: Grafici statistike r_{pb} za industrijske instace (gore) i za instance potekle iz problema bojenja grafova (dole) u funkciji broja izmešanih varijanti koje se koriste za njeno računanje.

raspodela nije razmatrano [31]. Brglez i saradnici naglašavaju značaj statističkog pristupa poređenju SAT rešavača [15, 16]. Dodatno, uočen je značaj raspodele vremena rešavanja za poređenje SAT rešavača. Oni su primenili statističke testove u poređenju SAT rešavača, ali samo na po jednoj instanci. Nije razvijena puna metodologija koja bi omogućila korišćenje korpusa instanci kombinovanjem rezultata testiranja na pojedinačnim instancama, niti je uočen značaj pojma veličine efekta, koji je važan u takvoj metodologiji. Rangiranje više rešavača i probelm netranzitivnosti takođe nisu razmatrani.

Metodama za rangiranje sistema u automatskom rezonovanju i svojstvima koja takve metode treba da imaju bavio se Pulina [85]. Neka od ovih svojstava su stabilnost pri variranju veličine korpusa, stabilnost pri variranju dozvoljenog vremena rešavanja i koreliranost ranga rešavača prilikom primene evaluirane metode rangiranja i njegovog doprinosa virtualnom najboljem rešavaču sastavljenom od svih rešavača koji se porede. U ovom radu se ne razmatra statističko poređenje rešavača.

Glava 4

CDCL pretraga za dokazivanje u koherentnoj logici

U ovoj glavi će biti formulisan sistem pretrage za dokazivanje u koherentnoj logici zasnovan na CDCL sistemu pretrage za ispitivanje zadovoljivosti iskaznih formula. Dokazana su i najvažnija svojstva ovog sistema — saglasnost i potpunost. Ovaj sistem predstavlja odgovor ili makar polaznu tačku za različite izazove u dokazivanju teorema u koherentnoj logici, koji će biti analizirani u nastavku. Pored CDCL sistema pretrage, prikazana je i njegova implementacija — dokazivač `Calypso`. Ovaj dokazivač se pokazao superiornim u odnosu na druge dokazivače za koherentnu logiku, kao i u odnosu na najefikasniji dokazivač za opštu logiku prvog reda primenjen u ovom fragmentu.

4.1 Osnovni pojmovi koherentne logike

Kohrentna logika je fragment logike prvog reda koji se sastoji od formula oblika:

$$\forall \vec{v}\ (p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v}) \Rightarrow \exists \vec{y}\ Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v}, \vec{y})),$$

gde su $\vec{v}$ i $\vec{y}$ konačni nizovi promenljivih, $p_1(\vec{v}), \ldots, p_n(\vec{v})$ označavaju atomičke formule koje uključuju promenljive iz niza $\vec{v}$, a $Q_1(\vec{v}, \vec{y}), \ldots, Q_m(\vec{v}, \vec{y})$ konjunkcije atomičkih formula koje uključuju promenljive iz nizova $\vec{v}$ i $\vec{y}$. U slučaju da važi $m = 0$, umesto desne strane implikacije, piše se $\perp$. Aksiome i mnoge teoreme u disciplinama poput geometrije i algebre se mogu prirodno predstaviti u ovoj logici. Postojanje prirodnog deduktivnog sistema, bliskog sistemu prirodne dedukcije, čini da koherentna logika bude posebno pogodna za automatsko generisanje čitljivih dokaza matematičkih teorema. U nastavku ovog poglavlja biće predstavljena sintaksa

i semantika logike prvog reda koji su primenljivi i u koherentnoj logici, prirodni deduktivni sistem koherentne logike i svojstva koherentne logike.

4.1.1 Sintaksa i semantika logike prvog reda

U tekstu se koriste naredne definicije osnovnih pojmova logike prvog reda [46].

Definicija 9 (Signatura) Signatura *je uređena trojka* $\mathcal{L}(\Sigma, \Pi, ar)$*, gde je* Σ *prebrojiv skup* funkcijskih simbola*,* Π *prebrojiv skup* predikatskih simbola*, a* $ar : \Sigma \cup \Pi \to \mathbb{N}$ *funkcija* arnosti*, pri čemu važi* $\Sigma \cap \Pi = \emptyset$*.*

Definicija 10 (Skup termova) *Skup* termova $\mathcal{T}_{\mathcal{L}}$ *nad signaturom* $\mathcal{L}$ *i prebrojivim skupom promenljivih* V *je najmanji skup za koji važi:*

- *svaki simbol konstante (funkcijski simbol arnosti 0) je term;*
- *svaki simbol promenljive je term;*
- *ako je* f *funkcijski simbol za koji je* $ar(f) = n$ *i* $t_1, \ldots, t_n$ *su termovi, onda je* $f(t_1, \ldots, t_n)$ *term.*

Definicija 11 (Skup atomičkih formula) Skup atomičkih formula *(ili kraće* atoma*) nad signaturom* $\mathcal{L}$ *i prebrojivim skupom promenljivih* V *je najmanji skup za koji važi:*

- *logičke konstante konstante* $\top$ *i* $\bot$ *su atomičke formule;*
- *ako je* p *predikatski simbol za koji je* $ar(p) = n$ *i* $t_1, \ldots, t_n$ *su termovi, onda je* $p(t_1, \ldots, t_n)$ *atomička formula.*

Definicija 12 (Skup formula logike prvog reda) Skup formula logike prvog reda *(ili kraće* formula*)* $\mathcal{F}_{\mathcal{L}}$ *nad signaturom* $\mathcal{L}$ *i prebrojivim skupom promenljivih* V *je najmanji skup za koji važi:*

- *svaka atomička formula je formula logike prvog reda;*
- *ako je* A *formula logike prvog reda, onda je i* $(\neg A)$ *formula logike prvog reda;*
- *ako su* A *i* B *formule logike prvog reda, onda su i* $(A \wedge B)$*,* $(A \vee B)$*,* $(A \Rightarrow B)$ *i* $(A \Leftrightarrow B)$ *formule logike prvog reda;*
- *ako je* A *formula logike prvog reda i* x *promenljiva, onda su* $(\forall x\ A)$ *i* $(\exists x\ B)$ *formule logike prvog reda.*

Prilikom pisanja formula, zagrade mogu biti izostavljene i u tom slučaju se podrazumeva prioritet operatora u sledećem poretku, od najvišeg ka najnižem: $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$.

Bazni term je term koji ne sadrži ni jednu promenljivu. *Bazna formula* je formula koja ne sadrži ni jednu promenljivu.

Definicija 13 (Slobodno i vezano pojavljivanje) Slobodno pojavljivanje *i* vezano pojavljivanje *u formuli se definiše na sledeći način:*

- *svako pojavljivanje promenljive u atomičkoj formuli je slobodno u toj formuli;*
- *svako pojavljivanje promenljive koje je slobodno u formuli A je slobodno i u formuli $\neg A$; svako pojavljivanje promenljive koje je vezano u formuli A je vezano i u formuli $\neg A$;*
- *svako pojavljivanje promenljive koje je slobodno u formuli A ili u formuli B je slobodno i u formulama $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ i $(A \Leftrightarrow B)$; svako pojavljivanje promenljive koje je vezano u formuli A ili u formuli B je vezano i u formulama $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ i $(A \Leftrightarrow B)$;*
- *svako slobodno pojavljivanje promenljive različite od x u formuli A je takođe slobodno u formulama $\forall x\ A$ i $\exists x\ A$; pojavljivanje promenljive x u formulama $\forall x\ A$ i $\exists x\ A$ je vezano.*

Promenljiva je slobodna *u formuli ako i samo ako ima slobodno pojavljivanje u toj formuli. Promenljiva je* vezana *u formuli ako i samo ako ima vezano pojavljivanje u toj formuli.*

Definicija 14 ($\mathcal{L}$-struktura) *Za datu signaturu $\mathcal{L}$, $\mathcal{L}$-struktura $\mathcal{D}$ je par (D, I), gde je D neprazan skup koji se naziva* domen*, a I funkcija, koja se naziva* interpretacijom*, pri čemu važi sledeće:*

- *svakom simbolu konstante c iz Σ, funkcija I pridružuje jedan element c_I iz D;*
- *svakom funkcijskom simbolu f iz Σ za koji je $ar(f) = n$ i $n > 0$, funkcija I pridružuje jednu totalnu funkciju $f_I : D^n \to D$;*
- *svakom predikatskom simbolu p iz Π za koji je $ar(p) = n$ i $n > 0$ funkcija I pridružuje jednu totalnu funkciju $p_I : D^n \to \{0, 1\}$.*

Valuacija za prebrojiv skup promenljivih V u odnosu na domen D je preslikavanje $v : V \to D$.

Definicija 15 (Funkcija istinitosne vrednosti) *Neka je $\mathcal{D} = (D, I)$ $\mathcal{L}$-struktura za neku signaturu $\mathcal{L}$ i v valuacija za skup promenljivih V u odnosu na domen D. Tada se* funkcija istinitosne vrednosti *I_v definiše na sledeći način:*

- *$I_v(x) = v(x)$ za $x \in V$;*
- *$I_v(c) = c_I$ za simbol konstante c iz Σ;*
- *$I_v(f(t_1, \ldots, t_n)) = f_I(I_v(t_1), \ldots, I_v(t_n))$ za funkcijski simbol f iz Σ za koji je $ar(f) = n$;*
- *$I_v(\top) = 1$, $I_v(\bot) = 0$;*

- $I_v(p(t_1, \ldots, t_n)) = p_I(I_v(t_1), \ldots, I_v(t_n))$ *za predikatski simbol* p *iz* Π *za koji je* $ar(f) = n$*;*
- $I_v(\neg A)$ *je 1 ako je* $I_v(A) = 0$*, a 0, inače;*
- $I_v(A \wedge B)$ *je 1 ako je* $I_v(A) = 1$ *i* $I_v(B) = 1$*, a 0, inače;*
- $I_v(A \vee B)$ *je 0 ako je* $I_v(A) = 1$ *i* $I_v(B) = 1$*, a 1, inače;*
- $I_v(A \Rightarrow B)$ *je 1 ako je* $I_v(A) \leq I_v(B)$*, a 0, inače;*
- $I_v(A \Leftrightarrow B)$ *je 1 ako je* $I_v(A) = I_v(B)$*, a 0, inače;*
- $I_v(\exists x\ A)$ *je 1 ako postoji valuacija* w *sa domenom* D *koja se eventualno razlikuje od* v *jedino u vrednosti promenljive* x *i za koju važi* $I_w(A) = 1$*, a 0, inače;*
- $I_v(\forall x\ A)$ *je 0 ako postoji valuacija* w *sa domenom* D *koja se eventualno razlikuje od* v *jedino u vrednosti promenljive* x *i za koju važi* $I_w(A) = 0$*, a 1, inače.*

Definicija 16 (Model, zadovoljivost i valjanost) *Neka je* I_v *interpretacija određena* $\mathcal{L}$*-strukturom* $\mathcal{D}$ *sa domenom* D *i valuacijom* v *nad domenom* D*. Formula* A *je tačna u interpretaciji* I_v *ako važi* $I_v(A) = 1$*.* $\mathcal{L}$*-struktura* $\mathcal{D}$ *sa valuacijom* v *se naziva* modelom *formule* A*, što se označava* $(\mathcal{D}, v) \models A$*. Formula je* zadovoljiva *ako ima model, a* nezadovoljiva *ili* kontradiktorna *ako ga nema. Ako važi* $(\mathcal{D}, v) \models A$ *za svaku* $\mathcal{L}$*-strukturu* $\mathcal{D}$ *sa domenom* D *i valuaciju* v *nad domenom* D*, formula* A *se naziva* valjanom*. Ako formula nije valjana, onda je* poreciva*. Skup formula je* zadovoljiv *ako postoji model koji zadovoljava svaku formulu iz tog skupa. U suprotnom, skup formula je* kontradiktoran*.*

Definicija 17 (Logička posledica i logička ekvivalentnost) *Neka je* Γ *skup formula i neka je* A *formula nad signaturom* $\mathcal{L}$*. Formula* A *se naziva* logičkom posledicom *skupa* Γ*, što se označava* $\Gamma \models A$*, ukoliko za svaki model* $(\mathcal{D}, v)$ *važi* $(\mathcal{D}, v) \models A$ *ako važi* $(\mathcal{D}, v) \models \Gamma$*. Formule* A *i* B *su* logički ekvivalentne *ako važi* $A \models B$ *i* $B \models A$*.*

Definicija 18 (Supstitucija u termu) Supstitucija *promenljivih* $x_1, \ldots, x_n$ *termovima* $t_1, \ldots t_n$ *u termu* t *je funkcija* $t[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] : \mathcal{T}_\mathcal{L} \times V^n \times \mathcal{T}_\mathcal{L}^n \to \mathcal{T}_\mathcal{L}$ *i definiše se na sledeći način:*

- *ako je* t *simbol konstante, onda je* $t[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = t$*;*
- *ako je* $t = x_i$ *za* $i = 1, \ldots, n$*, onda je* $t[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = t_i$*;*
- *ako je* $t = y$*, gde je* $y \neq x_i$ *za* $i = 1, \ldots, n$*, onda je* $t[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = t$
- *ako je* $t = f(t'_1, \ldots, t'_m)$*, onda je* $t[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = f(t'_1[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n], \ldots, t'_m[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n])$

Definicija 19 (Supstitucija u formuli) Supstitucija *promenljivih* $x_1, \ldots, x_n$ *termovima* $t_1, \ldots, t_n$ *u formuli* A *je funkcija* $A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] : \mathcal{F}_\mathcal{L} \times V^n \times \mathcal{T}_\mathcal{L}^n \to \mathcal{F}_\mathcal{L}$ *i definiše se na sledeći način:*

- $\top[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = \top$;
- $\bot[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = \bot$;
- *ako je* $A = p(t'_m, \ldots, t'_m)$, *onda je* $A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = p(t'_1[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n], \ldots, t'_n[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n])$
- $(\neg A)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = \neg(A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n])$
- $(A \wedge B)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] \wedge B[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$
- $(A \vee B)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] \vee B[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$
- $(A \Rightarrow B)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] \Rightarrow B[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$
- $(A \Leftrightarrow B)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = A[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] \Leftrightarrow B[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$
- $(\forall x_i\ A)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = (\forall x_i\ A)$ *za* $i = 1, \ldots, n$
- $(\exists x_i\ A)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = (\exists x_i\ A)$ *za* $i = 1, \ldots, n$
- *ako je* $x_i \neq y$ *za* $i = 1, \ldots, n$, *neka je* z *promenljiva koja se ne pojavljuje ni u* $\forall y\ A$ *ni u* t_i *za* $i = 1, \ldots, n$; *tada je* $(\forall y\ A)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = \forall z\ \ A[y \mapsto z][x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$;
- *ako je* $x_i \neq y$ *za* $i = 1, \ldots, n$, *neka je* z *promenljiva koja se ne pojavljuje ni u* $\exists y\ A$ *ni u* t_i *za* $i = 1, \ldots, n$; *tada je* $(\exists y\ A)[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] = \exists z\ A[y \mapsto z][x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$;

Definicija 20 (Kompozicja supstitucija) *Za supstitucije* $\phi = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ *i* $\lambda = [y_1 \mapsto s_1, \ldots, y_n \mapsto s_n]$, kompozicija supstitucija $\phi\lambda$ *je supstitucija* $[x_1 \mapsto t_1\lambda, \ldots, x_n \mapsto t_n\lambda, y_1 \mapsto s_1, \ldots, y_n \mapsto s_n]$ *bez zamena oblika* $x_i \mapsto x_i$ *i bez zamena oblika* $y_i \mapsto s_i$, *gde je* $y_i = x_j$ *za neko* i *i* j.

Za supstituciju se kaže da je *bazna supstitucija* nad nekim promenljivim ukoliko svakoj od tih promenljivih dodeljuje neki simbol konstante. Primene supstitucija će uvek biti pisane postfiksno.

Definicija 21 (Unifikator i najopštiji unifikator) *Ako su* e_1 *i* e_2 *dve formule ili dva terma, i ako postoji supstitucija* σ *takva da važi* $e_1\sigma = e_2\sigma$, *onda kažemo da su izrazi* e_1 *i* e_2 unifikabilni *i da je supstitucija* σ unifikator *za ta dva izraza. Supstitucija* σ *je* najopštiji unifikator *za* e_1 *i* e_2 *ako svaki unifikator* τ *za* e_1 *i* e_2 *može biti predstavljen u obliku* $\tau = \sigma\mu$ *za neku supstituciju* μ.

4.1.2 Koherentni dokazi

Standardni deduktivni sistem koherentne logike se zasniva na rezonovanju unapred na osnovu baznih atoma. Može se zapisati u stilu prirodne dedukcije na sledeći način [92]:

$$\frac{p_1(\vec{a}) \wedge \ldots \wedge p_n(\vec{a})}{p_i(\vec{a})}\ \wedge E \qquad \frac{p_1(\vec{a}) \vee \ldots \vee p_n(\vec{a}) \qquad \begin{matrix}[p_1(\vec{a})] \\ \vdots \\ A\end{matrix} \quad \ldots \quad \begin{matrix}[p_n(\vec{a})] \\ \vdots \\ A\end{matrix}}{A}\ \vee E \qquad \frac{\bot}{A}\ efq$$

$$\frac{p_1(\vec{a}) \quad \ldots \quad p_n(\vec{a}) \quad p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v}) \Rightarrow \exists \vec{y}_1\ Q_1(\vec{v}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ Q_m(\vec{v}, \vec{y}_m)}{Q_1(\vec{a}, \vec{w}_1) \vee \ldots \vee Q_m(\vec{a}, \vec{w}_m)}\ ax$$

gde je $\vec{a}$ niz konstanti, a $\vec{w}_j$ (za $1 \leq j \leq m$) su nizovi novih simbola konstanti, takozvanih *svedoka*, kojima se proširuje signatura. U zapisu se podrazumeva da su slobodne promenljive univerzalno kvantifikovane.

Koherentna formula

$$p_1(\vec{v}) \wedge \ldots \wedge p_n(\vec{v}) \Rightarrow \exists \vec{y}\ Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v}, \vec{y})$$

je *teorema* neke teorije zadate skupom aksioma $\mathcal{AX}$, ukoliko se iz aksioma $\mathcal{AX}$ i pretpostavki teoreme instanciranih konstantama koje se ne javljaju u polaznoj signaturi $p_1(\vec{a}), \ldots, p_n(\vec{a})$, primenom datih pravila mogu izvesti svi konjunkti formule $Q_j(\vec{a}, \vec{w})$ za neko j $(1 \leq j \leq m)$ i za neki niz konstanti $\vec{w}$.

Postoji više procedura dokazivanja koje se zasnivaju na ovom sistemu [10, 92]. One su u stanju da generišu *objektne dokaze*, odnosno dokaze opisane do nivoa aksioma i primena pravila deduktnivnog sistema, kako u formalnim jezicima interaktivnih dokazivača kao što su Isabelle [78] i Coq [25], tako i u pirodnom jeziku.

Dobar primer teorije za koju je pogodno koristiti koherentnu logiku je euklidska geometrija. Veliki delovi aksiomatskih sistema Hilberta i Tarskog, kao i relevantna tvrđenja se vrlo lako, često i bez ikakvih intervencija, zapisuju u koherentnoj logici. Kao ilustracija primene koherentnih dokazivača u dokazivanju geometrijskih teorema, dat je sledeći izlaz dokazivača `ArgoCLP` [92]. Izlaz uključuje formulaciju tvrđenja i njegov dokaz na prirodnom jeziku, za tvrđenje koje je u polaznom obliku zadato u formalnom jeziku.

Primer 5 (Preuzet iz rada [92]) *Assuming that $p \neq q$, and $q \neq r$, and the line p is incident to the plane α, and the line q is incident to the plane α, and the line r is incident to the plane α, and the lines p and q do not intersect, and the lines q and r do not intersect, and the point A is incident to the plane α, and the point A*

is incident to the line p, and the point A is incident to the line r, show that $p = r$.

Proof:

Let us prove that $p = r$ *by reductio ad absurdum.*

1. *Assume that* $p \neq r$.
    2. *It holds that the point A is incident to the line q or the point A is not incident to the line q (by axiom of excluded middle).*
    3. *Assume that the point A is incident to the line q.*
        4. *From the facts that* $p \neq q$, *and the point A is incident to the line p, and the point A is incident to the line q, it holds that the lines p and q intersect (by axiom ax_D5).*
        5. *From the facts that the lines p and q intersect, and the lines p and q do not intersect we get a contradiction.*

           Contradiction
    6. *Assume that the point A is not incident to the line q.*
        7. *From the facts that the lines p and q do not intersect, it holds that the lines q and p do not intersect (by axiom ax_nint_l_l_21).*
        8. *From the facts that the point A is not incident to the line q, and the point A is incident to the plane* α, *and the line q is incident to the plane* α, *and the point A is incident to the line p, and the line p is incident to the plane* α, *and the lines q and p do not intersect, and the point A is incident to the line r, and the line r is incident to the plane* α, *and the lines q and r do not intersect, it holds that* $p = r$ *(by axiom ax_E2).*
        9. *From the facts that* $p = r$, *and* $p \neq r$ *we get a contradiction.*
        10. *Contradiction.*

Therefore, it holds that $p = r$.
This proves the conjecture.

Pored euklidske geometrije, važan primer primene dokazivanja u koherentnoj logici je automatizacija dokaza Hesenbergove teoreme [11]. Ovaj dokaz nije sproveden potpuno automatski, već uz pomoć čoveka tako što su uočene leme koje je bilo moguće automatski dokazati i pomoću kojih je bilo moguće automatski dokazati Hesenbergovu teoremu. Iako se ne radi o potpunoj automatizaciji, ovaj pristup dokazivanju teorema je vrlo značajan jer omogućava čoveku da se skoncentriše na

zanimljive i inventivne delove dokaza, dok se računaru prepušta da sprovede delove tehničkog karaktera.

4.1.3 Svojstva koherentne logike

Koherentna logika je prvi put definisana od strane Skolema, početkom prošlog veka. U prethodnih 15 do 20 godina, ponovo postaje popularna zahvaljujući svojim svojstvima, značajnim u domenu formalizacije matematičkog znanja [10, 32, 11, 92]. Svaka formula logike prvog reda se može prevesti u logički ekvivalentan skup formula koherentne logike bez primene skolemizacije ili razbijanja formule do klauzalne forme. Stoga, koherentna forma se može smatrati normalnom formom za logiku prvog reda, poput klauzalne forme. Međutim, osnovni kvalitet koherentne logike je što njena izražajnost, koja se ogleda u prisustvu egzistencijalni kvantifikatora i u implikacijskoj formi, omogućava da za razne teorije ili pojedinačne teoreme, zapis u koherentnoj formi bude vrlo blizak ili čak identičan polaznom zapisu. Rezonovanje unapred, predstavlja prirodan, saglasan i potpun metod dokazivanja u koherentnoj logici [10]. Prethodna dva svojstva omogućavaju generisanje čitljivih dokaza — dokaza izraženih na jeziku koji je blizak jeziku polaznog problema, a izvedenih primenama aksioma na način svojstven čoveku.

Svodljivost problema provere valjanosti u logici prvog reda na proveru valjanosti u koherentnoj logici, pokazuje da provera valjanosti u koherentnoj logici nije odlučiv problem [9].

Rezonovanje unapred u koherentnoj logici je intuicionističko. Usled nepostojanja negacije u koherentnoj logici, isključenje trećeg nije moguće izraziti. Stoga su dokazi u koherentnoj logici konstruktivni. Naravno, ova konstruktivnost je od značaja samo ukoliko u prevođenju problema iz polazne forme u koherentnu formu nije urađena ni jedna transformacija klasične logike. Međutim, klasične transformacije su u opštem slučaju sastavni deo svakog prevođenja iz logike prvog reda u koherentnu logiku [83]. Ukoliko se u zapisu polaznog problema koristi negacija, ona se može simulirati u okviru koherentne logike. Ako je A atomička formula, $\neg A$ se može predstaviti u obliku $A \Rightarrow \bot$, ali se ovo ne može uraditi u slučaju proizvoljne neatomičke formule jer to koherentna forma ne dozvoljava. Uobičajeno, sastavni deo prevođenja formula koje uključuju negaciju u koherentnu logiku je uvođenje dodatnih predikatskih simbola koji odgovaraju negaciji polaznog simbola. Tako se za predikatski simbol p uvodi novi simbol $\overline{p}$, koji zamenjuje $\neg p$, i dodaju se sledeće aksiome: $\forall \vec{x}\ (p(\vec{x}) \wedge \overline{p}(\vec{x}) \Rightarrow \bot)$ i $\forall \vec{x}\ (p(\vec{x}) \vee \overline{p}(\vec{x}))$. Uvođenje druge aksiome je nepoželjno jer je po njoj moguće granati bez potrebe da se prethodno zadovolje neke pretpostavke. Nju je u nekim situacijama moguće izostaviti [83].

Koherentna logika ima svojstvo *kompaktnosti* — za svaku formulu za koju postoji klasični dokaz, postoji i intuicionistički dokaz [10].

4.2 Sistemi za dokazivanje u koherentnoj logici

Prvi dokazivač teorema zasnovan na koherentnoj logici je `Euklid` [47]. `Euklid` nije opšti dokazivač za koherentnu logiku, već je specijalizovan za geometriju i sistem aksioma je zadat kroz implementaciju i ne može se menjati. Rezonovanje koje `Euklid` koristi je bazno rezonovanje unapred sa jednostavnim vraćanjem u pretrazi na prvo prethodno grananje u dokazu. Ovaj dokazivač je u stanju da pruži dokaze na prirodnom jeziku. Poslužio je kao inspiracija za kasnije dokazivače za koherentnu logiku, uključujući i dokazivač `Calypso` zasnovan na sistemu pravila koji će biti predložen u ovoj glavi. Neke ideje, prvi put upotrebljene u dokazivaču `Euklid`, vezane za redosled primene aksioma i ograničavanje dubine pretrage, su korišćene i u kasnije predloženim dokazivačima.

`CL` je opšti dokazivač za koherentnu logiku koji se može primeniti na proizvoljno tvrđenje i proizvoljan aksiomatski sistem [10]. Koristi bazno rezonovanje unapred i, kao i `Euklid`, prilikom vraćanja u pretrazi uvek se vraća na prvo prethodno grananje. Uprkos jednostavnom načinu pretrage, ispostavilo se da postoje koherentna tvrđenja na kojima je `CL` neuporedivo efikasniji od najboljih dokazivača za logiku prvog reda. Ovaj rezultat je značajno ohrabrenje za razvoj specijalizovanih dokazivača za koherentnu logiku. `CL` može da dâ izlaz u jeziku interaktivnog dokazivača Coq [25]. Dokazivač teorema napravljen po ugledu na `CL` je ugrađen i u sistem Isabelle.

`Geolog` je dokazivač zasnovan na istim principima kao dokazivač `CL`, ali ima grafički korisnički interfejs koji omogućava izvršavanje dokazivača korak po korak i iscrtavnje stabala dokaza.

Prvi dokazivač teorema za koherentnu logiku, ili preciznije za fragment logike prvog reda vrlo blizak koherentnoj logici, koji implementira učenje lema je `Geo` [24]. Zahvaljujući učenju lema, prilikom vraćanja u pretrazi, primena naučene leme u svakom od prethodnih grananja koja nisu bila bitna za zatvaranje grane pretrage u kojoj je lema naučena, dovodi do toga da se pretraga nastavlja od prvog prethodnog grananja koje je relevantno za zatvaranje te grane pretrage. Rezultujuće ponašanje dokazivača je vrlo slično ponašanju koje bi se dobilo korišćenjem tehnike povratnih skokova. `Geo` je pre svega sistem za nalaženje minimalnih konačnih modela, ali se, kao takav, može upotrebiti i za dokazivanje teorema pobijanjem. Na takmičenju dokazivača teorema CASC[1] je nagrađen u kategoriji sistema za nalaženje modela. Re-

[1] `http://www.cs.miami.edu/~tptp/CASC/`

zonovanje koje `Geo` sprovodi je takođe bazno rezonovanje unapred i učenje lema koje je formulisano se ne može primeniti prilikom rezonovanja koje nije bazno. Takođe, forma naučenih lema je ograničena — naučene leme na desnoj strani implikacije mogu imati samo $\perp$. `Geo` ne pruža objektne dokaze tvrđenja.

`ArgoCLP` je dokazivač teorema za koherentnu logiku inspirisan sistemom `Euklid` [92] koji može da se primeni na proizvoljnu koherentnu teoriju. Koristi bazno rezonovanje unapred sa jednostavnim vraćanjem u pretrazi. U stanju je da pruži objektne dokaze u jeziku Isar [97], koji se koristi u interaktivnom dokazivaču Isabelle i dokaze na prirodnom jeziku. Uspešno je primenjen u dokazivanju teorema euklidske geometrije u Hilbertovom aksiomatskom sistemu [92], a u toku je njegova primena i na dokazivanje teorema u aksiomatskom sistemu Tarskog.

Dokazivač za koherentnu logiku `clp` zasnovan je na baznom rezonovanju unapred i implementira tehniku povratnih skokova [40]. Takođe, implementira i algoritam koji omogućava da se pokušaji unifikacije pretpostavki aksiome sa istim atomima ne ponavljaju, što je posao u koji čini veliki deo ukupnog vremena prilikom dokazivanja teorema rezonovanjem unapred. U eksperimentalnoj evaluaciji u kojoj su korišćene koherentne instance, ovaj dokazivač se pokazao efikasnijim od dokazivača `CL`, podjednako efikasnim kao dokazivač `Geolog`, ali manje efikasnim od dokazivača `Geo` [40]. Takođe, na istim instancama, pokazao se efikasnijim od dokazivača za opštu logiku prvog reda `Vampire` i `Eprover` [40].

Pored dokazivača za koherentnu logiku, za sistem koji će biti predložen relevantni su i postojeći sistemi zasnovani na DPLL proceduri. Jedan takav sistem je razvijen za fragment logike prvog reda poznat kao *efektivno iskazna logika* ili Bernejz-Šenfinkelov fragment [82]. Efektivno iskazna logika predstavlja klauzalni fragment logike prvog reda bez egzistencijalnih kvantifikatora i funkcijskih simbola osim konstanti. Ovaj fragment je odlučiv. *Račun evolucije modela* predstavlja podizanje moderne DPLL procedure u klauzalni fragment logike prvog reda [7] i osnovu za dokazivač `Darwin` [6]. U ovom pristupu prilikom transformacije u klauzalnu formu, razbija se struktura polazne teorije i uvode se Skolemove funkcije. U dokazivanju svojstava ovog računa, povratni skokovi se ne tretiraju kao deo računa, već kao implementaciona tehnika. Ovo je važno jer je, kao što je pomenuto u glavi 2.5, najveći deo težine dokaza potpunosti DPLL procedure vezan za prisustvo povratnih skokova.

4.3 Izazovi u dokazivanju teorema u koherentnoj logici

Kao što je opisano u prethodnom poglavlju, osnovu većine dokazivača za koherentnu logiku čini bazno rezonovanje unapred sa jednostavnim vraćanjem unazad na prvo prethodno grananje u dokazu [10, 92], ali postoje i dokazivači koji su u stanju da preskoče nerelevantna grananja u dokazu [40] ili, još važnije, da po zatvaranju svake grane nauče lemu koja uopštava razloge za zatvaranje te grane [24]. Neki od dokazivača su u stanju da pruže objektne dokaze za tvrđenja koja dokazuju [10, 92, 40]. Međutim, ni jedan od postojećih dokazivača ne objedinjuje sve ove elemente. Ovakvo stanje stvari u oblasti ne predstavlja slučajnost. Neke od ovih elemenata je teško formulisati u koherentnoj logici, a neke je teško uskladiti. U nastavku su izloženi neki od najvažnijih izazova vezanih za izgradnju dokazivača za koherentnu logiku.

Rezonovanje prvog reda. Bazno rezonovanje ima očigledne mane vezano za kompaktnost reprezentacije znanja do kog se dolazi u toku rada dokazivača. Na primer, u slučaju aksiomatskog sistema koji uključuje aksiome $\forall x\ p(x)$ i $\forall x\ (p(x) \Rightarrow q(x))$, za svaki simbol konstante a je moguće izvesti instancu prve aksiome $p(a)$, pa primenom druge aksiome izvesti $q(a)$. Kako broj simbola konstanti može biti veliki, i broj izvedenih atoma može biti veliki. Dodatno, ovaj postupak mora biti ponavljan prilikom svakog uvođenja novih svedoka dobijenih eliminacijom egzistencijalnog kvantifikatora. Pri ovakvom načinu rezonovanja, ne samo što se koristi veliki broj koraka u toku izvođenja, nego se dokazivač usporava usled česte pretrage kroz izvedene atome. Umesto toga, bilo bi poželjnije jednom primenom druge aksiome na prvu aksiomu izvesti zaključak $\forall x\ q(x)$. U drugom primeru, ako su date formule $\exists x\ p(x)$ i $\forall x\ (p(x) \Rightarrow \bot)$, bilo bi poželjno ustanoviti kontradikciju bez potrebe za eliminacijom egzistencijalnog kvantifikatora. Korist od takve mogućnosti ne bi bila zanemarljiva zato što u nekim sistemima korišćenje atoma sa novouvedenim konstantama nije odmah dozvoljeno, pa detekcija kontradikcije može biti dodatno odložena i nakon eliminacije egzistencijalnog kvantifikatora. Bilo bi poželjno detektovati kontradikciju i u slučaju sledećih formula $\exists x\ p(x) \vee \exists x\ q(x)$, $\forall x\ (p(x) \Rightarrow \bot)$ i $\forall x\ (q(x) \Rightarrow \bot)$, bez potrebe za grananjem po disjunktima i eliminacijom egzistencijalnih kvantifikatora.

Zbog opisanih problema, potrebno je omogućiti korišćenje atoma sa slobodnim promenljivim (implicitno univerzalno kvantifikovanim) i/ili proširenih egzistencijalnim kvantifikatorima. Ovakva odluka nosi sa sobom skrivene probleme vezane za

učenje lema, koji će biti pomenuti u nastavku.

Povratni skokovi i učenje lema. Povratni skokovi i učenje lema su dva najznačajnija unapređenja modernih SAT rešavača, koja su dovela do CDCL sistema pretrage. Dokazivači za koherentnu logiku koji ih implementiraju se pokazuju efikasnijim od onih koji se zasnivaju na običnom rezonovanju unapred [40, 24]. Međutim, učenje lema za logiku vrlo blisku koherentnoj, koje je implementirano u sistemu `Geo` [24] je primenljivo je samo u slučaju baznog rezonovanja unapred, što je značajno ograničenje. Takođe, kao što je pomenuto, leme koje `Geo` uči su ograničene forme što je poželjno izbeći.

U slučaju odluke da se dozvoli rezonovanje prvog reda, učenje lema nosi određene izazove i dovodi do potrebe za proširivanjem fragmenta logike u kojem se rezonuje. Na primer, ukoliko se korišćenjem aksioma $\forall x\ p(x)$ i $\forall x\ (p(x) \Rightarrow q(x))$ izvede zaključak $\forall x\ q(x)$, lema koja odgovara ovom zaključivanju je $(\forall x\ p(x)) \Rightarrow (\forall y\ q(y))$. Ova lema ne pripada koherentnoj logici zbog prisustva univerzalnog kvantifikatora u levoj strani implikacije. Međutim, ovakvo proširenje je neophodno, jer je potrebno da lema izrazi sve pretpostavke pod kojima se došlo do izvedenih zaključaka. Stoga, postoji potreba da se, zarad učenja lema, fragment logike u kojem se rezonuje proširi tako da se u levoj strani implikacije mogu pojavljivati onakve potformule kakve se mogu koristiti prilikom primene aksiome u rezonovanju unapred. Prilikom uopštavanja baznog rezonovanja za koherentnu logiku, dovoljno je razmatrati vrste potformula koje se mogu javiti u desnoj strani implickacije, pošto se prilikom rezonovanja unapred mogu izvoditi samo takve formule i samo one mogu služiti za primenu aksioma. Na taj način se može uočiti sledeća korespodencija vrste formula na koje se prilikom rezonovanja unapred mogu primenjivati aksiome i fragmenata logike koji su potrebni za učenje lema:

- U slučaju da se u rezonovanju unapred na osnovu koherentnih aksioma koriste samo bazni atomi, naučene leme se mogu predstaviti u okviru koherentne logike.

- U slučaju da se u rezonovanju unapred za primenu koherentnih aksioma koriste bazni atomi i formule oblika $\forall \vec{x}\ p(\vec{a}, \vec{x})$ i formule oblika $\exists \vec{x}\ p(\vec{a}, \vec{x})$, gde u oba slučaja $\vec{a}$ sadrži samo konstante, naučene leme se mogu predstaviti u sledećem obliku

$$\forall \vec{v}\ (\forall \vec{x}\ p_1(\vec{v}, \vec{x}) \wedge \ldots \wedge \forall \vec{x}\ p_n(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v}, \vec{y})),$$

pa je stoga potrebno proširiti razmatrani fragment logike na fragment koji se satoji iz ovakvih formula.

- U slučaju da se pored formula pomenutih oblika dozvoli primena aksioma i na formule oblika $\forall x \exists y\ p(\vec{a}, \vec{x}, \vec{y})$, naučene leme se mogu predstaviti u sledećem obliku

$$\forall \vec{v}\ (\forall \vec{x}\ \exists \vec{z}\ (\ p_1(\vec{v}, \vec{x}, \vec{z}) \wedge \ldots \wedge p_n(\vec{v}, \vec{x}, \vec{z})) \Rightarrow \exists \vec{y}\ Q_1(\vec{v}, \vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v}, \vec{y})),$$

pa je stoga potrebno proširiti razmatrani fragment logike na fragment koji se satoji iz ovakvih formula.

Učenje lema i čitljivi dokazi. Dokazivači za koherentnu logiku koji ne koriste učenje lema mogu da proizvedu objektni dokaz za dokazano tvrđenje memorisanjem svih izvedenih koraka prilikom rezonovanja unapred i njihovim prevođenjem u neki formalni ili prirodni jezik [92, 40, 10]. Neki od ovih dokazivača su u stanju da pruže dokaz iz kojeg su uklonjeni svi suvišni koraci, bilo postprocesiranjem [92], bilo pomoću mehanizmima koji se za to staraju u toku rada [40]. U slučaju sistema koji koristi učenje lema, postavlja se pitanje na koji način naučene leme treba da figurišu u objektnom dokazu. Naučene leme mogu biti dugačke u zapisu i čoveku nezanimljive ili teško razumljive. U slučaju da se u toku dokazivanja nauči puno takvih lema, praćenje dokaza u kojem one figurišu može biti vrlo naporno za čoveka. S druge strane, eliminacija lema iz dokaza u najgorem slučaju dovodi do eksponencijalnog rasta dokaza, što ga ponovo čini teškim za praćenje. Naravno, dokazivač koji ne koristi učenje lema bi u startu proizveo preveliki dokaz, tako da mogućnost korišćenja lema u dokazu nije mana, već prednost koju treba iskoristiti. Osnovno pitanje je kada lemu treba zadržati u dokazu, a kada je iz njega treba eliminisati.

Implementacione tehnike. Prilikom izgradnje modernih dokazivača, posvećuje se velika pažnja implementacionim tehnikama. Zahvaljujući njima, može se značajno dobiti na efikasnosti. U slučaju SAT rešavača, tehnika dva nadgledana literala koja služi za efikasno nalaženje literala koji se mogu propagirati je dovela do velikog ubrzanja rešavača [71]. Analogan problem u koherentnoj logici je problem unifikacije pretpostavki aksiome sa izvedenim formulama, u čemu se provodi najveći deo vremena u toku rada dokazivača [40]. Jedan algoritam za unifikaciju pretpostavki aksiome koji se koristi kod dokazivača `clp` je Rete algoritam formulisan za bazno

rezonovanje [40] koji omogućava čuvanje parcijalnih unifikatora za pretpostavke aksiome, zahvaljujući čemu se u kasnijim pokušajima unifikovanja ne pokušava unifikovanje sa istim atomima, već samo sa atomima koje mogu dovesti do proširivanja parcijalnog unifikatora, što smanjuje količinu posla u unifikaciji pretpostavki aksiome. Konstrukcija i unapređivanje ovakvih algoritama je od velikog značaja za efikasnost dokazivača.

Formulisanje heuristika. Moderni SAT rešavači svoju efikasnost u značajnoj meri duguju pametnim heuristikama, pre svega heuristici VSIDS za izbor promenljive prilikom odlučivanja [71], ali i heuristici čuvanja polariteta [80]. Formulisanje CDCL sistema pretrage za koherentnu logiku bi omogućilo da se ovakve heuristike ili njihovi analogoni formulišu i u tom domenu.

4.4 Novi CDCL sistem za dokazivanje u koherentnoj logici

Osnovna svrha predlaganja CDCL pristupa za dokazivanje u koherentnoj logici je formulisanje teorijskog okvira za izgradnju dokazivača za koherentnu logiku na osnovama koje su se pokazale vrlo efikasnim u rešavanju SAT problema — apstraktnom sistemu pravila za CDCL pretragu. Pristup koji se predlaže ima tri važna oslonca:

Pogodnost koherentne logike. Koherentna logika ima nekoliko poželjnih svojstava razmatranih u poglavlju 4.1. Njen značaj je pre svega u njenoj izražajnosti i mogućnosti generisanja čitljivih dokaza — dokaza nastalih na osnovu aksiomatskog sistema čija je forma ista ili bliska originalnoj umesto da je razbijena do nivoa klauzalne forme ili da se koriste Skolemove funkcije. Takođe, ovi dokazi mogu biti zapisani i u formalnim jezicima koje koriste interaktivni dokazivači. Eksploatacija ovih prednosti je važan faktor koji treba uzeti u obzir prilikom dizajna sistema pravila.

Praktična unapređenja u rešavanju SAT problema. U toku prethodne dve decenije napravljen je veliki napredak u rešavanju SAT problema. Predloženo je nekoliko algoritamskih i implementacionih poboljšanja, zahvaljujući kojima moderni SAT rešavači mogu da se nose sa industrijskim instancama koje uključuju stotine hiljada klauza i desetine hiljada promenljivih. Predloženi pristup bi trebalo da omogući prenos ovakvih tehnologija u dokazivanje u koherentnoj logici.

Teorijski napredak u rešavanju SAT problema. SAT rešavači su precizno opisani apstraktnim sistemima pravila, što omogućava njihovu temeljnu matematičku analizu. Kao što je ukazano u poglavlju 2.5, njihova ispravnost je dokazana prvo neformalno [72, 53], a potom i formalno uz pomoć interaktivnih dokazivača [65, 68]. Ovi rezultati su pomogli u razdvajanju različitih koncepata koje SAT rešavači koriste, a koji su često izmešani u tipičnim optimizovanim implementacijama. Dodatno, oni olakšavaju dublje razumevanje rada SAT rešavača. Ideje korišćene u apstraktnim sistemima pravila za rešavanje SAT problema su korišćene u dizajnu i opisu apstraktnog sistema pravila za koherentnu logiku i za dokazivanje njegovih svojstava.

4.4.1 Temelji CDCL dokazivanja u koherentnoj logici

Apstraktni sistem pravila za koherentnu logiku je uopštenje apstraktnog sistema pravila za iskaznu logiku koji je dat u poglavlju 2.3 i predstavlja osnovu za proceduru poluodlučivanja za koherentnu logiku. Ovi sistemi su formulisani u istom duhu. Pravila za iskaznu logiku imaju svoje analogone u sistemu za koherentnu logiku u nešto komplikovanijem obliku. U sistemu za koherentnu logiku nedostaju pravila otpočinjanja iznova i zaboravljanja. Otpočinjanje iznova nije uključeno jer je u slučaju ispitivanja zadovoljivosti iskaznih formula njegov značaj u slučaju nezadovoljive ulazne instance značajno manji nego u slučaju zadovoljive instance, a upravo je prvi slučaj taj koji odgovara dokazivanju teoreme, što je svrha predloženog sistema, dok drugi odgovara nalaženju kontramodela. Zaboravljanje nije pogodno koristiti kada je cilj generisanje dokaza, jer se pri zaboravljanju mogu izgubiti i leme neophodne za generisanje objektnih dokaza. Ipak, dodavanje bilo kog od ova dva pravila predstavlja legitimno proširenje predloženog sistema.

Predloženi sistem će uključivati klasično rezonovanje i objektni dokazi koji se na osnovu ovog sistema budu proizvodili mogu uključivati isključenje trećeg.

U ostatku teksta, ukoliko nije drugačije naglašeno, biće korišćenja konvencija da $p(\vec{t}_1, \ldots, \vec{t}_n)$ označava atom koji uključuje neke (možda nijedan) od elemenata iz nizova $\vec{t}_i$ $(i = 1, \ldots, n)$ promenljivih i/ili konstanti kao svoje argumente. Pri tome arnost simbola p ne mora biti n.

Definicija 22 (Relacija $\sim$) *Neka za atom l, l_i označava njegov i-ti argument. Neka su data dva atoma $l = p(\vec{t}_1, \ldots, \vec{t}_n)$ i $l' = p(\vec{t}'_1, \ldots, \vec{t}'_n)$. Neka je I_i skup indeksa takvih da važi $j \in I_i$ ako i samo ako $l_j \in \vec{t}_i$. Neka je I'_i skup indeksa takvih da važi $j \in I'_i$ ako i samo ako $l'_j \in \vec{t}'_i$. Kažemo da l i l'* odgovaraju *jedan drugom i pišemo $l \sim l'$ u koliko za svako $i \in \{1, \ldots, n\}$ važi $I_i = I'_i$ i postoji funkcija f_i takva da za*

svako $j \in I_i$ *važi* $l_j \overset{f_i}{\mapsto} l'_j$ *ili za svako* $j \in I'_i$ *važi* $l'_j \overset{f_i}{\mapsto} l_j$.

Na primer, neka se razmatraju zapisi $p(\vec{t}_1, \vec{t}_2)$ i $p(\vec{t}'_1, \vec{t}'_2)$, pri čemu je $\vec{t}_1 = x_1\ x_2$, $\vec{t}_2 = y_1\ y_2$, $\vec{t}'_1 = c$ i $\vec{t}'_2 = u$. Dve konkretne formule koje dati zapisi mogu označavati su $p(x_1, y_1, x_2, y_2)$ i $p(c, u, c, u) \Rightarrow \bot$.

U svim narednim formulama, smatra se da su slobodne promenljive implicitno univerzalno kvantifikovane.

U CDCL sistemu za SAT problem, pojam literala ima centralno mesto. Da bi se formulisao CDCL sistem za koherentnu logiku, potrebno je definisati relevantne elementarne formule koje će imati ulogu literala u tom sistemu. Te formule će biti izabrane tako da je moguće rezonovanje prvog reda.

Definicija 23 (Kvantifikovani literal) Pozitivni kvantifikovani literal *je formula oblika* $p(\vec{v})$ *ili formula oblika* $\exists\vec{y}\ p(\vec{y})$, *gde je* p *predikatski simbol i* $\exists\vec{y}\ p(\vec{y})$ *nema slobodnih promenljivih.* Negativni kvantifikovani literal *je formula oblika* $p(\vec{v}) \Rightarrow \bot$ *ili formula oblika* $(\forall\vec{x}\ p(\vec{x})) \Rightarrow \bot$ *gde je* p *predikatski simbol i* $\forall\vec{x}\ p(\vec{x})$ *nema slobodnih promenljivih.* Kvantifikovani literal *ili* k-literal *je pozitivni ili negativni kvantifikovani literal. Skup kvantifikovanih literala nad signaturom* Σ *se obeležava* $\mathcal{QL}(\Sigma)$.

Formule oblika $\forall\vec{x}\ p(\vec{v}, \vec{x})$ *i* $\exists\vec{y}\ p(\vec{v}, \vec{y})$ *su* pozitivni prošireno kvantifikovani literali. *Formula oblika* $(\forall\vec{x}\ p(\vec{v}, \vec{x})) \Rightarrow \bot$ *je* negativni prošireno kvantifikovani literal. Prošireni kvantifikovani literal *ili* pk-literal *je pozitivni ili negativni prošireno kvantifikovani literal.*

Umesto $l \Rightarrow \bot$, *može se pisati* $\overline{l}$. *Za skup literala* $S = \{l_1, \ldots, l_n\}$, $\overline{S}$ *označava skup* $\{\overline{l}_1, \ldots, \overline{l}_n\}$.

Primer 6 *U ovom i narednim primerima, na konstante* $c^1, c^2, \ldots$ *će se referisati oznakama* $a, b, \ldots$. *Smatraće se da važi* $\Pi = \{p, q, r, s, p', q', r', s', p'', \ldots\}$ *gde je* $ar(p) = ar(q) = ar(r) = 2$ *i* $ar(s) = 1$, *a arnosti predikatskih simbola sa oznakom* $'$ *je ista kao i arnost odgovarajućeg predikatskog simbola bez te oznake.*

Formule $p(a, b)$, $p(a, x)$ *i* $\exists y\ p(a, y)$ *su kvantifikovani atomi. Formule* $\forall x\ p(a, x)$, $\forall x\ p(v, x)$, *i* $\exists y\ p(x, y)$ *su prošireni kvantifikovani atomi (usled prisustva univerzalnog kvantifikatora u prva dva slučaja i egzistencijalnog kvantifikatora u trećem slučaju. Formule* $\overline{p(a, b)}$, $\overline{p(a, x)}$, $\overline{p(x, y)}$ *su negativni kvantifikovani literali, dok je formula* $\overline{\forall x\ p(x, y)}$ *negativni prošireno kvantifikovani literal.*

Prema razmatranju iz poglavlja 4.3, izbor elementarnih formula koje se koriste u rezonovanju, povlači i izbor fragmenta koji je dovoljan da obezbedi učenje lema.

Tamo predloženi fragment koji odgovara kvantifikovanim literalima kao elementarnim formulama je

$$\forall \vec{x}\ p_1(\vec{v},\vec{x}) \wedge \ldots \wedge \forall \vec{x}\ p_n(\vec{v},\vec{x}) \Rightarrow \exists \vec{y}\ Q_1(\vec{v},\vec{y}) \vee \ldots \vee \exists \vec{y}\ Q_m(\vec{v},\vec{y})$$

gde su Q_i, za $i = 1, \ldots, m$, konjunkcije pk-literala.

Ovaj izbor elementarnih formula i fragmenta predstavlja kompromis između povećanja izražajnosti u rezonovanju i tehničke komplikovanosti pri izgradnji sistema za rezonovanje u datom fragmentu.

Iako će se u nastavku intenzivno koristiti obe vrste literala, osnovna razlika u ulogama k-literala i pk-literala je da se samo na osnovu izvedenih k-literala može vršiti rezonovanje, odnosno da se samo na osnovu njih mogu propagirati pk-literali i ustanovljavati kontradikcije, što je u skladu sa upravo učinjenim izborom fragmenta. U nastavku će se uvek birati najmanje opšt izraz koji je primenljiv. Odnosno, ukoliko u nekom kontekstu figuriše k-literal, insistiraće se na tom izrazu umesto da se upotrebi opštiji izraz pk-literal.

Kako CDCL pristup nije pogodan za rezonovanje koje uključuje disjunkcije konjunkcija, već samo disjunkcije literala, forma dozvoljene formule se može dodatno modifikovati uvođenjem novih predikata kojima se zamenjuju konjunkcije u desnoj strani formula, pri čemu je potrebno uvesti dodatne aksiome kako bi se izrazila veza između konjunkcija i predikata koji ih zamenjuju:

$$\forall \vec{x}\ p_1(\vec{v},\vec{x}) \wedge \ldots \wedge \forall \vec{x}\ p_n(\vec{v},\vec{x}) \Rightarrow \exists \vec{y}\ q_1(\vec{v},\vec{y}) \vee \ldots \vee \exists \vec{y}\ q_m(\vec{v},\vec{y}), \tag{4.1}$$

pri čemu su $q_1, \ldots, q_m$ novouvedeni predikatski simboli koji se ne javljaju u signaturi i pri čemu se dodaju aksiome $q_i(\vec{v},\vec{y}) \Rightarrow Q_i^j(\vec{v},\vec{y})$ za svako i $(1 \leq i \leq m)$ i j $(1 \leq j \leq |Q_i|)$ gde je $|Q_i|$ broj atoma u konjunkciji Q_i, a Q_i^j j-ti konjunkt u toj konjunkciji i aksiome $Q_i(\vec{v},\vec{y}) \Rightarrow q_i(\vec{v},\vec{y})$ za svako i $(1 \leq i \leq m)$. Uvedeni predikati mogu biti eliminisani iz objektnog dokaza na osnovu ovih aksioma. U nastavku teksta, pod koherentnom formulom, podrazumeva se formula koja ima formu (4.1).

Formule koje se razlikuju samo u imenovanju promenljivih će se smatrati jednakim. Radi jednostavnije prezentacije, skup elemenata liste L će biti obeležavan takođe sa L, a prazna lista će biti obeležavana $\emptyset$. Dodatno, skup formula u konjunkciji $\mathcal{P}$, odnosno disjunkciji $\mathcal{Q}$ će biti označavan takođe sa $\mathcal{P}$, odnosno $\mathcal{Q}$. Ako je $\mathcal{P}$ jednako $\{p_1, \ldots, p_n\}$ i ako je $\mathcal{Q}$ jednako $\{q_1, \ldots, q_n\}$, onda $\forall \vec{x}\ \mathcal{P}$ označava $\{\forall \vec{x}\ p_1, \ldots, \forall \vec{x}\ p_n\}$, a $\exists \vec{y}\ \mathcal{Q}$ označava $\{\exists \vec{y}\ q_1, \ldots, \exists \vec{y}\ q_n\}$.

U nastavku teksta podrazumeva se da za svaku koherentnu formulu $\mathcal{P} \Rightarrow \mathcal{Q}$ važi $\mathcal{P} \cap \mathcal{Q} = \emptyset$. Formule koje ovo ne zadovoljavaju se izbacuju iz skupa aksioma, a biće

pokazano da ne mogu nastati primenom pravila sistema.

Ako je $\mathcal{F}$ formula, zapis $\mathcal{F}$, u zavisnosti od konteksta, može da označava i skup pk-literala formule $\mathcal{F}$. Neka je $m : \mathcal{F} \to \mathcal{X}$ funkcija. Funkcija $m^{-1} : m(\mathcal{F}) \to P(\mathcal{F})$, gde $P(\mathcal{F})$ označava partitivni skup skupa pk-literala formule $\mathcal{F}$, je definisana kao $m^{-1}(l) = \{l' \in \mathcal{F} \mid m(l') = l\}$.

4.4.2 Koherentna rezolucija i koherentno faktorisanje

Dva osnovna pojma na kojima počiva sistem koji će biti predložen su pojam *koherentne rezolucije* i pojam *koherentnog faktorisanja*. Prvo će biti definisan pojam koherentne rezolucije. Koherentna rezolucija je inspirisana klauzalnom rezolucijom, ali je formulisana tako da se vrši direktno nad implikacijama i kao rezolventu izvodi implikaciju. Takođe, kvantifikatori se ne eliminišu već su uključeni u pravilo rezolucije. Sledeća definicija uvodi pomoć ni pojam unifikatora pozitivnih pk-literala, a potom se uvodi pojam koraka koherentne rezolucije.

Definicija 24 (Unifikator pozitivnih pk-literala) *Supstitucija λ je unifikator za pozitivne pk-literale $\forall\vec{x}\, p(\vec{v},\vec{x})$ i $p(\vec{v}',\vec{v}'')$, odnosno za pozitivne pk-literale $\exists\vec{x}\, p(\vec{v},\vec{x})$ i $p(\vec{v}',\vec{v}'')$, ukoliko važi $p(\vec{v},\vec{x}) \sim p(\vec{v}',\vec{v}'')$ i ukoliko za neko $\vec{u}$ i $\vec{w}$ važi $p(\vec{v},\vec{x})\lambda = p(\vec{u},\vec{x})$ i $p(\vec{v}',\vec{v}'')\lambda = p(\vec{u},\vec{w})$, pri čemu vaǐ $p(\vec{u},\vec{x}) = p(\vec{u},\vec{w})\mu$ gde je μ preimenovanje promenljivih $\vec{w}$.*

Primer 7 *Supstitucija $\lambda = [x \mapsto y, z \mapsto y]$ je unifikator za pk-literale $p(x,z)$ i $p(y,y)$ jer važi $p(x,z)\lambda = p(y,y)$ i $p(y,y)\lambda = p(y,y)$. Supstitucija $\lambda = [x \mapsto z]$ je unifikator za pk-literale $p(x,z)$ i $\exists y\ p(y,y)$ jer važi $p(x,z)\lambda = p(z,z)$ i $p(y,y)\lambda = p(y,y)$, pri čemu je $p(y,y) = p(z,z)\mu$, gde je $\mu = [z \mapsto y]$ preimenovanje promenljive z.*

Definicija 25 (Relacija koraka koherentne rezolucije: *ResStep***)** Pravila koherentne rezolucije *su sledeća dva pravila*

$$\frac{\mathcal{P} \Rightarrow \mathcal{Q} \cup \{\exists\vec{y}\ r(\vec{v},\vec{y})\} \quad \{r(\vec{v}',\vec{v}'')\} \cup \mathcal{P}' \Rightarrow \mathcal{Q}'}{(\mathcal{P} \cup \forall\vec{v}''\ \mathcal{P}' \Rightarrow \mathcal{Q} \cup \exists\vec{v}''\ \mathcal{Q}')\lambda} \qquad \frac{\mathcal{P}' \Rightarrow \mathcal{Q}' \cup \{r(\vec{v}',\vec{v}'')\} \quad \{\forall\vec{x}\ r(\vec{v},\vec{x})\} \cup \mathcal{P} \Rightarrow \mathcal{Q}}{(\forall\vec{v}''\ \mathcal{P}' \cup \mathcal{P} \Rightarrow \exists\vec{v}''\ \mathcal{Q}' \cup \mathcal{Q})\lambda}$$

pri čemu je λ unifikator za pk-literale $\exists\vec{y}\ r(\vec{v},\vec{y})$ i $r(\vec{v}',\vec{v}'')$ u prvom slučaju, a za pk-literale $r(\vec{v}',\vec{v}'')$ i $\forall\vec{x}\ r(\vec{v},\vec{x})$ u drugom slučaju, i pri čemu formule koje se rezolviraju nemaju zajedničkih promenljivih. Ukoliko se formula $\mathcal{R}$ može dobiti jednom primenom nekog od pravila rezolucije na formule $\mathcal{F}_1$ i $\mathcal{F}_2$ po pk-literalima l_1 iz $\mathcal{F}_1$ i l_2 iz $\mathcal{F}_2$, pri unifikatoru λ, to se zapisuje $ResStep_{\lambda}^{l_1 \mapsto l_2}(\mathcal{F}_1, \mathcal{F}_2, \mathcal{R})$.

Ukoliko u datom kontekstu preslikavanje $l_1 \mapsto l_2$ ili supstitucija λ nisu bitni, mogu se izostaviti u zapisu.

Ukoliko je $\mathcal{S}$ *skup koherentnih formula,* $ResStep_\lambda(\mathcal{F}, \mathcal{S}, \bot)$ *označava da važi* $ResStep_\lambda(\mathcal{F}, \mathcal{F}', \bot)$ *za svaku formulu* $\mathcal{F}' \in \mathcal{S}$.

Vrednost funkcije uparivanja pri rezolviranju *formula po istaknutim pk-literalima* $r : \mathcal{P} \cup \mathcal{Q} \cup \mathcal{P}' \cup \mathcal{Q}' \to \mathcal{QL}(\Sigma)$, *za pk-literal* l *je*

- $l\lambda$, *ukoliko važi* $l \in \mathcal{P}$ *ili* $l \in \mathcal{Q}$,
- $\forall \vec{v}''\ l\lambda$, *ukoliko važi* $l \in \mathcal{P}'$,
- $\exists \vec{v}''\ l\lambda$, *ukoliko važi* $l \in \mathcal{Q}'$.

Smisao funkcije uparivanja je povezivanje pk-atoma polaznih formula sa pk-atomima rezultujuće formule koji su od njih nastali. Smisao funkcije uparivanja će i u narednim definicijama biti sličan. Ova funkcija je dobro definisana zahvaljujući pretpostavci da ni jedna koherentna formula nema isti pk-literal i u levoj i u desnoj strani i pretpostavci da su skupovi promenljivih formula koje se rezolviraju disjunktni. Ukoliko su dužina niza $\vec{y}$ u prvom pravilu i dužina niza $\vec{x}'$ u drugom pravilu jednaki 0, dva pravila rezolucije su jednaka i u tom slučaju nije važno razlikovati koje je pravilo upotrebljeno. Ukoliko nije naglašeno drugačije, u daljem tekstu se pod rezolucijom podrazumeva koherentna rezolucija.

Primer 8 *Neka su date formule* $p(v,u) \Rightarrow q(v,u) \vee \exists y\ r(v,y)$ *i* $r(v',u') \wedge p'(v',u') \Rightarrow q'(v',u')$. *Primenom prvog pravila koherentne rezolucije na ove formule se može dobiti rezolventa* $p(v,u) \wedge \forall u'\ p'(v,u') \Rightarrow q(v,u) \vee \exists u'\ q'(v,u')$. *U ovom slučaju, funkcija upraivanja pri rezolviranju je data sledećim pridruživanjima* $p(v,u) \mapsto p(v,u)$, $p'(v',u') \mapsto \forall u'\ p'(v,u')$, $q(v,u) \mapsto q(v,u)$ *i* $q'(v',u') \mapsto \exists u'\ q'(v,u')$. *Neka su date formule* $p(v,u) \Rightarrow q(v,u) \vee r(v,u)$ *i* $\forall x'\ r(v',x') \wedge p'(v',u') \Rightarrow q'(v',u')$. *Primenom drugog pravila koherentne rezolucije na ove formule se može dobiti rezolventa* $\forall u\ p(v,u) \wedge p'(v,u') \Rightarrow \exists u\ q(v,u) \vee q'(v,u')$. *U ovom slučaju, funkcija upraivanja pri rezolviranju je data sledećim pridruživanjima* $p(v,u) \mapsto \forall u\ p(v,u)$, $p'(v',u') \mapsto p'(v,u')$, $q(v,u) \mapsto \exists u\ q(v,u)$ *i* $q'(v',u') \mapsto \exists u'\ q'(v,u')$.

Sledeća definicija uvodi koherentno faktorisanje. Primer koji ilustruje potrebu za uvođenjem ovog pojma je sledeći. Neka su date formule $s(u) \Rightarrow \exists v \exists w\ p(v,w)$ i $p(x,y) \wedge p(x,z) \Rightarrow \bot$. Očigledno je da se na osnovu datih formula, rezonovanjem unapred može dokazati formula $s(u) \Rightarrow \bot$. Rezolviranjem po drugom atomu prve formule i prvom atomu druge, dobija se $s(u) \wedge \forall x \forall z\ p(x,z) \Rightarrow \bot$, što je formula koja se ne može ponovo rezolvirati sa prvom formulom. S druge strane, ako se iz polazne formule prvo izvede formula $p(x,y) \Rightarrow \bot$, iz nje se rezolviranjem dobija formula $s(u) \Rightarrow \bot$. Pošto se pravilo rezolucije primenjuje samo na po jedan pk-literal iz obe

formule, potrebno je uvesti relaciju faktorisanja. Prvo se uvodi relacija faktorisanja za samo jednu grupu pk-literala, a onda se induktivno definiše relacija potpunog faktorisanja formule koje je korisna u slučaju da formula sadrži više različitih grupa pk-literala po kojima je moguće uraditi faktorisanje.

Definicija 26 (Relacija koraka koherentnog faktorisanja: *FactStep*) *Neka su* $\mathcal{P} \Rightarrow \mathcal{Q}$ *i* $\mathcal{P}' \Rightarrow \mathcal{Q}'$ *koherentne formule i* $m : (\mathcal{P} \Rightarrow \mathcal{Q}) \to \mathcal{QL}(\Sigma)$ *preslikavanje. Neka za neki k-literal* l_c *iz* $m(\mathcal{P} \Rightarrow \mathcal{Q})$ *i skup* $\mathcal{S} = m^{-1}(l_c)$ *važi* $|\mathcal{S}| > 1$*. Takođe, neka važi jedan od sledećih uslova:*

- l_c *je pozitivan k-literal, postoji pk-literal* $\forall\vec{x}\ l$ *takav da važi* $ResStep(l_c, \overline{\forall\vec{x}\ l}, \bot)$ *i da važi* $ResStep_\mu(l\sigma, \overline{\mathcal{S}}\lambda, \bot)$ *i važi* $\mathcal{P}' = (\mathcal{P} \setminus \mathcal{S})\lambda \cup \{\forall\vec{x}\ l\}$ *i* $\mathcal{Q}' = \mathcal{Q}\lambda$*;*
- l_c *je negativan k-literal, postoji pk-literal* $\exists\vec{y}\ l$ *takav da važi* $ResStep(l_c, \exists\vec{y}\ l, \bot)$ *i da važi* $ResStep_\mu(\overline{l\sigma}, \mathcal{S}\lambda, \bot)$ *i važi* $\mathcal{P}' = \mathcal{P}\lambda$ *i* $\mathcal{Q}' = (\mathcal{Q} \setminus \mathcal{S})\lambda \cup \{\exists\vec{y}\ l\}$*,*

pri čemu je σ *supstitucija koja slobodnim promenljivim pk-literala* $\forall\vec{x}\ l$ *(u prvom slučaju), odnosno* $\exists\vec{y}\ l$ *(u drugom slučaju), dodeljuje međusobno različite simbole konstanti koji nisu u signaturi,* λ *je supstitucija u kojoj ne učestvuju konstante koje nisu u signaturi, a* μ *je supstitucija koja uključuje samo zamene promenljivih konstantama koje nisu u signaturi. Kaže se da se formula* $\mathcal{P}' \Rightarrow \mathcal{Q}'$ *dobija* faktorisanjem *formule* $\mathcal{P} \Rightarrow \mathcal{Q}$ *po skupu* $\mathcal{S}$ *u odnosu na preslikavanje* m*, što se zapisuje* $FactStep^{\mathcal{S}}_{\lambda}(\mathcal{P} \Rightarrow \mathcal{Q}, m, \mathcal{P}' \Rightarrow \mathcal{Q}')$*.*

Ukoliko u datom kontekstu skup $\mathcal{S}$ *ili supstitucija* λ *nisu bitni, mogu se izostaviti u zapisu.*

Vrednost funkcije uparivanja pri faktorisanju *formule* $\mathcal{P} \Rightarrow \mathcal{Q}$ *po skupu* $\mathcal{S}$ *u odnosu na preslikavanje* m*,* $f : \mathcal{P} \cup \mathcal{Q} \to \mathcal{P}' \cup \mathcal{Q}'$*, za pk-literal* l' *je*

- $\forall\vec{x}\ l$ *ukoliko je* l_c *pozitivan k-literal i važi* $l' \in \mathcal{S}$*,*
- $\exists\vec{y}\ l$ *ukoliko je* l_c *negativan k-literal i važi* $l' \in \mathcal{S}$ *ili*
- $l'\lambda$ *ukoliko važi* $l' \notin \mathcal{S}$*.*

Svrha uslova $ResStep(l_c, \overline{\forall\vec{x}\ l}, \bot)$ je da k-literal l_c koji može da zadovolji pk-literale u levoj strani aksiome, takođe može da zadovolji i pk-literal koji ih zamenjuje u formuli dobijenoj faktorisanjem. Svrha uslova $ResStep_\mu(l\sigma, \overline{\mathcal{S}}\lambda, \bot)$ je da proizvedeni pk-literal zadovoljava sve pk-literale u faktorisanoj formuli koji odgovaraju pk-literalima koje je zamenio. Svrha uslova u drugom slučaju definicije je slična.

Primer 9 *Neka je data formula* $\mathcal{F} = q(x,y) \wedge \forall z\ p(x,y,z) \wedge \forall z\ p(z,z,v) \Rightarrow \bot$ *i k-literali* $q(u,v)$ *i* $p(u,u,w)$*. Neka je* m *preslikavanje* $[q(x,y) \mapsto q(u,v), \forall z\ p(x,y,z) \mapsto p(u,u,w), \forall z\ p(z,z,v) \mapsto p(u,u,w)]$ *i neka je* $\mathcal{F}' = q(u,u) \wedge \forall x' \forall z'\ p(x',x',z') \Rightarrow \bot$*. Neka je* $\mathcal{S} = \{\forall z\ p(x,y,z), \forall z\ p(z,z,v)\}$ *i* $\lambda = [x \mapsto u, y \mapsto u]$*. Tada, na osnovu prvog slučaja definicije relacije* $FactStep$*, važi* $FactStep^{\mathcal{S}}_{\lambda}(\mathcal{F}, m, \mathcal{F}')$*. To pokazujemo detaljnije. Očigledno važi* $ResStep(p(u,u,w), \overline{\forall x' \forall z'\ p(x',x',z')}, \bot)$*. Takođe važi* $ResStep_{[]}(\forall x' \forall z'\ p(x',x',z'), \{\overline{\forall z\ p(u,u,z)}, \overline{\forall z\ p(z,z,v)}\}, \bot)$*, što znači da je prvi uslov definicije ispunjen. U ovom primeru su i* μ *i* σ *prazne supstitucije. Funkcija uparivanja* f *je* $[q(x,y) \mapsto q(u,u), \forall z\ p(x,y,z) \mapsto \forall x' \forall z'\ p(x',x',z'), \forall z\ p(z,z,v) \mapsto \forall x' \forall z'\ p(x',x',z')]$*.*

Treba primetiti da je faktorisanje uvek moguće u slučaju kad važi $|\mathcal{S}| > 1$. Ukoliko je $l_c = p(\vec{x})$, za pk-literal $\forall \vec{x}\ l$ se može uzeti $\forall \vec{x}\ p(\vec{x})$. Ukoliko je $l_c = \exists \vec{y}\ p(\vec{y})$, za pk-literal $\forall \vec{x}\ l$ se može uzeti $p(\vec{y})$ (niz $\vec{x}$ je prazan). Ukoliko je $l_c = \overline{p(\vec{y})}$, za pk-literal $\exists \vec{y}\ l$ se može uzeti $\exists \vec{y}\ p(\vec{y})$. Ukoliko je $l_c = \overline{\forall \vec{x}\ p(\vec{x})}$, za pk-literal $\exists \vec{y}\ l$ se može uzeti $p(\vec{x})$ (niz $\vec{y}$ je prazan).

Definicija 27 (Relacija potpunog koherentnog faktorisanja: $Fact$**)** *Formula* $\mathcal{F}'$ *se dobija* potpunim faktorisanjem *od formule* $\mathcal{F}$ *u odnosu na preslikavanje* $m : \mathcal{F} \to \mathcal{QL}(\Sigma)$*, što se zapisuje* $Fact_{\lambda}(\mathcal{F}, m, \mathcal{F}')$*, ukoliko važi jedan od uslova:*

- $\mathcal{F} = \mathcal{F}'$ *i* $\lambda = []$*;*
- $FactStep^{\mathcal{S}}_{\lambda_1}(\mathcal{F}, m, \mathcal{F}'')$ *i* $Fact_{\lambda_2}(\mathcal{F}'', m \circ f^{-1}, \mathcal{F}')$ *pri čemu je* $\lambda = \lambda_1\lambda_2$*, a* f *funkcija uparivanja pri faktorisanju formule* $\mathcal{F}$ *po skupu* $\mathcal{S}$ *pri preslikavanju* m*.*

Vrednost funkcije uparivanja pri potpunom faktorisanju *formule* $\mathcal{F}$ *u odnosu na preslikavanje* m*,* $f' : \mathcal{F} \to \mathcal{QL}(\Sigma)$ *je u prvom slučaju identička funkcija, a u drugom se definiše kao* $f' = f'' \circ f$ *pri čemu je* f'' *funkcija uparivanja pri potpunom faktorisanju formule* $\mathcal{F}''$ *u odnosu na preslikavanje* $m \circ f^{-1}$*.*

Primer 10 *Neka je data formula* $\mathcal{F} = \forall x\ p(v,x) \wedge \forall x\ p(x,v) \Rightarrow \exists y\ q(v,y) \vee \exists y\ q(y,v)$ *i preslikavanje* $m = [\forall x\ p(v,x) \mapsto p(u',v'), \forall x\ p(x,v) \mapsto p(u',v')]$*,* $\exists y\ q(v,y) \mapsto \overline{q(u',v')}$ *i* $\exists y\ q(y,v) \mapsto \overline{q(u',v')}$*. Tada važi* $Fact(\mathcal{F}, m, \mathcal{F}'')$*, gde je* $\mathcal{F}'' = \forall u' \forall v'\ p(u',v') \Rightarrow \exists u' \exists v'\ q(u',v')$ *jer važi* $FactStep^{\{\forall x\ p(v,x), \forall x\ p(x,v)\}}(\mathcal{F}, m, \mathcal{F}')$*, gde je* $\mathcal{F}' = \forall u' \forall v'\ p(u',v') \Rightarrow \exists y\ q(v,y) \vee \exists y\ q(y,v)$ *i* $Fact(\mathcal{F}', \mathcal{F}'')$*. Poslednja relacija važi zato što važi* $FactStep\{\exists y\ q(v,y), \exists y\ q(y,v)\}(\mathcal{F}', m, \mathcal{F}'')$*.*

Na osnovu pravila rezolviranja i faktorisanja, može se definisati uopšteno pravilo koherentne rezolucije. U njemu formule koje se rezolviraju neće biti tretirane simetrično. Samo će prva formula koja učestvuje u rezoluciji biti faktorisana pre primene

pravila rezolucije. I druga bi mogla biti faktorisana, ali, kao što će se ispostaviti, to nije potrebno u predloženom sistemu.

Definicija 28 (Relacija uopštenog koherentnog rezolviranja: *ResGen*) *Formula $\mathcal{R}$ se dobija od formula $\mathcal{F}_1$ i $\mathcal{F}_2$ uopštenom koherentnom rezolucijom u odnosu na preslikavanja $m_1 : \mathcal{F}_1 \to \mathcal{QL}(\Sigma)$ i $m_2 : \mathcal{F}_2 \to \mathcal{QL}(\Sigma)$, što se zapisuje $ResGen_{\lambda}^{\mathcal{S}_1 \mapsto l_2}(\mathcal{F}_1, \mathcal{F}_2, m_1, m_2, \mathcal{R})$ ukoliko su ispunjeni uslovi $Fact_{\lambda_1}(\mathcal{F}_1, m_1, \mathcal{F}_1')$, $ResStep_{\lambda_{r'}}^{l_1 \mapsto l_2}(\mathcal{F}_1', \mathcal{F}_2, \mathcal{R}')$ i $Fact_{\lambda_r}(\mathcal{R}', m_r, \mathcal{R})$, pri čemu važi*

- *Formule $\mathcal{F}_1$ i $\mathcal{F}_2$ nemaju zajedničkih promenljivih;*
- *$\mathcal{S}_1 = f_1^{-1}(l_1)$ pri čemu je f_1 funkcija uparivanja pri potpunom faktorisanju formule $\mathcal{F}_1$;*
- *$\lambda = \lambda_1 \lambda_{r'} \lambda_r$*
- *preslikavanje $m_r : \mathcal{R}' \to \mathcal{QL}(\Sigma)$ se definiše kao $m_r(l) = m_1(f_1^{-1}(r^{-1}(l)))$ ukoliko važi $r^{-1}(l) \subseteq \mathcal{F}_1$, a $m_r(l) = m_2(r^{-1}(l))$ ukoliko važi $r^{-1}(l) \in \mathcal{F}_2$.*

Vrednost funkcije uparivanja pri uopštenom rezolviranju *formula $\mathcal{F}_1$ i $\mathcal{F}_2$ u odnosu na preslikavnja m_1 i m_2, $u : \mathcal{F}_1 \cup \mathcal{F}_2 \to \mathcal{R}$, za pk-literal l se definiše kao $u(l) = f(r(f_1(l)))$ ukoliko važi $l \in \mathcal{F}_1$ i $f_1(l) \neq l_1$, a kao $u(l) = f(r(l))$ ukoliko važi $l \in \mathcal{F}_2$ i $l \neq l_2$ pri čemu je r funkcija uparivanja pri rezolviranju formula $\mathcal{F}_1'$ i $\mathcal{F}_2'$, a f funkcija uparivanja pri potpunom faktorisanju formule $\mathcal{R}'$.*

Funkcija uparivanja je dobro definisana zahvaljujući pretpostavci da nijedna koherentna formula nema isti pk-literal i u levoj i u desnoj strani i pretpostavci da su skupovi promenljivih formula koje se rezolviraju disjunktni.

Sledeća definicija uvodi relaciju rezolviranja koja važi ukoliko se formula $\mathcal{R}$ može dobiti od formule $\mathcal{F}$ tako što se jedan po jedan k-literal iz liste L rezolvira sa formulom $\mathcal{F}$, odnosno njenim rezolventama iz prethodnih rezolviranja. Ovakva relacija je korisna za izražavanje konflikta formule sa skupom k-literala ili mogućnosti da se na osnovu neke formule i skupa k-literala propagira pk-literal. Funkcija m koja se pojavljuje u definiciji relacije rezolviranja povezuje pk-literale iz formule $\mathcal{F}$ sa k-literalima iz liste L po kojima je vršeno rezolviranje.

Definicija 29 (Relacija izvođenja rezolucijom: *Res*) *Formula $\mathcal{R}$ se* izvodi rezolucijom *iz formule $\mathcal{F}$ i k-literala iz liste L, što se označava $Res_{\lambda}^{m}(\mathcal{F}, L, \mathcal{R})$, ukoliko važi*

- *$\mathcal{F} = \mathcal{R}$, pri čemu je m prazno preslikavanje, a λ prazna supstitucija ili*

- *postoje preslikavanja* $m_F : \mathcal{F} \to \mathcal{QL}(\Sigma)$ *i* $m_L : \{l\} \to \mathcal{QL}(\Sigma)$*, gde je* l *k-literal iz* L*, takva da važi* $ResGen^{\mathcal{S}\mapsto l}_{\lambda_1}(\mathcal{F}, l, m_F, m_L, \mathcal{R}')$ *za neku formulu* $\mathcal{R}'$ *i* $Res^{m_2}_{\lambda_2}(\mathcal{R}', L, \mathcal{R})$*, pri čemu je* m *definisano kao* $m(l') = l$ *ukoliko važi* $l' \in \mathcal{S}$*,* a $m(l') = m_2(u(l'))$ *ukoliko važi* $l' \notin \mathcal{S}$ *gde je* u *funkcija uparivanja pri uopštenom rezolviranju formula* $\mathcal{F}$ *i* l *u odnosu na preslikavanja* m_F *i* m_L *i pri čemu je* $\lambda = \lambda_1\lambda_2$*.*

Ukoliko važi $Res^m_\lambda(\mathcal{F}, L, \mathcal{R})$ *za neko* m*, piše se i* $Res_\lambda(\mathcal{F}, L, \mathcal{R})$*. Ukoliko važi* $Res^m_\lambda(\mathcal{F}, L, \mathcal{R})$ *za neko* λ*, piše se i* $Res^m(\mathcal{F}, L, \mathcal{R})$*. Ukoliko važi* $Res^m_\lambda(\mathcal{F}, L, \mathcal{R})$ *za neko* m *i neko* λ*, piše se i* $Res(\mathcal{F}, L, \mathcal{R})$*. Skup* $m(\mathcal{F})$ *se onda naziva* konfliktnim skupom*, k-literali u njemu* konfliktnim literalima*, a* m konfliktnim preslikvanjem *za formulu* $\mathcal{F}$*.*

Treba primetiti da se u definiciji konfliktnih literala insistira na tome da su oni k-literali, nasuprot pk-literalima. Razlog za ovo potiče od izbora fragmenta logike u kojem očekujemo da budu naučene leme.

Primer 11 *Važi* $Res^m(s(x) \Rightarrow \exists y\ p(x,y), [\exists x\ s(x)], \exists x \exists y\ p(x,y))$ *pri čemu je* $m = [s(x) \mapsto \exists x\ s(x)]$*, a* $r = [\exists y\ p(x,y) \mapsto \exists x \exists y\ p(x,y)]$*.*

Važi $Res^{m'}(\exists x \exists y\ p(x,y), [\overline{p(x,y)}], \bot)$*, pri čemu je* m' *pridruživanje* $\exists x \exists y\ p(x,y) \mapsto \overline{p(x,y)}$*. Dodatno, važi* $Res^m(s(x) \Rightarrow \exists y\ p(x,y), [\overline{p(x,y)}, \exists x\ s(x)], \bot)$ *za* $m(s(x)) = \exists x\ s(x)$ *i* $m(\exists y\ p(x,y)) = m'(r(\exists y\ p(x,y))) = m'(\exists x \exists y\ p(x,y)) = \overline{p(x,y)}$*.*

4.4.3 Apstraktni sistem pravila za koherentnu logiku

U nastavku se definišu osnovni elementi apstraktnog sistema pravila za koherentnu logiku, kao i sam sistem.

Definicija 30 (Signatura i koherentno tvrđenje) *Neka je* $\mathcal{L} = (\Sigma^\infty, \Pi, ar)$ *signatura takva da je* $\Sigma^\infty = \{c^i \mid i \in \mathbf{N} \setminus \{0\}\}$*, pri čemu za svako* $i = 1, \ldots$ *važi* $ar(c^i) = 0$ *i neka je* Π *konačan skup predikatskih simbola. Neka je* no_cflct *specijalni simbol koji se ne pojavljuje u signaturi.*

Neka je data koherentna teorija $\mathcal{T}$*, to jest konačan skup koherentnih aksioma* $\mathcal{AX}$*, nad prebrojivim skupom promenljivih* V *i signaturom* $(\Sigma_\mathcal{T}, \Pi, ar)$ *gde je* $\Sigma_\mathcal{T} = \{c^1, \ldots, c^k\} \subseteq \Sigma^\infty$ *i* $k \geq 0$*, i* koherentno tvrđenje $\Phi = \forall \vec{x}\ \mathcal{H}(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v}, \vec{y})$ *nad istom signaturom, pri čemu je* $\mathcal{H}(\vec{v}, \vec{x})$ *jednako* $h_1(\vec{v}, \vec{x}) \wedge \ldots \wedge h_m(\vec{v}, \vec{x})$ *i* $\mathcal{G}(\vec{v}, \vec{y})$ *jednako* $g_1(\vec{v}, \vec{y}) \vee \ldots \vee g_n(\vec{v}, \vec{y})$*, a* h_i *i* g_i *su predikatski simboli iz* Π*. Sa* $\mathcal{PR}(\Phi)$ *označava se skup* $\{h_1(\vec{v}, \vec{x}), \ldots, h_m(\vec{v}, \vec{x}), \overline{g_1(\vec{v}, \vec{y})}, \ldots, \overline{g_n(\vec{v}, \vec{y})}\}\lambda$*, pri čemu je* λ *bazna supstitucija nad* $\vec{v}$ *gde sve konstante koje se javljaju u supstituciji* λ *pripadaju skupu* $\Sigma^\infty \setminus \Sigma_\mathcal{T}$ *i međusobno su različite.*

U daljem tekstu pod izrazom tvrđenje će se podrazumevati koherentno tvrđenje. Tvrđenje će u predloženom sistemu biti dokazivano pobijanjem, a prilikom primene sistema pravila, skup $\mathcal{PR}(\Phi)$ će služiti kao skup polaznih pretpostavki.

Definicija 31 (Stanje) Stanje *je uređena petorka* $(\Sigma, \Gamma, M, \mathcal{C}, \ell)$*, gde je* Σ *konačna lista elemenata iz* Σ^∞*,* Γ *je konačna lista koherentnih formula nad prebrojivim skupom promenljivih* V *i signaturom* (Σ_T, Π, ar)*,* M *je lista različitih prošireno kvantifikovanih literala nad skupom promenljivih* V *i signaturom* (Σ, Π, ar)*,* $\mathcal{C}$ *je formula koja se naziva* konfliktnom implikacijom *ili simbol* no_cflct*, a* ℓ *je indeks poslednje uvedene konstante.* Polazno stanje *za tvrđenje* Φ *je stanje* $S_0 = (\Sigma_0, \mathcal{AX}, \mathcal{PR}(\Phi), no\_cflct, n)$*, gde je* Σ_0 *skup konstanti iz* $\mathcal{PR}(\Phi)$ *i* $\Sigma_\mathcal{T}$ *(ako je* Σ_0 *prazna lista, u nju se dodaje prva konstanta iz* Σ^∞*), a* n *je maksimum indeksa konstanti u listi* Σ_0*.*

Intuitivno, uloga komponenti stanja je sledeća: Σ sadrži konstante iz polazne teorije i uvedene svedoke, Γ sadrži aksiome i naučene leme, M sadrži pretpostavljene i izvedene pk-literale. Izvedeni pk-literali predstavljaju logičke posledice aksioma i pretpostavljenih literala. Formula $\mathcal{C}$ se koristi u procesu *analize konflikta* čiji je cilj pronalaženje opštijeg razloga zbog kojih je došlo do konflikta u cilju učenja leme.

Definicija 32 (Nivoi odlučivanja) *Elementi listi* M *i* Σ *se mogu podeliti na* nivoe odlučivanja*. Elementi različitih nivoa odlučivanja se razdvajaju simbolom* $|$*. Redni broj (počevši od 0) nivoa u listi* M *na kojem se nalazi pk-literal* l *se označava* $level(l)$*.*

Prefiks liste L *koji uključuje tačno elemente prvih* m *nivoa odlučivanja se obeležava sa* L^m*.*

Za listu L *i element* e*, zapis* $L\ e$ *označava listu dobijenu od* L *dodavanjem elementa* e *na kraj liste* L *ukoliko važi* $e \notin L$*, a listu* L *ukoliko važi* $e \in L$*.*

Definicija 33 (Relacija prethođenja: $\prec$**)** *Ako u nekom stanju za listu* M *važi* $M = M_1\ l\ M_2\ l'\ M_3$*, pri čemu bilo koja od listi* M_i *(*$i = 1, 2, 3$*) može biti prazna, to se zapisuje* $l \prec l'$*. Ako za skup pk-literala* S *i pk-literal* l' *važi* $l \prec l'$ *za svaki literal* $l \in S$*, to se zapisuje* $S \prec l'$*.*

Primer 12 *Neka je* $M = [p(a, b), q(x, y), r(x, y)]$*. Tada važi* $p(a, b) \prec q(x, y)$*. Takođe važi* $\{p(a, b), q(x, y)\} \prec r(x, y)$*.*

Sledeća definicija uvodi relaciju tačnosti pk-literala u odnosu na listu M. Intuitivno, relacija tačnosti se definiše na osnovu relacije rezolviranja po principu da jedan pk-literal povlači drugi ako je u kontradikciji sa njegovom negacijom. U definiciji se pribegava korišćenju skolemovih konstanti za eliminaciju egzistencijalnih kvantifikatora nastalih negiranjem implicinih univerzalnih kvantifikatora.

Definicija 34 (Relacija tačnosti: $\uparrow$) *K-literal l je* tačan *u odnosu na listu M, što se zapisuje $l \uparrow$ ukoliko postoji k-literal l' u listi M, takav da važi jedan od sledećih uslova*

- $l = p(\vec{v})$ *i* $ResStep(l', \overline{p(\vec{v})\sigma}, \bot)$
- $l = \exists \vec{y}\ p(\vec{y})$ *i* $ResStep(l', \overline{p(\vec{y})}, \bot)$
- $l = \overline{p(\vec{v})}$ *i* $ResStep(l', p(\vec{v})\sigma, \bot)$
- $l = \forall \vec{x}\ \overline{p(\vec{x})}$ *i* $ResStep(l', p(\vec{x}), \bot)$

pri čemu je u oba slučaja σ supstitucija koja promenljivim $\vec{v}$ dodeljuje međusobno različite simbole konstanti koji nisu u signaturi.

Pk-literal l koji nije k-literal je tačan u odnosu na listu M ukoliko postoji bilo kakav pk-literal l' u listi M koji zadovoljava neki od prethodnih uslova.

Razlika u tretmanu k-literala i pk-literala dovodi do toga da k-literal može biti ubačen u listu M čak i ako u njoj postoji pk-literal čija je dati k-literal posledica. Naime, kako izbor fragmenta logike u kojem se radi nameće ograničenje na vrstu elementarnih formula koje će biti korišćene u rezonovanju, upotrebna vrednost pk-literala i k-literala nije ista. Na primer, ukoliko važi $p(x, y), \overline{\forall x\ q(x, y)} \in M$, može se zaključiti da je formula $p(x, y) \Rightarrow q(x, y)$ u kontradikciji sa listom M. Međutim, u opštem slučaju, upotreba pk-literala u cilju detekcije kontradikcije ili jedinične propagacije može dovesti do toga da se lema koju bi trebalo naučiti ne nalazi u korišćenom fragmentu. S druge strane, zaključak o kontradiktornosti bi mogao biti izveden i u slučaju da je u listi prisutan k-literal $\overline{\forall x\ q(x, a)}$, a korišćenje k-literala u pomenute svrhe ne može dovesti do navedenog problema. Stoga se dozvoljava dodavanje k-literala u listu M, čak i ako je u njoj već prisutan pk-literal čija je on posledica.

Primer 13 *Ukoliko važi $p(u, v) \in M$, onda važi $p(x, x) \uparrow$, zahvaljujući tome sto važi $ResStep(p(u, v), \overline{p(x, x)}[x \mapsto c], \bot)$, pri čemu je c simbol konstante koji nije u signaturi. Smisao definicije postaje jasniji eksplicitnim navođenjem univerzalne kvantifikacije slobodnih promenljivih i korišćenjem negacije. Pk-literal $p(x, x)$ predstavlja formulu $\forall x\ p(x, x)$. Njena negacija je $\neg\forall x\ p(x, x) = \exists x\ \neg p(x, x)$. Skolemizacijom, izostavljanjem univerzalnog kvantifikatora i zapisa negacije u implikacijskoj formi se dobija $\overline{p(x, x)}[x \mapsto c]$. Kontradiktornost (ustanovljena rezolucijom) formula $p(u, v)$ i $\overline{p(c, c)}$ svedoči da je pk-literal $p(x, x)$ logička posledica pk-literala $p(u, v)$, odnosno liste M u kojoj se poslednji pk-literal nalazi.*

Sledeća definicija uvodi relaciju konflikta literala sa listom M.

Definicija 35 (Relacija konflikta: $\downarrow$) *Pk-literal l je u* konfliktu *sa listom M, što se zapisuje $l\downarrow$, ukoliko postoji pk-literal l u listi M, takav da važi $ResStep(l, l', \bot)$.*

Definicija 36 (Relacija neodređenosti: ?) *Pk-literal je* neodređen *u odnosu na listu M, što se zapisuje $l?$, ukoliko ne važi ni $l\uparrow$, ni $l\downarrow$.*

Definicija 37 (Apstraktni sistem pravila za koherentnu logiku) *Za datu fiksiranu signaturu $\mathcal{L} = (\Sigma^\infty, \Pi, ar)$,* apstraktni sistem pravila za koherentnu logiku *je sistem pravila*[2] *prikazan na slici 4.1. Svako pravilo, kada je primenljivo, preslikava jedno stanje u drugo. Pre primene pravila, vrši se preimenovanje promenljivih u formulama na koje se referiše u preduslovima pravila, tako da formule nemaju zajedničkih promenljivih.*

Decide:
$$\frac{l \in \mathcal{QL}(\Sigma) \qquad l?}{M := M|l \qquad \Sigma := \Sigma|}$$

Intro:
$$\frac{\exists\vec{y}\; l \in M \qquad \exists\vec{y}\; l\lambda \in \mathcal{QL}(\Sigma) \qquad l\lambda\lambda'? \text{ za svako } \lambda'}{M := M\; l\lambda[y_1 \mapsto c^{\ell+1}, \ldots, y_k \mapsto c^{\ell+k}] \qquad \Sigma := \Sigma\; c^{\ell+1}, \ldots, c^{\ell+k} \qquad \ell := \ell + k}$$

Unit propagation:
$$\frac{\mathcal{F} \in \Gamma \qquad Res(\mathcal{F}, M, l) \qquad l?}{M := M\; l}$$

Conflict:
$$\frac{\mathcal{C} = no\_cflct \qquad \mathcal{F} \in \Gamma \qquad Res(\mathcal{F}, M, \bot)}{\mathcal{C} := \mathcal{F}}$$

Explain:
$$\frac{\begin{array}{ccc}\mathcal{C} \neq no\_cflct & Res^m(\mathcal{C}, M, \bot) & l \in m(\mathcal{C}) \qquad \mathcal{F} \in \Gamma \\ Res^{m'}(\mathcal{F}, M, l) & m'(\mathcal{F}) \prec l & ResGen(\mathcal{C}, \mathcal{F}, m, m', \mathcal{R})\end{array}}{\mathcal{C} := \mathcal{R}}$$

Learn:
$$\frac{\mathcal{C} \neq no\_cflct \qquad \mathcal{C} \notin \Gamma}{\Gamma := \Gamma\; \mathcal{C}}$$

Backjump:
$$\frac{\mathcal{C} \in \Gamma \qquad Res(\mathcal{C}, M, \bot) \qquad Res(\mathcal{C}, M^n, l)}{M := M^n\; l \qquad \Sigma := \Sigma^n \qquad \mathcal{C} := no\_cflct}$$

Slika 4.1: Apstraktni sistem pravila za koherentnu logiku. Iz konteksta je jasno koja je vrsta objekata koji figurišu u sistemu.

[2]Objašnjenja pravila, koja je preporučljivo čitati paralelno sa njihovim definicijama, kao i primer izvršavanja pravila sistema, dati su u nastavku.

Decide: Ovim pravilom se uvodi pretpostavka. Pretpostavka može biti k-literal koji je neodređen u odnosu na listu M (na primer, ako je kvantifikovani atom $s(y)$ u listi, tada se pravilo ne može primeniti za kvantifikovane atome $s(c)$ i $\exists x\ s(x)$). U ovom koraku se u listama M i Σ označava da počinje novi nivo (koji zavisi od napravljene pretpostavke).

Intro: Ovim pravilom se eliminiše egzistencijalni kvantifikator i uvodi se novi simbol konstante kao svedok. Na primer, ukoliko je $M = [\exists y\ q(a, y)]$ i $\Sigma = [a]$, ovo pravilo može da doda novi simbol konstante b u listu Σ i kvantifikovani atom $q(a, b)$ u listu M.

Unit propagation: Ukoliko se rezolviranjem formule sa kvantifikovanim literalima iz liste M može dobiti pk-literal l koji je neodređen u odnosu na listu M, onda se on može dodati u listu M. Ako je $\overline{p(x, y)} \in M$, na osnovu formule $s(x) \Rightarrow \exists y\ p(x, y)$ je moguće propagirati pk-literal $\overline{s(x)}$. Ukoliko je $\overline{p(a, y)} \in M$, može se propagirati $\overline{s(a)}$. Ukoliko je $\exists z\ s(z) \in M$, propagira se pk-literal $\exists z \exists y\ p(z, y)$.

Conflict: Ukoliko se rezolviranjem formule sa k-literalima iz liste M može izvesti prazna klauza, onda je skup trenutnih pretpostavki iz liste M i aksioma kontradiktoran i potreban je povratak u pretrazi. Tada započinje proces *analize kofnlikta*, čiji je cilj pronalaženje opštijeg razloga konflikta na osnovu čega se može naučiti lema koja se može koristiti za zatvaranje kasnijih grana pretrage koje se mogu zatvoriti izvođenjem analognim onome koje je dovelo do tekućeg konflikta.

Explain: Ovo pravilo vrši analizu konflikta rezolviranjem konfliktne implikacije i formule iz Γ koja je mogla propagirati k-literal u konfliktnom skupu. Neka je konfliktna implikacija $p(u, v) \wedge q(u, v) \Rightarrow r(u, v)$ i neka postoji aksioma $s(x) \Rightarrow \exists y\ p(x, y)$ koja je propagirala k-literal koji je u konfliktnom skupu konfliktne implikacije. Rezolviranjem ovih formula dobija se nova konfliktna implikacija $s(x) \wedge \forall y\ q(x, y) \Rightarrow \exists y\ r(x, y)$ koja zamenjuje staru.

Učenje: U procesu analize konflikta, izvodi se formula $\mathcal{C}$. Pošto je ova formula posledica aksioma može se dodati u skup Γ kao naučena lema.

Povratni skok: Pošto je konfliktna implikacija $\mathcal{C}$ u konfliktu sa listom M, neki od k-literala iz liste M moraju biti uklonjeni, pa se stoga primenjuje povratni skok. Nivo na koji se povratni skok vrši je bilo koji takav da se ukloni samo k-literal iz konfliktnog skupa sa najvišim nivoom u listi M. Kako su svi ostali k-literal iz konfliktnog skupa i nakon povratnog skoka prisutni u listi M, iz konfliktne implikacije se može izvesti novi pk-literal koji sprečava da se uklonjeni k-literal iz konfliktnog skupa ponovo pojavi u listi M.

Definicija 38 (Relacija prelaska: $\xrightarrow{r}$) *$S \xrightarrow{r} S'$ označava da stanje S' može biti dobijeno iz stanja S primenom pravila r. $S \to S'$ označava da važi $S \xrightarrow{r} S'$ za neko*

pravilo r. Niz stanja S_i, takvih da važi $S_i \to S_{i+1}$ za svako i, se naziva lanac.

Naredna definicija uvodi pojam završnih stanja. S obzirom na to da je namena sistema dokazivanje teorema, a ne pronalaženje modela, stanje se smatra prihvatajućim ukoliko je pobijanje uspešno izvršeno.

Definicija 39 (Završna stanja) Prihvatajuće stanje *je stanje u kojem važi $\mathcal{C} \neq no\_cflct$, $Res^m(\mathcal{C}, M, \bot)$ i $m(\mathcal{C}) \subseteq \mathcal{PR}(\Phi)$, pri čemu je Φ tvrđenje ili stanje S_0 ako u njemu važi $Res(l, M, \bot)$ za neki k-literal $l \in M$. To se zapisuje $\mathcal{AX} \vdash_{CL} \forall \vec{x}\ \mathcal{H}(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v}, \vec{y})$.* Odbacujuće stanje *je stanje S koje nije prihvatajuće i za koje nema stanja S' takvog da važi $S \to S'$.* Završno stanje *je prihvatajuće stanje ili odbacujuće stanje.*

Sledeći primer ilustruje primenu pravila predloženog sistema.

Primer 14 *Neka su aksiome $\mathcal{AX}$ teorije $\mathcal{T}$ implicitno univerzalno zatvorene formule:*
(Ax1) $p(x,y) \wedge q(x,y) \wedge r(x,y) \Rightarrow \bot$,
(Ax2) $s(x) \Rightarrow \exists y\ q(x,y)$,
(Ax3) $q(x,y) \Rightarrow r(x,y)$ *i*
(Ax4) $s(x) \vee q(y,y)$
i neka je tvrđenje $\Phi = \forall z\ p(x,z) \Rightarrow \bot$.

Slobodna promenljiva x u tvrđenju se instancira novom konstantom a i važi $\mathcal{PR}(\Phi) = \{p(a,z)\}$. Polazno stanje je $S_0 = (\{a\}, \mathcal{AX}, p(a,z), \emptyset, \emptyset, 1)$. Detalji jednog mogućeg izvršavanja sistema su dati u tabeli. Pošto redosled primene pravila nije fiksiran, moguća su i drugačija izvršavanja. Kako je poslednje stanje u tabeli prihvatajuće stanje, tvrđenje je dokazano.

Pravilo	Σ	$\Gamma \setminus \mathcal{AX}$ *(leme)*	M	$\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
	a	$\emptyset$	$p(a,z)$	$\emptyset \Rightarrow \{no\_cflct\}$
Decide	$a\|$	$\emptyset$	$p(a,z)\|s(a)$	$\emptyset \Rightarrow \{no\_cflct\}$
UP (Ax2)	$a\|$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y)$	$\emptyset \Rightarrow \{no\_cflct\}$
Intro	$a\|b$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b)$	$\emptyset \Rightarrow \{no\_cflct\}$
UP (Ax1)	$a\|b$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\emptyset \Rightarrow \{no\_cflct\}$
Conflict (Ax3)	$a\|b$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$q(x,y) \Rightarrow r(x,y)$
Explain (Ax1)	$a\|b$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$p(x,y) \wedge q(x,y) \Rightarrow \bot$
Explain (Ax2)	$a\|b$	$\emptyset$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$
Learn	$a\|b$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z)\|s(a), \exists y\ q(a,y), q(a,b), \overline{r(a,b)}$	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$
Backjump	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}$	$\emptyset \Rightarrow \{no\_cflct\}$
UP (Ax4)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y)$	$\emptyset \Rightarrow \{no\_cflct\}$
UP (Ax1)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$\emptyset \Rightarrow \{no\_cflct\}$
Conflict (Ax3)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$q(x,y) \Rightarrow r(x,y)$
Explain (Ax1)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,y) \wedge q(x,y) \Rightarrow \bot$
Explain (Ax4)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,x) \Rightarrow s(z)$
Explain (lema)	a	$\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot$	$p(a,z), \overline{s(a)}, q(y,y), \overline{r(a,a)}$	$p(x,x) \wedge \forall u\ p(z,u) \Rightarrow \bot$

U automatskom dokazivanju teorema, zarad obezbeđivanja potpunosti sistema ili procedura za dokazivanje, često se koristi sistematična pretraga. Na primer, u slučaju rezolucije, formulišu se strategije izbora klauza za rezolviranje koje garantuju da će prazna klauza biti izvedena ukoliko je tvrđenje koje se dokazuje zaista teorema date teorije. U slučaju računa evolucije modela, ograničava se dubina termova koji mogu figurisati u klauzama koje se smeju koristiti i ta se dubina postepeno povećava. Sličnu strategiju je moguće formulisati i za predloženi sistem, što će biti rešeno uvođenjem dodatnog pravila i malim modifikovanjem postojećih. Stanje se proširuje sa dva elementa — listom brojeva Δ i brojem δ. Stoga, stanje je sedmorka $(\Sigma, \Gamma, M, \mathcal{C}, \ell, \Delta, \delta)$. U polaznom stanju S_0 važi $\delta_0 = \ell_0$ i $\Delta_0 = \delta_0$. Dodaje se pravilo *Limiter*:

Limiter:

$$\frac{\text{Druga pravila nisu primenjiva} \qquad \delta < max\{i \mid c^i \in \Sigma\}}{\delta := \delta + 1}$$

Pravilo Intro se menja tako da je primenljivo samo ukoliko za svaku konstantu c^i iz pk-literala $(\exists \vec{y}\, l)\lambda$ važi $i \leq \delta$. Akcije pravila Decide se proširuju dodelom $\Delta := \Delta|\ \delta$, a pravila Backjump dodelama $\Delta := \Delta^m$ i $\delta := \delta'$ gde je δ' poslednji element u Δ^m.

4.4.4 Odnos CDCL sistema za koherentnu logiku sa CDCL sistemom za SAT

Predloženi CDCL sistem za koherentnu logiku je uopštenje CDCL sistema za SAT. Jedina neophodna izmena koju je potrebno učiniti kako bi se dobio sistem ekvivalentan CDCL sistemu za SAT je zamena skupa kvantifikovanih literala $\mathcal{QL}(\Sigma)$ skupom iskaznih literala L. Uklanjanjem elementa stanja Σ i referisanja na njega i uklanjanjem pravila Intro se dobija jednostavnija i preglednija reprezentacija sistema, ali to nije neophodno. Jedna od važnih razlika u prezentaciji dva sistema je korišćenje konfliktne implikacije, a ne konfliktnog skupa, kao osnovnog elementa za analizu konflikta. Razlog za takvu odluku je to što korespodencija između konfliktne implikacije i konfliktnog skupa u koherentnoj logici nije trivijalna kao korespodencija između konfliktne klauze i konfliktnog skupa u iskaznoj logici. Prednost se daje konfliktnoj implikaciji zato što je nju potrebno rezolvirati u toku procesa analize konflikta i zato što se na osnovu nje može naći i konfliktni skup, dok obratno ne važi.

4.4.5 Procedura `Calypso`

Apstraktni sistem pravila prikazan u prethodnom potpoglavlju ne definiše redosled primene pravila. Redosled pravila nije strogo impliciran ni potrebom za saglasnošću i potpunošću postupka dokazivanja koji bi bio zasnovan na ovim pravilima, ali primena pravila ne može biti ni sasvim proizvoljna. Sledeći primer, koji je već razmatran u drugom kontekstu, to potvrđuje.

Primer 15 *Neka su date aksiome* $\exists y\ p(y)$, $p(x) \Rightarrow q(x)$ *i* $p(x) \wedge q(x) \Rightarrow \bot$. *Tada se može doći do konfliktnog stanja primenom pravila Intro na prvu aksiomu što daje atom* $p(a)$ *za novu konstantu* a, *a onda se može propagirati* $q(a)$, *što dovodi do mogućnosti primene pravila konflikt, pri kojoj aksioma* $p(x) \wedge q(x) \Rightarrow \bot$ *postaje konfliktna implikacija. Ukoliko se pravilo Explain primeni na konfliktnu implikaciju i prvu aksiomu, dobija se nova konfliktna implikacija* $\forall \vec{x}\ q(x) \Rightarrow \bot$. *Rezolviranjem sa drugom aksiomom, dobija se* $\forall \vec{x}\ p(x) \Rightarrow \bot$, *što se više ne može rezolvirati i proces analize konflikta ostaje nedovršen. U slučaju da se pravilo Explain primeni prvo na konfliktnu implikaciju i drugu aksiomu, dobija se* $p(x) \wedge p(x) \Rightarrow \bot$. *Rezolviranjem sa prvom aksiomom, dobija se* $\bot$, *što znači da je polazni skup aksioma kontradiktoran.*

Dati primer demonstrira potrebu za uređivanjem redosleda primene pravila. Na slici 4.2 prikazana je procedura `Calypso` — jedna procedura poluodlučivanja za problem $\mathcal{AX} \vdash_{CL} \mathcal{F}$ (gde je $\mathcal{AX}$ skup koherentnih aksioma, a $\mathcal{F}$ koherentno tvrđenje) zasnovana na predloženom sistemu.

U datoj proceduri se pretpostavlja da su svi elementi stanja globalne promenljive. Funkcije čija imena počinju sa `applicable` proveravaju preduslove odgovarajućih pravila i eventualno još neke uslove ukoliko je to posebno naglašeno. Funkcije čije ime počinje sa `apply` izvršavaju akcije predviđene odgovarajućim pravilom i eventualno još neke komande ukoliko je to posebno naglašeno.

Važna struktura podataka koja se koristi u implementacijama CDCL sistema za SAT je asocijativni niz razloga literala u kojem se za svaki literal pamti na osnovu koje klauze je propagiran. Ova struktura se koristi za usmeravanje pravila Explain. Slična struktura će biti korišćena i u proceduri `Calypso`. Pretpostavlja se da postoji niz k-literala `conflictSet` koji igra ulogu konfliktnog skupa $m(\mathcal{C})$ i asocijativni nizovi `reasonFormula` i `reasonLiterals` koji za svaki pk-literal u listi M ili signaliziraju da on nije propagiran ili čuvaju formulu i skup k-literala na osnovu kojih je pk-literal propagiran u listu M.

- funkcija `applyDecide` za k-literal l koji je odlučen postavlja odgovarajuće vrednosti nizova komandama

```
function calypso(AX : clFormula[], F : clFormula) : bool
begin
    setInitialState(AX,F)
    repeat begin
        while applicableUnitPropagate() do
            applyUnitPropagate()
        if applicableConflict() then
        begin
            applyConflict()
            while not suitableToLearn(C) do
            begin
                if applicableExplain() then
                    applyExplain()
                if isAcceptingState() then
                    return true
            end
            applyLearn()
            applyBackjump()
        end
        else
        begin
            if applicableIntro() then
                applyIntro()
            else if applicableDecide() then
                applyDecide()
            else if applicableLimiter() then
                applyLimiter()
            else
                return false
        end
    end
end
```

Slika 4.2: Procedura `Calypso` koja proverava da li je data koherentna formula teorema skupa koherentnih aksioma.

```
reasonFormula[l] := 0
reasonLiterals[l] := 0
```

što označava da k-literal l nije propagiran.

- funkcija `applyUnitPropagate`, za pk-literal l koji propagira, postavlja odgovarajuće vrednosti nizova komandama

```
reasonFormula[l] := F
reasonLiterals[l] := getConflictSet(F')
```

gde je $\mathcal{F}$ formula iz koje je pk-literal l propagiran, a $\mathcal{F}'$ njen deo bez pk-literala koji odgovara propagiranom pk-literalu l i koji je u konfliktu sa listom M,

- funkcija `applyIntro` za k-literal l' dobijen eliminacijom egzistencijalnog kvantifikatora postavlja odgovarajuće vrednosti nizova komandama

  ```
  reasonFormula[l'] := reasonFormula[l]
  reasonLiterals[l'] := reasonLiterals[l]
  ```

  gde je l pk-literal iz kojeg je literal l' dobijen,

- funkcija `applyConflict` postavlja niz `conflictSet` komandom

  ```
  conflictSet := getConflictSet(C)
  ```

- funkcija `applicableExplain` vraća `false` ukoliko važi `reason[l] = 0` gde je l k-literal iz konfliktnog skupa takav da važi $l' \prec l$ za svaki k-literal l' iz konfliktnog skupa i

- funkcija `applyExplain` je definisana na sledeći način:

  ```
  function applyExplain() : void
  begin
      select l such that l ∈ conflictSet and
                          if l' ∈ conflictSet then l' ≺ l
      F := reasonFormula[l]
      resolve(C, F, l)
      conflictSet := remove(conflictSet, l)
      conflictSet := join(conflictSet, reasonLiterals)
  end
  ```

Funkcija `suitableToLearn` određuje koje će rezolvente biti smatrane pogodnim za naučene leme. Ona može biti definisana na različite načine, ali će biti pretpostavljeno da vraća `false` ukoliko konfliktna implikacija ne zadovoljava uslove u preduslovima pravila Backjump osim uslova $\mathcal{C} \in \Gamma$.

4.4.6 Saglasnost i potpunost apstraktnog sistema pravila i procedure `Calypso`

U ovom poglavlju se dokazuju osnovna svojstva apstraktnog sistema pravila i procedure `Calypso` — saglasnost i potpunost. Saglasnost sistema pravila znači da ukoliko postoji lanac koji počinje u polaznom stanju, a završava se u prihvatajućem, onda je razmatrano tvrđenje zaista logička posledica skupa aksioma. Kako procedura `Calypso` koristi samo pravila pomenutog sistema, njenom izvršavanju uvek odgovara neki lanac. Stoga je saglasnost ove procedure direktna posledica saglasnosti sistema pravila. Potpunost sistema pravila znači da za svaku koherentnu logičku posledicu skupa koherentnih aksioma postoji lanac čije je prvo stanje polazno, a poslednje prihvatajuće. Ukoliko je procedura `Calypso` potpuna, odnosno ukoliko svakom koherentnom tvrđenju koje je logička posledica skupa koherentnih aksioma odgovara jedan takav lanac, onda je potpun i sistem pravila. Stoga, kako bi saglasnost i potpunost sistema pravila i procedure `Calypso` bili dokazani, dovoljno je dokazati saglasnost sistema pravila i potpunost procedure.

Saglasnost

Dokaz saglasnosti sistema pravila i procedure `Calypso` se zasniva na tome da pravilo Explain izvodi logičku posledicu formula na koja se primenjuje. Ostala pravila nisu bitna za saglasnost sistema. Razlog za to je što se definicija prihvatajućeg stanja zasniva na tome da je tvrđenje logička posledica poslednje naučene leme, a ta lema je dobijena samo primenom pravila Explain na prethodno naučene leme i aksiome.

Lema 1 *Neka za koherentne formule $\mathcal{F}_1$, $\mathcal{F}_2$ i $\mathcal{R}$ važi $ResStep(\mathcal{F}_1, \mathcal{F}_2, \mathcal{R})$. Onda važi $\mathcal{F}_1, \mathcal{F}_2 \models \mathcal{R}$.*

Dokaz. Formula $\mathcal{R}$ se može dobiti primenom jednog od dva pravila koherentne rezolucije. Dokaz će biti izveden za slučaj primene prvog pravila. Dokaz za slučaj primene drugog pravila se izvodi analogno. Neka je $\mathcal{F}_1 = \mathcal{P}_1 \Rightarrow \mathcal{Q}_1 \cup \{\exists \vec{y}\ r(v, y)\}$ i $\mathcal{F}_2 = \{r(v', v'')\} \cup \mathcal{P}_2 \Rightarrow \mathcal{Q}_2$ gde su $\exists \vec{y}\ p(v, y)$ i $p(v', v'')$ pk-literali po kojima se vrši rezolviranje kojim se iz formula $\mathcal{F}_1$ i $\mathcal{F}_2$ dobija formula $\mathcal{R}$. Tada je $\mathcal{R} = \forall \vec{v} \forall \vec{v}'\ (\mathcal{P}_1\lambda \cup \forall \vec{v}''\ \mathcal{P}_2\lambda \Rightarrow \mathcal{Q}_1\lambda \cup \exists \vec{v}''\ \mathcal{Q}_2\lambda)$ za λ za koje važi $ResStep^m_\lambda(\mathcal{F}_1, \mathcal{F}_2, \mathcal{R})$, pri čemu je $m = [\exists \vec{y}\ p(v, y) \mapsto p(v', v'')]$.

Dovoljno je ustanoviti kontradiktornost formula $\forall \vec{v}\ (\mathcal{P}_1 \Rightarrow \mathcal{Q}_1 \vee \exists \vec{y}\ r(v, y))$, $\forall \vec{v}' \forall \vec{v}''\ (r(v', v'') \wedge \mathcal{P}_2 \Rightarrow \mathcal{Q}_2)$ i $\neg \forall \vec{v} \forall \vec{v}'\ (\mathcal{P}_1\lambda \wedge \forall \vec{v}''\ \mathcal{P}_2\lambda \Rightarrow \mathcal{Q}_1\lambda \vee \exists \vec{v}''\ \mathcal{Q}_2\lambda)$, odnosno formula $\forall \vec{v}\ (\neg \mathcal{P}_1 \vee \mathcal{Q}_1 \vee \exists \vec{y}\ r(v, y))$, $\forall \vec{v}' \forall \vec{v}''\ (\neg r(v', v'') \vee \neg \mathcal{P}_2 \vee \mathcal{Q}_2)$ i $\exists \vec{v} \exists \vec{v}'\ (\mathcal{P}_1\lambda \wedge$

$\forall\vec{v}''\ \mathcal{P}_2\lambda \wedge \neg\mathcal{Q}_1\lambda \wedge \forall\vec{v}''\ \neg\mathcal{Q}_2\lambda$). Dokaz će biti sproveden pobijanjem pomoću klauzalne rezolucije. Dovoljno je dokazati da je sledeći skup klauza nezadovoljiv:

$$\neg p_1^1(\vec{v}, \vec{f}_p^1(\vec{v})), \ldots, \neg p_{m_1}^1(\vec{v}, \vec{f}_p^1(\vec{v})), q_1^1(\vec{v}, \vec{f}_q^1(\vec{v})), \ldots, q_{n_1}^1(\vec{v}, \vec{f}_q^1(\vec{v})), r(\vec{v}, \vec{f}_q^1(\vec{v}))$$
$$[\neg p_1^2(\vec{v}', \vec{v}'', \vec{f}_p^2(\vec{v}', \vec{v}'')), \ldots, \neg p_{m_2}^2(\vec{v}', \vec{v}'', \vec{f}_p^2(\vec{v}', \vec{v}'')),$$
$$q_1^2(\vec{v}', \vec{v}'', \vec{f}_q^2(\vec{v}', \vec{v}'')), \ldots, q_{n_2}^2(\vec{v}', \vec{v}'', \vec{f}_q^2(\vec{v}', \vec{v}'')), \neg r(\vec{v}', \vec{v}'')]$$
$$p_1^1(\vec{v}\lambda, \vec{x})[\vec{v} \mapsto \vec{c}]$$
$$\vdots$$
$$p_{m_1}^1(\vec{v}\lambda, \vec{x})[\vec{v} \mapsto \vec{c}]$$
$$\vdots$$
$$p_1^2(\vec{v}'\lambda, \vec{v}'', \vec{x}')[\vec{v}' \mapsto \vec{c}']$$
$$\vdots$$
$$p_{m_2}^2(\vec{v}'\lambda, \vec{v}'', \vec{x}')[\vec{v}' \mapsto \vec{c}']$$
$$\vdots$$
$$\neg q_1^1(\vec{v}\lambda, \vec{v}'')[\vec{v} \mapsto \vec{c}]$$
$$\vdots$$
$$\neg q_{n_1}^1(\vec{v}\lambda, \vec{v}'')[\vec{v} \mapsto \vec{c}]$$
$$\vdots$$
$$\neg q_1^2(\vec{v}'\lambda, \vec{v}'', \vec{y}')[\vec{v}' \mapsto \vec{c}']$$
$$\vdots$$
$$\neg q_{n_2}^2(\vec{v}'\lambda, \vec{v}'', \vec{y}')[\vec{v}' \mapsto \vec{c}']$$

pri čemu je prva klauza dobijena tranformisanjem prve formule, druga klauza transformisanjem druge formule, dok ostale klauze potiču od treće i pri čemu su $\vec{f}_p^1$, $\vec{f}_q^1$, $\vec{f}_p^2$ i $\vec{f}_q^2$ nizovi funkcija dobijenih skolemizacijom u prve dve formule, a $\vec{c}$ i $\vec{c}'$ nizovi konstanti dobijeni skolemizacijom u trećoj formuli.

Literali $r(\vec{v}, \vec{f}_q^1(\vec{v}))$ i $\neg r(\vec{v}', \vec{v}'')$ se mogu rezolvirati pri unifikatoru $\lambda[\vec{v}'' \mapsto \vec{f}_q^1(\vec{v})]$. Rezolviranjem prve dve klauze po tim literalima dobija se klauza:

$$\neg p_1^1(\vec{v}\lambda, \vec{f}_p^1(\vec{v}\lambda)), \ldots, \neg p_{m_1}^1(\vec{v}\lambda, \vec{f}_p^1(\vec{v}\lambda)), q_1^1(\vec{v}\lambda, \vec{f}_q^1(\vec{v}\lambda)), \ldots, q_{n_1}^1(\vec{v}\lambda, \vec{f}_q^1(\vec{v}\lambda)),$$
$$\neg p_1^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}), \vec{f}_p^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}))), \ldots, \neg p_{m_2}^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}), \vec{f}_p^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}))),$$
$$q_1^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}), \vec{f}_q^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}))), \ldots, q_{n_2}^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v}), \vec{f}_q^2(\vec{v}'\lambda, \vec{f}_q^1(\vec{v})))$$

Izvedena rezolventa se može rezolvirati sa jednočlanim klauzama do prazne klauze, čime je dokaz završen. □

Lema 2 *Ako za koherentne formule $\mathcal{F}$ i $\mathcal{F}'$ i preslikavanje m važi $FactStep(\mathcal{F}, m, \mathcal{F}')$, onda važi $\mathcal{F} \models \mathcal{F}'$.*

Dokaz. Formula $\mathcal{F}'$ se može dobiti od formule $\mathcal{F}$ faktorisanjem po skupu literala u levoj ili u desnoj strani formule $\mathcal{F}$. Dokaz će biti izveden za prvi slučaj, a za drugi slučaj se izvodi analogno.

Za razliku od dosadašnje prakse da $p(\vec{x}, \vec{y})$ označava atom sa predikatskim simbolom p koji uključuje neke od promenljivih iz nizova $\vec{x}$ i $\vec{y}$, u nastavku dokaza će u slučaju predikatskog simbola r (i samo u tom slučaju) uvek biti podrazumevani tačno oni argumenti koje pk-literal sa tim predikatskim simbolom uključuje.

Neka je $\mathcal{F} = \mathcal{S} \cup \mathcal{P} \Rightarrow \mathcal{Q}$ formula sa slobodnim promenljivim $\vec{v}$, pri čemu je $\mathcal{S} = \{\forall \vec{x}_1\ r(\vec{v}_1, \vec{x}_1), \ldots, \forall \vec{x}_k\ r(\vec{v}_k, \vec{x}_k)\}$ gde su $\vec{v}_i \subseteq \vec{v}$ i neka je $\mathcal{F}' = \{\forall \vec{x}'\ r(\vec{v}'_r, \vec{x}_r)\} \cup \mathcal{P}\lambda \Rightarrow \mathcal{Q}\lambda$ za $\mathcal{S}$ i λ za koje važi $FactStep^{\mathcal{S}}_{\lambda}(\mathcal{F}, m, \mathcal{F}')$, pri čemu je $\forall \vec{x}'\ r(\vec{v}', \vec{x}')$ pk-literal koji zamenjuje $\mathcal{S}$ u formuli dobijenoj faktorisanjem. Onda važi $\forall \vec{x}_i\ r(\vec{v}_i, \vec{x}_i)\lambda = \forall \vec{x}_i\ r(\vec{v}'_r, \vec{x}_i)$ za svako $i = 1, \ldots, k$.

Da bi se dokazalo tvrđenje, dovoljno je dokazati kontradiktornost formula $\forall \vec{v}\ (\mathcal{S} \wedge \mathcal{P} \Rightarrow \mathcal{Q})$ i $\neg \forall \vec{v}'\ (\forall \vec{x}'\ r(\vec{v}', \vec{x}') \wedge \mathcal{P}\lambda \Rightarrow \mathcal{Q}\lambda)$, odnosno formula $\forall \vec{v}\ (\neg \mathcal{S} \vee \neg \mathcal{P} \vee \mathcal{Q})$ i $\exists \vec{v}' (\forall \vec{x}'\ r(\vec{v}', \vec{x}') \wedge \mathcal{P}\lambda \wedge \neg \mathcal{Q}\lambda)$. Za ovo je dovoljno pokazati klauzalnom rezolucijom da je sledeći skup klauza nezadovoljiv:

$$
\begin{gathered}
[\neg r(\vec{v}_1, \vec{f}^{\,1}_p(\vec{v})), \ldots, \neg r(\vec{v}_k, \vec{f}^{\,k}_p(\vec{v})), \neg p_1(\vec{v}, \vec{f}_p(\vec{v})), \ldots, \\
\neg p_m(\vec{v}, \vec{f}_p(\vec{v})), q_1(\vec{v}, \vec{f}_q(\vec{v})), \ldots, q_n(\vec{v}, \vec{f}_q(\vec{v}))] \\
r(\vec{c}_r, \vec{x}_r) \\
p_1(\vec{c}, \vec{x}) \\
\vdots \\
p_m(\vec{c}, \vec{x}) \\
\vdots \\
\neg q_1(\vec{c}, \vec{y}) \\
\vdots \\
\neg q_n(\vec{c}, \vec{y})
\end{gathered}
$$

pri čemu je prva klauza dobijena transformisanjem formule $\mathcal{F}$ i pri čemu su literali sa predikatskim simbolom r dobijeni transformisanjem podformule $\mathcal{S}$, a ostale klauze su dobijene transformisanjem formule $\mathcal{F}'$. Pri tome su $\vec{f}_p$ i $\vec{f}_q$ nizovi funkcija dobijeni skolemizacijom u prvoj formuli, $\vec{c}$ konstante dobijene skolemizacijom u drugoj formuli. Za svako $i = 1, \ldots, k$, za nizove promenljivih $\vec{v}_i$ važi $\vec{v}_i \subseteq \vec{v}$, a za nizove funkcija $\vec{f}^{\,i}_p$ važi $\vec{f}^{\,i}_p \subseteq \vec{f}_p$. Takođe važi $\vec{c}_r \subseteq \vec{c}$.

Na osnovu definicije relacije $FactStep$, pošto su $\vec{c}$ međusobno različite promenljive koje nisu u signaturi, važi $ResStep_{\mu}(r(\vec{c}_r, \vec{x}_r), \overline{\mathcal{S}}\lambda, \bot)$ za neke supstitucije μ i λ takve da μ uključuje samo zamene promenljivih konstantama iz $\vec{c}$, a u λ ne figurišu konstante iz $\vec{c}$. To znači da se $r(\vec{c}_r, \vec{x}_r)$ može koherentno rezolvirati sa svakim od literala $\overline{\forall \vec{x}_i\ r(\vec{v}_i, \vec{x}_i)}$ pri unifikatoru $\lambda\mu$, odnosno da, za svako

i, važi $\overline{\forall \vec{x}_i\ r(\vec{v}_i, \vec{x}_i)}\lambda\mu = \overline{\forall \vec{x}_i\ r(\vec{c}_r, \vec{x}_i)}$. Odatle sledi da se $r(\vec{c}_r, \vec{x}')$ i $\neg r(\vec{v}_i, \vec{f}_p^i(\vec{v}))$ mogu rezolvirati pri unifikatoru $\lambda\mu\nu_i$, odnosno važi $r(\vec{c}_r, \vec{x}')\lambda\mu\nu_i = r(\vec{c}_r, \vec{f}_p^i(\vec{v}))$ i $\neg r(\vec{v}_i, \vec{f}_p^i(\vec{v}))\lambda\mu\nu_i = \neg r(\vec{c}_r, \vec{f}_p^i(\vec{v}))$. Odatle sledi da se $r(\vec{c}, \vec{x}')$ može rezolvirati sa svim $\neg r(\vec{v}_i, \vec{f}_p^i(\vec{v}))$ pri unifikatoru $\lambda\mu\nu$ gde je $\nu = \nu_1 \dots \nu_k$. Imajući u vidu da promenljivih $\vec{x}'$ na koje deluje supstitucija ν nema u prvoj klauzi, odatle sledi da se rezolviranjem $r(\vec{c}, \vec{x}')$ sa prvom klauzom dobija rezolventa

$$(\neg p_1(\vec{v}, \vec{f}_p(\vec{v})), \dots, \neg p_m(\vec{v}, \vec{f}_p(\vec{v})), q_1(\vec{v}, \vec{f}_q(\vec{v})), \dots, q_n(\vec{v}, \vec{f}_q(\vec{v})))\lambda\mu$$

odnosno

$$(\neg p_1(\vec{v}', \vec{f}_p(\vec{v}')), \dots, \neg p_m(\vec{v}', \vec{f}_p(\vec{v}')), q_1(\vec{v}', \vec{f}_q(\vec{v}')), \dots, q_n(\vec{v}', \vec{f}_q(\vec{v}')))\mu$$

Pošto na osnovu za svako $i = 1, \dots, k$ važi $\forall \vec{x}_i\ p(\vec{v}_i, \vec{x}_i)\lambda\mu = \forall \vec{x}_i\ p(\vec{v}'_r, \vec{x}_i)\mu = \forall \vec{x}_i\ p(\vec{c}_r, \vec{x}_i)$, zaključuje se da važi $\mu = [\vec{v}'_r \mapsto \vec{c}_r]$, odnosno da supstitucija μ slika neke od promenljivih $\vec{v}'$ u one konstante iz $\vec{c}$ koje su im pridružene prilikom skolemizacije (kojom su uvedene konstante $\vec{c}$). Na osnovu toga, iz ove klauze se rezolviranjem sa ostalim jediničnim klauzama pri unifikatoru $[\vec{v}' \mapsto \vec{c}, \vec{x} \mapsto \vec{f}_p(\vec{v})]$ može izvesti prazna klauza. □

Lema 3 *Neka za koherentne formule $\mathcal{F}$ i $\mathcal{F}'$ i preslikavanje m važi $Fact(\mathcal{F}, m, \mathcal{F}')$. Onda važi $\mathcal{F} \models \mathcal{F}'$.*

Dokaz. Neka je $n = |m(\mathcal{F})|$. Dokaz će biti izveden indukcijom po n. Neka je $n = 1$ i neka je $\{l_c\} = m(\mathcal{F})$. Ukoliko je $|m^{-1}(l_c)| = 1$, tada na osnovu definicije relacija $Fact$ i $FactStep$ važi $\mathcal{F} = \mathcal{F}'$, pa je tvrđenje dokazano. Ukoliko je $|m^{-1}(l_c)| > 1$, tada na osnovu definicije relacije $Fact$ važi $FactStep(\mathcal{F}, m, \mathcal{F}')$. Na osnovu leme 2, tvrđenje važi.

Neka tvrđenje važi za $n \leq k$ i neka je $n = k + 1$. Neka je $l_c \in m(\mathcal{F})$. Ukoliko je $|m^{-1}(l_c)| = 1$, tada na osnovu definicije relacija $Fact$ i $FactStep$ važi $\mathcal{F} = \mathcal{F}'$, pa je tvrđenje dokazano. Ukoliko je $|m^{-1}(l_c)| > 1$, tada na osnovu definicije relacije $Fact$ važi $FactStep(\mathcal{F}, m, \mathcal{F}'')$ i $Fact(\mathcal{F}'', m \circ f^{-1}, \mathcal{F}')$, pri čemu je f^{-1} funkcija uparivanja pri faktorisanju formule $\mathcal{F}$. Na osnovu leme 2, tvrđenje važi $\mathcal{F} \models \mathcal{F}''$. Na osnovu induktivne hipoteze i i $Fact(\mathcal{F}'', m \circ f^{-1}, \mathcal{F}')$, važi $\mathcal{F}'' \models \mathcal{F}'$. Stoga, važi i $\mathcal{F} \models \mathcal{F}'$, čime je dokaz završen. □

Lema 4 *Neka za koherentne formule $\mathcal{F}$ i $\mathcal{F}'$ i neka preslikavanja m i m' važi $ResGen(\mathcal{F}, \mathcal{F}', m, m', \mathcal{R})$. Onda važi $\mathcal{F}, \mathcal{F}' \models \mathcal{R}$.*

Dokaz. Tvrđenje važi na osnovu definicije relacije $ResGen$ i lema 1 i 3. □

Sledeća teorema izražava saglasnost sistema pravila, odnosno da ukoliko za neki skup aksioma i neko tvrđenje postoji lanac sa završnim stanjem, onda je tvrđenje logička posledica skipa aksioma. Osnovna ideja dokaza je da u završnom stanju postoji stablo u čijem se svakom čvoru nalazi koherentna formula, pri čemu je u korenu poslednja izvedena lema i pri čemu je svaka formula dobijena rezolucijom nad prethodno izvedenim formulama i aksioma. Kako je pravilo koherentnog rezolviranja saglasno, ovo stablo predstavlja rezolucijski dokaz za poslednju izvedenu lemu. Potom se pokazuje je polazno tvrđenje logička posledica izvedene leme.

Teorema 4 (Saglasnost sistema pravila) *Ukoliko za skup koherentnih aksioma* $\mathcal{AX}$ *i koherentno tvrđenje* $\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$ *važi* $\mathcal{AX} \vdash_{CL} \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$*, onda važi* $\mathcal{AX} \models \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$.

Dokaz. Ukoliko važi $\mathcal{AX} \vdash_{CL} \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}^0(\vec{v},\vec{y})$, po definiciji relacije $\vdash_{CL}$, postoji stanje S takvo da važi $S_0 \xrightarrow{*} S$, $\mathcal{C}_2 \neq \{no\_cflct\}$, $Res^m(\mathcal{C}, M, \bot)$ i $m(\mathcal{C}) \subseteq \mathcal{PR}(\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}^0(\vec{v},\vec{y}))$. Formula $\mathcal{C}$ je izvedena pomoću konačnog broja primena pravila Explain na osnovu prethodno izvedenih rezolventi (uključujući naučene leme) i aksioma. Stoga, za lemu $\mathcal{C}$ postoji binarno stablo $T_{\mathcal{C}}$, takvo da je lema $\mathcal{C}$ u korenu i deci N_1 i N_2 svakog čvora N, odgovaraju formule $\mathcal{F}_1$ i $\mathcal{F}_2$ na osnovu kojih je primenom pravila Explain dobijena formula $\mathcal{F}$ koja odgovara čvoru N. Kako je na osnovu leme 4 rezolventa logička posledica formula od kojih je dobijena, jednostavnom indukcijom po dubini stabla $T_{\mathcal{C}}$ se može dokazati da važi $\mathcal{AX} \models \mathcal{C}$. Da bi se dokazala teorema, dovoljno je dokazati da važi $\mathcal{C} \models \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$, odnosno da važi $\mathcal{C} \models (\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y}))\sigma$ za proizvoljnu baznu supstituciju σ koja promenljivim $\vec{v}$ dodeljuje međusobno različite simbole konstanti. Kako je supstitucija λ iz definicije 30 proizvoljno izabrana takva supstitucija, dovoljno je izvesti dokaz za $\sigma = \lambda$. Dovoljno je dokazati da važi $\mathcal{C}, \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x})\lambda \models \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})\lambda$, odnosno da važi $\mathcal{C}, \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x})\lambda, \neg\exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})\lambda \models \bot$, odnosno $\mathcal{C}, \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x})\lambda, \forall\vec{y}\ \neg\mathcal{G}(\vec{v},\vec{y})\lambda \models \bot$, za šta je dovoljno pokazati da je sledeći skup klauza nezadovoljiv:

$$\neg p_1(\vec{v'}, \vec{f_p}(\vec{v'})), \ldots, \neg p_k(\vec{v'}, \vec{f_p}(\vec{v'})), q_1(\vec{v'}, \vec{f_q}(\vec{v'})), \ldots, q_l(\vec{v'}, \vec{f_q}(\vec{v'}))$$
$$h_1(\vec{v},\vec{x})\lambda$$
$$\vdots$$
$$h_m(\vec{v},\vec{x})\lambda$$
$$\vdots$$
$$\neg g_1(\vec{v},\vec{y})\lambda$$
$$\vdots$$
$$\neg g_n(\vec{v},\vec{y})\lambda$$

pri čemu je prva klauza dobijena transformisanjem konfliktne implikacije, a ostale transformisanjem preostale dve formule i pri čemu su nizovi funkcija $\vec{f_p}$ i $\vec{f_q}$ dobijeni skolemizacijom u konfliktnoj implikaciji. Na osnovu uslova $m(\mathcal{C}) \subseteq \mathcal{PR}(\forall \vec{x}\, \mathcal{H}(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\, \mathcal{G}^0(\vec{v}, \vec{y}))$, iz ovog skupa klauza se može izvesti prazna klauza. □

Direktna posledica ove teoreme je teorema o saglasnosti procedure `Calypso`.

Teorema 5 (Saglasnost procedure `Calypso`) *Ukoliko za skup koherentnih aksioma $\mathcal{AX}$ i koherentno tvrđenje $\forall \vec{x}\ \mathcal{H}(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v}, \vec{y})$ procedura* `Calypso` *vrati vrednost* `true`*, onda važi $\mathcal{AX} \models \forall \vec{x}\ \mathcal{H}(\vec{v}, \vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v}, \vec{y})$.*

Potpunost

Zarad prezentacije dokaza potpunosti, prvo se uvode definicije pojmova i oznaka koji nisu bili neophodni za prezentaciju sistema, ali su potrebni u dokazu potpunosti. Prefiks liste L koji uključuje sve elemente koji prethode elementu e se označava L^e. Ukoliko važi $e \notin L$, onda je $L^e = L$. Sva pravila osim pravila Limiter se nazivaju *osnovnim pravilima*. Skup svih simbola konstanti iz skupa Σ^∞ sa indeksima manjim ili jednakim δ se naziva skupom *dopustivih konstanti* i označava Υ. U kontekstima u kojima se referiše na više različitih stanja, da bi se naglasila veza između elementa stanja i stanja, oznaka stanja će se dodavati na poziciji indeksa. Na primer, M_S označava listu M u stanju S. Vrednost funkcije $reasonFormula(l, S)$ je vrednost polja `reasonFormula[l]` u proceduri `Calypso` u stanju S. `Calypso` lanac je lanac stanja koji odgovara izvršavanju procedure `Calypso`. Naredna definicija predstavlja jednostavno proširenje relacije tačnosti.

Definicija 40 (Relacija posledice: $\Vdash$) *Pk-literal l je* posledica *liste L, što se zapisuje $L \vdash l$ ukoliko postoji pk-literal l' u listi L, takav da važi jedan od sledećih uslova*

- $l = p(\vec{v})$ *i* $ResStep(l', \overline{p(\vec{v})\sigma}, \bot)$
- $l = \exists \vec{y}\ p(\vec{y})$ *i* $ResStep(l', \overline{p(\vec{y})}, \bot)$
- $l = \overline{p(\vec{v})}$ *i* $ResStep(l', p(\vec{v})\sigma, \bot)$
- $l = \forall \vec{x}\ \overline{p(\vec{x})}$ *i* $ResStep(l', p(\vec{x}), \bot)$

pri čemu je u oba slučaja σ supstitucija koja promenljivim $\vec{v}$ dodeljuje međusobno različite simbole konstanti koji nisu u signaturi.

Definicija 41 *Neka je $S_0 \rightarrow \ldots \rightarrow S$ lanac i neka je $M_S = M_{S_0}\ M'$. Lista M_S je* kontradiktorna *ukoliko postoje pk-literali $l_1 \in M$ i $l_2 \in M'$ takvi da važi*

$ResStep(l_1, l_2, \bot)$. *Za listu* M_S *kažemo da je* ispravna *ukoliko nije kontradiktorna i ne sadrži ponavljanja pk-literala.*

Primetimo da je uslov definiciji kontradiktornosti da jedan pk-literal ne pripada M_{S_0} tehnička pogodnost koja vodi elegantnijim formulacijama kasnijih lema. U slučaju da su oba međusobno kontradiktorna pk-literala iz M_{S_0}, S_0 je prihvatajuće stanje, pa se u njemu procedura `Calypso` zaustavlja.

Lema 5 *Neka je* $C = S_0 \to \ldots \to S$ `Calypso` *lanac. Tada je lista* M_S *ispravna.*

Dokaz. Tvrđenje će biti dokazano indukcijom po broju prelaza n od S_0 do S. Neka je $n = 0$. Lista M_0 po definiciji kontradiktornosti ne može biti kontradiktorna. Dodatno, M_0 po definiciji stanja S_0 sadrži k-literale iz skupa $\mathcal{PR}(\Phi)$, zbog čije skupovne prirode ne može sadržati ponovljene pk-literale. Neka tvrđenje važi za $n \leq k$ za neko k i neka važi $n = k + 1$. Neka je S' stanje iz kojeg je S dobijeno primenom pravila r. Tada prema induktivnoj pretpostavci u stanju S, lista M je ispravna. Ukoliko je r neko od pravila Conflict, Explain, Learn ili Limiter, stvrđenje važi pošto ta pravila ne menjaju listu M. Ukoliko je r pravilo Decide ili Unit propagation, uslovi tih pravila zabranjuju umetanje pk-literala koji su u njoj već sadržani ili čine listu M kontradiktornom. Ako je r pravilo Intro, neka je l k-literal dodat u listu M tim pravilom i neka je $\exists * l' \in M$ pk-literal iz kojeg je l izveden. Pošto je lista $M_{S'}$ ispravna, nema pk-literala $l'' \in M_{S'}$ takvog da važi $ResStep(l'', \exists * l', \bot)$ ili $l'' \Vdash \exists * l'$. Ukoliko bi važilo $ResStep(l'', l, \bot)$, onda bi važilo i $ResStep(l'', \exists * l', \bot)$, što je kontradikcija. Dodatno, uslov pravila Intro ne dozvoljava dodavanje k-literala koji je već prisutan u list M. Kako je l jedini k-literal dodat u listu M u prelazu iz S' u S, lista M_S je ispravna. Neka je r pravilo Backjump, neka je l pk-literal koji se primenom pravila Backjump dodaje u listu M, l' k-literal iz konfliktnog skupa koji je uklonjen primenom pravila Backjump i neka je t nivo liste M_S. Prema poslednja dva uslova pravila Backjump, lista M_S je neprotivrečna, a prema definiciji notacije $L\ e$, kad važi $e \in L$, važi $L\ e = L$, pa stoga pravilo Backjump ne može u listu dodati pk-literal koji je već prisutan u listi. Time je tvrđenje dokazano. □

Lema 6 *Neka je* $S_0 \to \ldots \to S \to \ldots$ `Calypso` *lanac u kojem posle stanja* S *nema prelaza pomoću pravila Backjump i Limiter. Tada je broj prelaza uz pomoć pravila Decide, Intro i Unit propagation konačan.*

Dokaz. Kako nakon stanja S nema prelaza pomoću pravila Limiter niti Backjump, skup Υ je u narednim stanjima konstantan i skup konstanti koje mogu figurisati u literalima je konačan. Skup predikata Π, nad kojim se grade formule, je takođe

konačan. Zahvaljujući tome što se formule koje se razlikuju samo u preimenovanju promenljivih smatraju jednakim, broj pk-literala nad skupom predikata Π i konstanti Υ je konačan. Kako na osnovu leme 5, lista M ne može sadržati ponovljene pk-literale, svi pk-literali u njoj moraju biti različiti, što znači da je dužina liste M u stanjima koja slede S ograničena. Kako nakon stanja S nema primena pravila Backjump, lista M se ne može smanjivati, postoji poslednje stanje u lancu u kojem dolazi do izmene liste M. Neka je to stanje S'. Kako posle stanja S' nema izmena liste M, to znači da posle njega nema prelaza pomoću pravila Decide, Intro, i Unit propagation, čime je tvrđenje dokazano. □

Naredna lema tvrdi da se, bez korišćenja povratnih skokova i proširivanja skupa dopustivih konstanti, analiza konflikta mora završiti u konačnom broju koraka.

Lema 7 *Neka je* $C = S_0 \to \ldots \to S \to \ldots$ `Calypso` *lanac i neka u njemu posle stanja* S *nema prelaza primenom pravila Backjump i Limiter. Tada, je broj prelaza upotrebom pravila Conflict, Explain i Learn konačan.*

Dokaz. Kako je C `Calypso` lanac, ukoliko posle S nema primene pravila Backjump, nema ni primene pravila Learn, pošto iza primene tog pravila uvek sledi primena pravila Backjump.

Na osnovu leme 6, sva ostala pravila, osim eventualno Conflict i Explain se mogu primeniti samo konačno mnogo puta nakon stanja S. Neka je S' stanje nakon kojeg nema primena pravila osim eventualno pravila Conflict i Explain. Sva stanja nakon S' imaju istu listu $M = M_{S'}$. Ukoliko nakon S' nema primena ni ovih pravila, tvrđenje je dokazano. Pretpostavimo suprotno. Jedino pravilo koje postavlja $\mathcal{C}$ na no_cflct je pravilo Backjump. Stoga, ukoliko se pravilo Conflict primeni nakon S, nije primenljivo u narednim stanjima. Odnosno, pravilo Conflict je primenljivo najviše jednom. Ukoliko u narednim stanjima nije primenljivo pravilo Explain, tvrđenje je dokazano. U suprotnom, u nekom od narednih stanja važi $\mathcal{C} \neq no\_cflct$. Neka je $CS_1, CS_2, \ldots$ niz konfliktnih skupova u narednim stanjima koja se dobijaju primenom pravila Explain. Pri svakoj primeni pravila Explain, neki k-literal se uklanja iz konfliktnog skupa i potencijalno se dodaju drugi k-literali koji mu prethode u listi M. Iskoristićemo ovo svojstvo kako bismo pokazali da je niz $CS_1, CS_2, \ldots$ konačan i time, da ima konačno mnogo primena pravila Explain nakon stanja S.

Neka je $index_M(l)$ indeks (koji polazi od 0) pk-listerala $l \in M$. Označimo

$$\gamma(L) = \sum_{l \in L} 2^{index_M(l)}$$

Pošto pri svakoj primeni pravila Explain dolazi do uklanjanja k-literala iz konfliktnog skupa i dodavanja k-literala nižeg indeksa i pošto važi $1 + 2 + \ldots + 2^n < 2^{n+1}$,

zaključujemo da važi $\gamma(CS_{i+1}) < \gamma(CS_i)$. Kako je $\gamma(L)$ nenegativan broj, može biti svega konačno mnogo primena pravila Explain nakon stanja S. □

Lema 8 *Neka je $C = S_0 \to \ldots \to S \to \ldots$ lanac bez prihvatajućeg stanja, neka u stanju S važi $\mathcal{C} \neq no\_cflct$ i neka je pravilo Explain neprimenljivo. Tada je u stanju S primenljivo pravilo Learn ili pravilo Backjump.*

Dokaz. Ako važi $\mathcal{C} \neq \Gamma$, pravilo Learn je primenljivo. Pretpostavimo važi $\mathcal{C} \in \Gamma$. Pošto stanje S nije prihvatajuće, postoji k-literal l u konfliktnom skupu koji nije među inicijalnim k-literalima $\mathcal{PR}(\Phi)$. Neka je njegov nivo n. Pretpostavimo da je $n = 0$. Tada l nije dodat u M primenom pravila decide. Ukoliko je l dodat u M primenom pravila Unit propagation ili Backjump, onda postoji $reasonFormula(l, S)$ takva da je pravilo Explain primenljivo, što je kontradikcija. Ukoliko je dodat primenom pravila Intro, onda postoji pk-literal l' na koji je pravilo intro primenjeno. Ovaj pk-literal ne može biti iz $\mathcal{PR}(\Phi)$ jer sadrži egzistencijalni kvantifikator. To znači da je dodat pravilom Unit propagation ili Backjump i da postoji $reasonFormula(l', S)$ takva da se na nju i k-literal l može primeniti pravilo Explain, što je kontradikcija. Stoga, mora važiti $n > 0$.

Pretpostavimo da postoje dva k-literala u $m(\mathcal{C})$ na nivou n. Pošto ne mogu oba biti odlučeni k-literali, jedan od njih to nije. Onda je morao biti dodat u M nekim od pravila Intro, Unit propagation ili Backjump i za njega se istim rezonovanjem kao u prošlom slučaju izvodi da je pravilo Explain primenljivo što je kontradikcija. Stoga, na nivou n liste M postoji tačno jedan k-literal iz $m(\mathcal{C})$. U tom slučaju važi $Res(\mathcal{C}, M^{n-1}, l')$ za neki pk-literal l', čime su uslovi za primenu pravila Backjump ispunjeni. □

Naredne definicije uvode poredak na skupu listi pk-literala. Ova relacija će biti korišćena za dokaživanje da je broj promena koje se mogu učiniti nad listom pk-literala M u toku izvršavanja procedure `Calypso` konačan.

Definicija 42 *Neka su M_1 i M_2 liste pk-literala. Neka je $\bar{l}$ prvi negativni kvantifikovani literal u M_2 takav da postoji $l' \in M_1$ takav da važi $ResStep(\bar{l}, l', \bot)$. Pišemo $M_1 <_t M_2$ ukoliko važi $M_2^{\bar{l}} \subseteq M_1^{l'}$. Za stanja S i S' pišemo $S <_s S'$ ukoliko važi $M_S <_s M_{S'}$.*

Naredna lema trivijalno sledi iz prethodne definicije i definicije pravila Backjump.

Lema 9 *Ukoliko važi $S \xrightarrow{Backump} S'$, onda važi $S <_s S'$.*

Lema 10 *Neka je $\mathcal{M}$ skup svih listi pk-literala u svim stanjima S takvim da važi $S_0 \to \ldots \to S$. Na skupu $\mathcal{M}$, relacija $<_t$ je tranzitivna, nerefleksivna i aciklična.*

Dokaz. Tranzitivnost relacije $<_t$ je trivijalna. Pretpostavimo da važi $M <_t M$ za neku listu $M \in \mathcal{M}$. Tada je M kontradiktorna lista što nije moguće prema lemi 5, što znači da je relacija $<_t$ nerefleksivna. Pretpostavimo da postoji niz $M_1, M_2, \ldots$ takav da važi $M_1 <_t \ldots <_t M_k < M_1$. Tada, na osnovu tranzitivnosti važi $M_1 <_t M_1$, što je kontradikcija sa nerefleksivnošću relacije $<_t$. Ovo dokazuje da je relacija $<_t$ aciklična. Svojstva relacije $<_s$ slede iz svojstava relacije $<_t$. □

Naredna lema tvrdi da će konstanta koja je trajno prisutna u signaturi od nekog stanja postati dopustiva.

Lema 11 *Neka je* $C = S_0 \to \ldots \to S \to \ldots$ `Calypso` *lanac bez prihvatajućeg stanja. Ukoliko u stanju* S *i u svim narednim stanjima važi* $c \in \Sigma$*, tada je u stanju* S *ili u nekom od narednih stanja konstanta* c *dopustiva i ni jedno osnovno pravilo nije primenljivo.*

Dokaz. Pretpostavimo da za neku konstantu važi $c^k \in \Sigma$ i $\delta < k$ u stanju S i svim narednim stanjima u lancu C. Na osnovu lema 9, 10 i konačnosti skupa listi pk-literala nad ograničenim skupom dopustivih konstanti $\Upsilon \subseteq \Sigma_S^{c^k}$ i predikata Π, može biti konačno mnogo primena pravila Backjump nakon stanja S. Neka je S' stanje iz lanca C dobijeno poslednjom primenom pravila Backjump nakon stanja S ili ukoliko takvo stanje ne postoji neka važi $S' = S$. Pošto nema primene pravila Backjump nakon stanja S'. δ se ne može smanjivati. Pošto važi $\delta < k$ postoji poslednje stanjeu kojem je pravilo Limiter primenjeno i neka je to stanje S'' ukoliko je to stanje nakon S', a $S'' = S'$ u suprontom. Nakon stanja S'' nema primena pravila Backjump i Limiter. Na osnovu lema 6 i 7 postoji konačno mnogo stanja u C nakon stanja S'' u kojima je primenljivo bilo koje osnovno pravilo. U pravom stanju nakon S'' u kojem nije primenljivo ni jedno osnovno pravilo, primenljivo je pravilo Limiter jer je u tom stanju $\delta < k \leq \max\{i \mid c^i \in \Sigma\}$, što je kontradikcija. Stoga, u stanju S ili u nekom od narednih stanja važi $c^k \in \Upsilon$. Označimo to stanje sa S_1. Pretpostavimo da nakon tog stanja nema stanja u kojem ni jedno osnovno pravilo nije primenljivo. To znači da ni u jednom sledećem stanju, pravilo Limiter nije primenljivo i da se δ ne menja. Stoga se ni Υ ne menja. Kao i ranije, izvodi se zaključak da mora postojati neko naredno stanje u kojem je pravilo Limiter primenljivo što je kontradikcija. Stoga tvrđenje važi. □

Naredna definicija uvodi važnu strukturu koja će služiti kao model za skup aksioma i kontramodel za tvrđenje prilikom dokazivanja potpunosti procedure `Calypso`.

Definicija 43 (Erbranov univerzum i Erbranova struktura za lanac) *Neka je* C *lanac i neka je* L *skup svih pk-literala* l *za koje postoji stanje* S *u lancu* C *takvo da važi* $l \in M$ *u stanju* S *i u svim stanjima iz* C *posle njega.* Erbranov univerzum

Σ_C za lanac C, je skup svih simbola konstanti koje se pojavljuju u atomima iz L. Erbranova struktura *$\mathcal{E}_C$ za lanac C, je skup baznih atoma, nad konstantama iz Σ_C i predikatima iz Π takvih da je bazni atom l u $\mathcal{E}_C$ ako i samo ako važi $L \Vdash l$. Bazni atom je tačan u Erbranovoj strukturi za lanac C ako i samo ako joj pripada.*

Naredna tema tvrdi da ukoliko su premise aksiome zadovoljene atomima iz liste M u nekom stanju i u svim narednim stanjima u lancu, onda postoji stanje takvo da su zaključci aksiome takođe zadovoljeni u tom i narednim stanjima.

Lema 12 *Neka je $C = S_0 \to \ldots \to S' \to \ldots$ `Calypso` lanac bez prihvatajućeg stanja. Neka u stanju S' važi $\mathcal{P} \Rightarrow \mathcal{Q} \in \Gamma$ i neka je σ bazna supstitucija nad slobodnim promenljivim formule $\mathcal{P} \Rightarrow \mathcal{Q}$ i konstantama iz Σ_C. Neka u stanju S' i svim narednim stanjima važi $l \uparrow$ za svako $l \in \mathcal{P}\sigma$. Tada postoji k-literal $\exists * l \in \mathcal{Q}\sigma$ i supstitucija σ' nad nekim slobodnim promenljivim iz l i konstantama iz Σ_C, tako da od nekog stanja, u svim stanjima iz C važi $l\sigma' \uparrow$.*

Dokaz. Po definiciji Erbranovog univerzuma Σ_C, mora postojati stanje u lancu C takvo da su sve konstante iz supstitucije σ u skupu Σ u tom stanju i svim narednim stanjima iz lanca C. Na osnovu leme 11 u lancu C mora postojate stanje S nakon S' tako da su sve konstante koje figurišu u supstituciji σ dopustive i ni jedno osnovno pravilo nije prihvatljivo i neka je S prvo takvo stanje u C počevši od S'.

Prvo dokazujemo da u stanju S i svim narednim stanjima u kojima ni jedno osnovno pravilo nije primenljivo, mora postojati pk-literal $l' \in M$ bez egzistencijalnih kvantifikatora takav da važi $l' \Vdash \exists * l$ za neki k-literal $\exists * l \in \mathcal{Q}\sigma$. Pošto pravilo Decide nije primenljivo, za svaki k-literal $\exists * l \in \mathcal{Q}\sigma$ u tom stanju važi ili $\exists * l \uparrow$ ili $\exists * l \downarrow$. Ukoliko važi $\exists * l \downarrow$ za svaki k-literal $\exists * l \in \mathcal{Q}\sigma$, onda u tom stanju mora da važi i $\mathcal{C} \neq no\_cflct$ jer bi u suprotnom bilo primenljivo pravilo Conflict. Takođe mora da važi i $\mathcal{C} \in \Gamma$, inače bi bilo primenljivo pravilo Learn. Pošto pravilo Explain nije primenljivo, na osnovu leme 8, pravilo Learn ili pravilo Backjump mora biti primenljivo što je kontradikcija. Stoga, mora da važi $\exists * l \uparrow$ za neki k-literal $\exists * l \in \mathcal{Q}\sigma$. Kako su sve konstante iz $\exists * l$ dopustive (pošto su sve konstante iz supstitucije σ dopustive u stanjima lanca C počevši od S) i kako je pravilo Intro neprimenljivo, mora postojati k-literal $l' \in M$ bez egzistencijalnih kvantifikatora takav da važi $l' \Vdash \exists * l$.

Neka je $\mathcal{S}$ niz $S^1, S'^1, S^2, S'^2, \ldots$ svih stanja takvih da važi

- $S^1 = S$,
- u stanju S^i ni jedno osnovno pravilo nije primenljivo (i stoga važi $\exists * l \uparrow$ za neki k-literal $\exists * l \in \mathcal{Q}\sigma$),

- S'^i je prvo stanje nakon S^i takvo da ni za jedan k-literal $\exists * l \in \mathcal{Q}\sigma$ ne postoji k-literal $l' \in M$ bez egzistencijalnih kvantifikatora takav da važi $l' \Vdash \exists * l$,
- S^i je za $i > 1$ prvo stanje nakon S'^{i-1} takvo da ni jedno osnovno pravilo nije primenljivo.

Primetimo da su sva stanja S'^i dobijena primenom pravila Backjump pošto je to prvo stanje u kojem je uklonjen određeni k-literal, što je moguće samo usled primene pravila Backjump.

Za sva stanja iz niza $\mathcal{S}$ ćemo dokazati da važi $\Upsilon \subseteq \Upsilon_S$ (gde je Υ skup dopustivih konstanti). To po definicji niza $\mathcal{S}$ važi za stanje S^1. Neka tvrđenje važi za stanje S^i. Tada $\delta_{S'^i}$ ne može biti veće od δ_{S^i} Stanje S'^i je dobijeno primenom pravila Backjump koje postavlja vrednost δ na poslednju vrednost u Δ^m gde je m nivo povratnog skoka. Kako je ovim pravilom izbačen iz liste M k-literal koji je u njoj u stanju S^i, m mora biti manje nego nivo u stanju S^i. Kako su vrednosti u Δ neopadajuće, važi $\delta_{S'^i} \leq \delta_{S^i}$, odnosno $\Upsilon_{S'^i} \subseteq \Upsilon_S$. Neka tvrđenje važi za stanje S'^j. Pošto je S^{j+1} prvo stanje nakon S'^j u kojem ni jedno osnovno pravilo nije primenljivo, mora biti $\delta_{S^{j+1}} = \delta_{S'^j}$ pa važi $\Upsilon_{S^{j+1}} \subseteq \Upsilon_S$. Zaključujemo da važi $\Upsilon \subseteq \Upsilon_S$ u svakom stanju niza $\mathcal{S}$

U nastavku pokazujemo da je niz $\mathcal{S}$ konačan. Pretpostavimo da je niz $\mathcal{S}$ beskonačan. Neka je l'_i pk-literal ubačen u M od strane pravila Backjump kojim je dobijeno stanje S'^i. Pk-literal l'_i je ili pristuan u svim narednim stanjima ili biva uklonjen primenom pravila Backjump u nekom od sledećih stanja. Ukoliko biva uklonjen pravilom Backjump, drugi pk-literal se umeće na niži nivo. Neka je l'^j_i niz pk-literala takvih da je $l'^1_i = l'_i$ i l'^{j+1}_i je pk-literal ubačen primenom pravila Backjump kojom je uklonjen pk-literal l'^j_i. Za svako i i j važi da je nivo u listi M pk-literala l'^{j+1}_i manji od nivoa pk-literala l'^j_i. Međutim, kako su nivoi u listi M ograničen odozdo, u svakom od nizova, mora da postoji poslednji pk-literal l''_i koji nikad ne biva uklonjen. Kako je $\mathcal{S}$ beskonačan i ponovljeni pk-literali se, prema lemi 5, ne dodaju u M, niz l''_i za $i = 1, \ldots$ mora imati beskonačno mnogo različitih pk-literala. Pošto se oni nikad ne uklanjaju iz liste M, postoji stanje S'^{k_i} takvo da je l''_i u listi M u stanju S'^{k_i} i ostalima. Pokazaćemo da je ovo kontradiukcija se činjenicom da važi $\Upsilon_{S'^i} \subseteq \Upsilon_S$ za svako $i = 1, \ldots$. Neka je $K_i = |\Sigma_{S_i}|$. Kako se sve nove konstante koje uvodi pravilo Intro dodaju na kraj liste Σ i maksimalni nivo u stanju S'_i je manji od nivoa u stanju S_i, važi $|\Sigma_{S'_i}| \leq K_i$ za svako $i = 1, \ldots$. Broj pk-literala nad Υ_S je konačan i broj konstanti koje se mogu dobiti primenom pravila Intro nad tim pk-literalima je konačan. Stoga, broj konstanti koje se mogu pojaviti u Σ_{S_i} je ograničen nekim brojem K za $i = 1, \ldots$. Tada takođe važi $|\Sigma_{S'^i}| \leq K$ za svako $i = 1, \ldots$. Neka je N ukupan broj pk-literala sa predikatskim simbolom iz

Π koji se mogu konstruisati nad bilo kojim skupom konstanti veličine K. U stanju $S'^{k_{N+1}}$ postoji bar $N+1$ različiti pk-literal, što je kontradikcija. Otud niz $\mathcal{S}$ mora biti konačan.

U poslednjem stanju niza $\mathcal{S}$ mora da važi $\exists * l \uparrow$ za neki k-literal $\exists * l \in \mathcal{Q}\sigma$. Poslednje stanje mora biti jedno od stanja S^i, a ne S'^i, inače bi na osnovu leme 11 postojao još jedan element niza iza tog stanja, što je nemoguće. Neka je l' prvi k-literal bez egzistencijalnih u listi M za koji važi $l' \Vdash \exists * l$. Ovaj k-literal neće više biti uklanjan iz liste M jer bi u suprotnom postojalo još jedno stanje S'^i iza poslednjeg stanja niza $\mathcal{S}$, što je nemoguće. Primetimo da to znači da su sve konstante iz l' u Σ_C. Kako su l i l' pozitivni atomi bez kvantifikatora, na osnovu $l' \Vdash \exists * l$ mora da važi $l'\sigma' = l\sigma'$ gde je σ' najopštiji unifikator za l' i l. Pošto je σ' najopštiji unifikator, on ne sadrži konstante koje nisu prisutne u l' i l. Oba skupa konstanti su iz Σ_C, pa je σ' supstitucija nad Σ_C, čime je tvrđenje dokazano. $\square$

Naredna teorema izražava potpunost procedure `Calypso`, odnosno da ukoliko je neka formula logička posledica skupa aksioma, onda se izvršavanje procedure `Calypso` zaustavlja u prihvatajućem stanju. Osnovna ideja dokaza je da se pretpostavi suprotno, da se procedura ne zaustavlja, na osnovu čega se pokazuje da postoji Erbranova struktura u kojoj su sve aksiome tačne, ali tvrđenje nije. Kako je tvrđenje po pretpostavci logička posledica aksioma, ovo je kontradikcija i zaključuje se da se procedura mora zaustaviti.

Teorema 6 (Potpunost procedure `Calypso`) *Ukoliko za skup koherentnih aksioma $\mathcal{AX}$ i koherentno tvrđenje $\forall \vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v},\vec{y})$ važi $\mathcal{AX} \models \forall \vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists \vec{y}\ \mathcal{G}(\vec{v},\vec{y})$, onda u svakom `Calypso` lancu postoji prihvatajuće stanje.*

Dokaz. Pretpostavimo suprotno. Neka u nekom `Calypso` lancu C ne postoji prihvatajuće stanje. Takav lanac je ili beskonačan ili se završava odbacujućim stanjem. U nastavku se pokazuje da ni jedan od ova dva slučaja ne može važiti i da stoga teorema važi.

Neka je $\mathcal{E}_C$ Erbranova struktura za lanac C. Neka važi $\mathcal{P} \Rightarrow \mathcal{Q} \in \mathcal{AX}$ i neka je konjunkcija $\mathcal{P}\sigma$ tačna u $\mathcal{E}_C$ za neku baznu supstituciju σ nad slobodnim promenljivim iz $\mathcal{P} \Rightarrow \mathcal{Q}$ i konstantama iz Σ_C. Ukoliko je konjunkcija $\mathcal{P}\sigma$ tačna u $\mathcal{E}_C$, tada postoji stanje S u C, takvo da u njemu i u svim narednim stanjima iz C važi $l \uparrow$ za svako $l \in \mathcal{P}\sigma$. Tada, na osnovu leme 12, postoji naredno stanje S' u C, takvo da za neko $\exists * l \in \mathcal{Q}\sigma$ postoji supstitucija σ' nad Σ_C za koju važi $l\sigma' \uparrow$ u stanju S' i svim narednim stanjima. Na osnovu toga, disjunkcija $\mathcal{Q}\sigma$ je tačna u strukturi $\mathcal{E}_C$. Odatle, struktura $\mathcal{E}_C$ je model za aksiome iz $\mathcal{AX}$.

Razmotrimo taňost tvrđenja u strukturi $\mathcal{E}_C$. Pošto su svi kvantifikovani literali iz konjunkcije $\mathcal{H}$ u L, konjunkcija $\mathcal{H}$ je tačna u strukturi $\mathcal{E}_C$. Potrebno je dokazati

da disjunkcija $\mathcal{G}$ nije tačna u ovoj strukturi. Ukoliko važi $\mathcal{G} = \bot$, onda disjunkcija $\mathcal{G}$ nije tačna u strukturi $\mathcal{E}_C$. Ukoliko važi $\mathcal{G} \neq \bot$, pretpostavimo da je $\mathcal{G}$ is tačno u strukturi $\mathcal{E}_C$. Tada postoji stanje S u lancu C i kvantifikovani atom l takav da u stanju S i stanjima posle njega važi $l \in M$ i $l \Vdash \mathcal{G}$. Na osnovu toga što se u inicijalizaciji u listi M dodaju atomi takvi da važi $ResStep(l, l', \bot)$ koji u njoj ostaju u svim narednim stanjima i leme 5 zaključuje se da je ovo kontradikcija.

Ovo znači da je $\mathcal{E}_C$ model za skup aksioma $\mathcal{AX}$, ali ne i za formulu $\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$ što je nemoguće jer je pretpostavljeno da važi $\mathcal{AX} \models \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$. Na osnovu toga, nije moguće da postoji Calypso lanac bez prihvatajućeg stanja za tvrđenje $\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$. □

Direktna posledica ove teoreme je potpunost sistema pravila.

Teorema 7 (Potpunost sistema pravila) *Ukoliko za skup koherentnih aksioma $\mathcal{AX}$ i koherentno tvrđenje $\forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$ važi $\mathcal{AX} \models \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$, onda važi i $\mathcal{AX} \vdash_{CL} \forall\vec{x}\ \mathcal{H}(\vec{v},\vec{x}) \Rightarrow \exists\vec{y}\ \mathcal{G}(\vec{v},\vec{y})$.*

4.5 Dokazivač Calypso

U ovom poglavlju će biti opisani detalji implementacije dokazivača Calypso. Najvažniji deo implementacije su struktura i algoritam koji se koriste za za otkrivanje konflikata i k-literala koji se mogu propagirati ili odlučeni. Biće dato i poređenje dokazivača Calypso sa drugim dokazivačima.

4.5.1 Implementacija

Dokazivač Calypso je implementiran u programskom jeziku C++. Najvažniji implementacioni detalji se odnose na sledeće probleme:

- izbor k-literala u pravilu Decide,
- nalaženje pk-literala koji se mogu propagirati pravilom Unit propagation,
- detekcija konflikta,
- uslovi za dodavanje konfliktne implikacije u skup formula kao naučene leme i
- izbor unifikatora pri rezolviranju u pravilu Explain,

Odgovor na prva četiri problema je u potpunosti ili delimično vezan za modifikaciju Rete algoritma koji je već korišćen u dokazivaču clp [40] i biće dat kroz opis i razmatranje ovog algoritma.

Tabela 4.1: Elementi čvorova tipa α i njihovi tipovi.

Element	Tip
`literal`	referenca na pk-literal
`unifiers`	niz unifikatora
`children`	niz referenci na čvorove tipa β

Tabela 4.2: Elementi čvorova tipa β i njihovi tipovi.

Element	Tip
unifikatori	niz unifikatora
`a-parent`	referenca na čvor tipa α
`b-parent`	referenca na čvor tipa β
`child`	referenca na čvor tipa β

Modifikovani Rete algoritam. Rete algoritam je algoritam za unifikaciju pretpostavki aksiome sa atomima u bazi znanja korišćen u dokazivaču `clp` za standardnu koherentnu logiku [33, 40]. Njegov osnovni kvalitet je što ne ponavlja pokušaje unifikacije sa istim činjenicama. Algoritam pokušava da unifikuje pretpostavke aksiome sa atomima u bazi znanja polazeći od prvog atoma u formuli i proširujući parcijalni unifikator za prvih $k-1$ atoma pretpostavki svim saglasnim unifikatorima k-tog atoma sa atomima iz baze znanja. Ovaj algoritam je proširen tako da radi u proširenom fragmentu koherentne logike za koji je formulisan apstraktni sistem, unifikuje i levu i desnu stranu aksiome i ima mogućnost ažuriranja pri povratku u pretrazi.

Neka je L skup svih pk-literala l koji se javljaju u desnoj strani neke formule iz Γ i pk-literala $\bar{l}$ takvih da se l javlja u levoj strani neke formule iz Γ. Ovaj skup može biti proširivan prilikom učenja lema. Neka svakom od pk-literala iz L odgovara jedan čvor tipa α. *Čvor tipa* α je struktura čiji su elementi dati u tabeli 4.1. Neka je $\mathcal{F} = l_1 \wedge \ldots l_k \Rightarrow l_{k+1} \vee \ldots \vee l_m$ formula iz skupa Γ. Formuli $\mathcal{F}$ odgovara niz β čvorova dužine $m+1$. *Čvor tipa* β je struktura čiji su elementi dati u tabeli 4.2. Tabela 4.3 prikazuje strukturu unifikator. Uloge elemenata pomenutih struktura će biti opisane kasnije. *Rete mreža* je neusmereni graf (V,E) gde je V skup α čvorova koji odgovaraju pk-literalima u formulama iz Γ i β čvorova koji odgovaraju formulama iz Γ. E je skup grana. Postoji grana između u i v ukoliko su u i v susedni čvorovi tipa β u nizu koji odgovara nekoj formuli iz Γ. U tom slučaju se čvor sa manjim indeksom u nizu naziva roditeljem čvora sa većim indeksom, a čvor sa većim indeksom detetom čvora sa manjim indeksom. Takođe postoji grana između u i v ukoliko je u čvor tipa α koji odgovara pk-literalu l_i iz neke formule $\mathcal{F}$ i v je i-ti β čvor u nizu koji odgovara formuli $\mathcal{F}$. U tom slučaju se α čvor naziva roditeljem odgovarajućeg β čvora, a β čvor detetom tog α čvora.

Tabela 4.3: Elementi strukture unifikator i njihovi tipovi.

Element	Tip
`equalities`	niz jednakosti
`a-parent`	referenca na unifikator
`b-parent`	referenca na unifikator
`literal`	referenca na pk-literal
`delete`	{`true`,`false`, `undef`}

Struktura α čvora sadrži referencu na pk-literal iz formule sa kojim je povezana, niz unifikatora sa k-literalima iz liste M i niz referenci na decu čvorove tipa β. Struktura β čvora sadrži niz parcijalnih unifikatora za deo formule koji prethodi pk-literalu kojem čvor odgovara, referencu na α čvor koji odgovara istom pk-literalu i referencu na prethodni i naredni β čvor. Struktura unifikator sadrži niz jednakosti, reference na unifikatore čijim spajanjem je unifikator nastao, referencu na pk-literal na osnovu kojeg je nastao α-roditelj i indikator koji se koristi tokom brisanja unifikatora prilikom vraćanja u pretrazi.

Prilikom učenja lema, Rete mreža se proširuje čvorovima i granama koji odgovaraju naučenoj lemi. Prilikom ubacivanja k-literala u listu M, o tome se obaveštava Rete mreža pozivom funkcije `notify` koja je data na slici 4.3, a koja poziva funkciju `insert` koja je data na slici 4.4. Funkcija `clashUnifier` kao argumente uzima dva pk-literala l i l' i vraća najopštiju supstituciju λ za koju važi $ResStep_\lambda(l, l', \bot)$. Ukoliko to ne važi ni za jednu supstituciju λ, funkcija `clashUnifier` vraća `null`.

Funkcija `signalConflict` signalizira da je primećen konflikt. Na osnovu člana `b-parent` mogu se naći svi preci unifikatora koji figuriše u konfliktu. Kako taj unifikator i svi njegovi preci imaju član `literal`, mogu se pronaći svi k-literali iz liste M koji učestvuju u konfliktu.

Funkcija `signalUnitPropagate` signalizira da je moguće propagirati pk-literal, osim ukoliko se on ili opštiji pk-literal već nalaze u listi M. Kao i u slučaju konflikta, moguće je naći sve k-literale iz liste M koji su zaslužni za propagaciju. Ovako definisan algoritam je u stanju da pronađe jedinične propagacije samo ukoliko je se propagacija vrši na osnovu posldenjeg pk-literala u formuli. Algoritam je moguće proširiti tako da nalazi sve jedinične propagacije, ali u eksperimentima cena nalaženja propagacija se pokazala jednakom dobitku, pa to proširenje neće biti prikazano.

Rete mreža omogućava i nalaženje k-literala za primenu pravila Decide. Funkcija `getDecideLiteral` koja nalazi takav literal je prikazana na slici 4.5. Pri tome funkcija `findAcceptableInstance` kao argumente uzima pk-literal l i vraća k-literal l' koji je instanca pk-literala l, takva da važi l'? ukoliko takva instanca postoji, a `null` ukoliko takva instanca ne postoji.

```
function notify(l : literal) : void
begin
    foreach a in alphaNodes do
    begin
        u := clashUnifier(a.literal, l)
        if u = null then
            return
        a.unifiers := a.unifiers u
        foreach c in a.children do
        begin
            if c.parent = null then
                unifiers := [null]
            else
                unifiers := c.unifiers
            foreach pu in unifiers do
            begin
                mu := mergeUnifiers(pu, u)
                if mu != null then
                    insert(c.child, mu)
            end
        end
    end
end
```

Slika 4.3: Funkcija koja obaveštava mrežu o novom k-atomu koji je dodat u listu M.

Na slici 4.6 prikazana je funkcija `backtrack(lvl : integer)` koja ažurira Rete mrežu pri povratku u pretrazi. Uklanjaju se svi unifikatori pri čijem je dobijanju učestvovao k-literal sa nivoa liste M koji je manji od nivoa na koji se vrši povratak.

Izbor unifikatora pri rezolviranju. Za razliku od klauzalne rezolucije kod koje je izbor unifikatora pri rezolviranju trivijalan problem (uvek se bira najopštiji) kod koherentne rezolucije izbor unifikatora igra važnu ulogu. Primera radi, ako su date formule $p(x) \wedge q(x,y) \Rightarrow \bot$, $\exists y\, p(y)$ i $q(y,x)$ i prva se rezolvira sa drugom pri praznom unifikatoru, dobija se rezolventa $\forall \vec{x}\ q(x,y) \Rightarrow \bot$. Dalje rezolviranje sa trećom formulom je nemoguće. Međutim, ukoliko se rezolviranje uradi pri unifikatoru $[y \mapsto x]$, dobija se rezolventa $\forall \vec{x}\, q(x,x) \Rightarrow \bot$, koja se može rezolvirati sa trećom formulom. Stoga je izbor unifikatora važan. U nastavku je opisan njegov izbor.

Sledeća definicija uvodi funkciju eliminacije simbola iz supstitucije.

Definicija 44 (Funkcija eliminacije simbola: *elim*) *Neka je data supstitucija* μ *i skup simbola* $S = \{s_1, \ldots, s_n\}$ *i neka važi* $\mu = \mu_1\mu_2[x_{11} \mapsto s_1, \ldots, x_{1m_1} \mapsto s_1, \ldots, x_{n1} \mapsto s_n, \ldots, x_{nm_n} \mapsto s_n]$*, pri čemu u* μ_1 *nema simbola iz* S*,* μ_2 *deluje samo*

```
function insert(b : betaNode, u : unifier) : void
begin
    foreach bu in b.unifiers do
    begin
        if bu.equalities = u.equalities then
            if level(u.literal) < level(bu.literal) then
            begin
                bu.a-parent = u.a-parent
                bu.b-parent = u.b-parent
                bu.literal = u.literal
                break
            end
    end

    b.unifiers := b.unifiers u

    if b.child = null then
    begin
        signalConflict()
        return
    end

    merged := false

    foreach au in b.a-parent.unifiers do
    begin
        mu := mergeUnifiers(u, au)
        if mu != null then
        begin
            merged := true
            insert(b.child, mu)
        end
    end

    if not merged and b.child.child = null then
        signalUnitPropagate()
end
```

Slika 4.4: Funkcija koja propagira novi unifikator.

na simbole iz S i simboli x_{ij} nisu iz S ni za jedno i i j. Neka je $\lambda = \mu_1[x_{12} \mapsto x_{11}, \ldots, x_{1m_1} \mapsto x_{11}, \ldots, x_{n2} \mapsto x_{n1}, \ldots, x_{nm_n} \mapsto x_{n1}]$. Supstitucija λ se dobija od supstitucije μ eliminacijom simbola *is skupa S, što se zapisuje $\lambda = elim(\mu, S)$.*

Primer 16 *Važi $[z \mapsto x, u \mapsto v] = elim([x \mapsto a, y \mapsto b, z \mapsto a, u \mapsto v], \{a, y\})$.*

Neka su $\mathcal{F}_1$ i $\mathcal{F}_2$ formule takve da važi $ResGen_{\mu}^{\mathcal{S}\mapsto l}(\mathcal{F}_1, \mathcal{F}_2, m_1, m_2, \mathcal{R})$ pri čemu je μ najopštija supstitucija za koju data relacija važi. Neka je l' pk-literal za koji

```
function getDecideLiteral() : literal
begin
    foreach f in Gamma do
    begin
        m := length(f)
        for i := m - 1 to 0 do
        begin
            b := getBetaNode(f,i)
            foreach u in b.unifiers do
            begin
                l := applyUnifier(f[i],u)
                l' := findAcceptableInstance(l)
                if l' != null then
                    return l'
            end
        end
    end
end
```

Slika 4.5: Funkcija koja bira literal za pravilo Decide.

važi $l' \in \mathcal{F}_1$ i $l' \notin \mathcal{S}$ ili $l' \in \mathcal{F}_2$ i $l' \neq l$. Neka je $l'' = m(l')$. Neka je I skup svih indeksa na kojima se u k-literalu l'' nalaze slobodne promenljive. Neka je λ najopštiji unifikator za skup jednakosti $\{l'_i = l''_i \mid i \in I\}$, neka je $C = \{l''_i \mid i \in I\}$ i neka je $\lambda' = elim(\lambda, C)$. Neka je $\lambda = \mu\lambda_1 \cdots \lambda_n$, pri čemu su λ_i sve supstitucije koje odgovaraju pk-literalima koji zadovoljavaju uslov za l'. Unifikator koji se bira pri rezolviranju je upravo λ. Na ovaj način se obezbeđuje da se rezolventa može rezolvirati sa istim konfliktnim literalima sa kojima se mogu rezolvirati formule od kojih je nastala.

4.5.2 Generisanje dokaza

Dokazivač Calypso je u stanju da generiše dokaze u sistemu koherentne rezolucije zapisane u TPTP formatu[3]. Ovaj format je jedan od standardnih formata za zapisivanje dokaza u logici prvog reda i između ostalog se koristi u takmičenjima dokazivača za logiku prvog reda. Pogodnost korišćenja ovog formata je mogućnost automatske provere generisanih dokaza pomoću sistema za proveru javno dostupnog na TPTP sajtu[4]. Svaka linija dokaza predstavlja formulu koja može biti aksioma ili rezolventa uz koju se čuva informacija o tome na osnovu koje dve prethodne formule

[3]Više detalja o ovom formatu se može naći na adresi http://www.cs.miami.edu/~tptp/TPTP/QuickGuide/Derivations.html

[4]http://www.cs.miami.edu/~tptp/cgi-bin/SystemOnTSTP

```
function backtrack(lvl : integer) : void
begin
    foreach a in alphaNodes do
        backtrack(a, lvl)
    foreach b in betaNodes do
        backtrack(b, lvl)
end

function backtrack(a : alphaNode, lvl : integer) : void
begin
    foreach u in a.unifiers do
        if level(u.literal) > lvl then
            remove(a.unifiers, u)
end

function backtrack(b: betaNode, lvl : integer) : void
begin
    foreach u in b.unifiers do
        if u.delete = true or checkToDelete(u, lvl) = true then
            remove(b.unifiers, u)
end

function checkToDelete(u : unifier, lvl : integer) : {true, fale, undef}
begin
    if u = null then
        return false
    if u.delete != undef then
        return u.delete
    u.delete := checkToDelete(u.b-parent, lvl)
    if u.delete = false then
        if level(u.literal) > lvl then
            u.delete = true
    return u.delete
end
```

Slika 4.6: Funkcije za ažuriranje Rete mreže pri povratku u pretrazi.

je dobijena. Dokazi se konstruišu tako što se prilikom rada dokazivača pri svakoj primeni pravila explain uz novu konfliktnu implikaciju čuvaju pokazivači na formule iz kojih je ona nastala. Pri tome, iako se mnoge od konfliktnih implikacija koje se izvode u analizi konflikta ne dodaju u skup formula, one se čuvaju u memoriji kako bi se na kraju mogao rekonstruisati dokaz. Na kraju rada dokazivača, dokaz se generiše jednostavnim obilaskom binarnog stabla u čijem korenu je korenu poslednja konfliktna implikacija.

4.5.3 Poređenje sa drugim sistemima

Dokazivač Calypso je upoređen sa drugim relevantnim sistemima. Ti sistemi su koherentni dokazivači CL, clp i Geo opisani u glavi 4.2 i najefikasniji rezolucijski dokazivač Vampire [86]. Dokazivači su pokrenuti na skupu od 66 problema koji je korišćen i u ranijim istraživanjima [40]. Za svaki problem, dat je skup aksioma i tvrđenje koje je potrebno dokazati. U ovim problemima, aksiomatski sistemi sadrže i do 30000 aksioma. Prilikom dokazivanja, korišćeno je vremensko ograničenje od 300 sekundi. Meren je broj dokazanih teorema i prosečno vreme dokazivanja. Broj dokazanih teorema za svaki od dokazivača, dat je u tabeli 4.4. Iz tabele se vidi

Tabela 4.4: Broj dokazanih teorema i prosečno vreme izvršavanja u sekundama za svaki od dokazivača.

Dokazivač	CL	clp	Geo	Vampire	Calypso
Dokazanih	43	50	55	57	58
Vreme (s)	103.6	73.5	50.7	47.7	40.7

da je Calypso najefikasniji dokazivač i po broju dokazanih teorema i po vremenu izvršavanja. Kako je za sve dokazivače većina dokazanih teorema dokazana za manje od jedne sekunde, na vreme izvršavanja najviše utiče broj nerešenih instanci, pa stoga ove mere kvaliteta prenose sličnu informaciju. Treba imati u vidu da se radi o osnovnoj implementaciji sistema Calypso koja ostavlja puno prostora za dalja poboljšanja poput heuristika navođenja i implementacionih trikova od kojih se očekuje dalje ubrzanje sistema, dok su Geo i Vampire dokazivači na kojima je duže rađeno i koji su učestvovali i na takmičenjima dokazivača za logiku prvog reda.

Glava 5

Zaključci i dalji rad

U ovoj glavi se sumiraju rezultati prikazani u radu i izvode zaključci rada. Potom se skiciraju pravci daljeg rada kako u oblasti prilagodljivog rešavanja SAT problema, tako i u oblasti dokazivanja teorema u koherentnoj logici.

5.1 Zaključci

U ovom radu je razmatran problem usmeravanja pretrage u automatskom dokazivanju teorema. Fokus je bio na CDCL sistemu pretrage za koji je pokazano da se njegova efikasnost može značajno unaprediti jednostavnim navođenjem, ali takođe i da se može uopštiti za bogatije logike od iskazne za koju je polazno formulisan.

U prvom delu rada razmatran je problem jednostavnog usmeravanja pretrage — izborom rešavača, njegovih heuristika i njihovih parametara, a u zavisnosti od svojstava instance koju je potrebno rešiti. Osnova predloženih metoda za izbor algoritama je sintaksna sličnost formula koja se odražava na njihovu grafovsku strukturu. Ova sličnost je prvi put pouzdano ustanovljena i analizirana pomoću originalne mere sličnosti grafova (koja se pokazala korisnom i u drugim domenima). Praktični pristupi merenju sličnosti formula se zbog računske efikasnosti ipak zasnivaju na numeričkim atributima iskaznih formula. Predložene su dve jednostavne metode izbora algoritma zasnovane na algoritmu k najbližih suseda. Prva tehnika, `ArgoSmArT`, se zasniva na klasifikaciji instance u jednu od unapred zadatih familija za koje su poznati algoritmi koji ih efikasno rešavaju. Instanca se rešava algoritmom koji odgovara familiji u koju je instanca klasifikovana. Druga tehnika, `ArgoSmArT k-NN`, se zasniva na nalaženju nekoliko sličnih instanci u trening skupu za koje je poznato vreme rešavanja pomoću svih algoritama iz portfolija. Instanca se rešava algoritmom koji se najbolje ponaša na pronađenim instancama. Tehnika `ArgoSmArT` je pogodnija za izbor konfiguracije SAT rešavača, a `ArgoSmArT k-NN` za izbor samog SAT

rešavača. Tehnika `ArgoSmArT k-NN` se pokazala značajno efikasnijom od najvažnijeg i pritom vrlo složenog sistema za izbor SAT rešavača — sistema `SATzilla`. Pored problema izbora KNF SAT rešavača i njihovih heuristika, razmatran je i problem izbora ne-KNF SAT rešavača u kojem fokus nije bio na tehnikama izbora rešavača, pošto se predložene tehnike direktno primenjuju i na taj problem, već na atributima kojima se ne-KNF instance mogu opisati, a koji do sad nisu predloženi. Rezultati u ovom domenu su pozitivni, ali za sada ograničeni. Osnovni razlog za to je nedostatak većeg broja ne-KNF rešavača raznovrsnog ponašanja, što ne iznenađuje s obzirom da je ova vrsta rešavača tek u svom povoju.

Pored konstrukcije efikasnog portfolija za SAT problem, prikazana je i metodologija poređenja SAT rešavača zasnovana na statističkom testiranju hipoteza. Potreba za ovakvom metodologijom proizilazi iz velike varijacije vremena rešavanja jedne formule od strane jednog SAT rešavača, što može dovesti do različitih redosleda SAT rešavača prilikom poređenja njihovih performansi ili rangiranja, što je i eksperimentalno demonstrirano. Predložena metodologija pruža ocenu statističke značajnosti testiranja i ocenu veličine efekta, poput verovatnoće da jedan SAT rešavač bude brži od drugog.

Drugi deo rada se odnosi na uopštavanje CDCL sistema pretrage na bogatije fragmente logike prvog reda. Predloženi sistem može predstavljati osnovu za efikasno dokazivanje u nekoj logici ukoliko je u njoj moguće definisati pravila rezolucije i faktorisanja. Ova pravila su definisana za jedno uopštenje koherentne logike. Za ovaj sistem su dokazana svojstva potpunosti i saglasnosti. Sistem ima nekoliko važnih karakteristika koje su posledica prethodno sprovedene analize izazova u dokazivanju u koherentnoj logici. Sistem omogućava rezonovanje prvog reda, umesto bazno što je karakteristika svih postojećih dokazivača za koherentnu logiku. Takođe, sistem koristi povratne skokove i učenje lema. Prilikom dizajna sistema, posebna pažnja je posvećena mogućnosti da se na osnovu rada dokazivača generišu čitljivi dokazi. Ova mogućnost je jedna od osnovnih prednosti koherentne logike, ali to nije lako postići pri korišćenju CDCL sistema pretrage. Jedno od svojstava ovog sistema proisteklo iz potrebe za čitljivim dokazima je očuvanje kvantifikatora prilikom dokazivanja i vrlo je nekarakteristično za postojeće CDCL sisteme. Takođe, prednost formulisanja CDCL sistema je mogućnost prenošenja heuristika koje su se već pokazale korisne u rešavanju SAT problema.

Nad predloženim sistemom, definisana je procedura `Calypso`, za dokazivanje u proširenoj koherentnoj logici, koju je moguće primeniti i na standardnu koherentnu logiku. Predloženo je proširenje algoritma Rete koje omogućava nalaženje literala koji se mogu propagirati, detekciju konflikata i predlaganje literala za pravilo De-

cide. Procedura `Calypso` je implementirana u jeziku C++. Evaluirana je na reprezentativnim problemima zapisanim u koherentnoj logici i na njima se pokazala superiornom u odnosu na druge dokazivače za koherentnu logiku, kao i dokazivač `Vampire`, najefikasniji dokazivač za logiku prvog reda.

Na osnovu rezultata izloženih u ovom radu, može se zaključiti da je potvrđena njegova osnovna teza — da se sistem pretrage koji se koristi u CDCL SAT rešavačima može značajno unaprediti jednostavnim usmeravnjem, a da se takođe može uspešno formulisati i za fragmente logike prvog reda kao što je koherentna logika, kao i da su dati konkretni odgovori na pitanje kako se to može uraditi.

5.2 Dalji rad

Postoji više pravaca daljeg rada. Što se tiče prilagodljivog SAT rešavanja, razvijene tehnike mogu biti upotrebljene u problemu predikcije vremena izvršavanja SAT rešavača. Tekući pristupi uključuju kompleksne sheme treniranja regresionih modela na velikim skupovima instanci. SAT rešavači mogu ispoljavati drastično različita ponašanja na različitim grupama instanci koje su zastupljene u okviru ovih korpusa. Očigledna mana postojećih pristupa je što se pomoću jednog regresionog modela pokušavaju pokriti evidentno različite zakonitosti. Sličnost instanci uočena u ovom radu, koja se automatski može prepoznati pomoću predloženih metoda klasifikacije visoke preciznosti, se može iskoristiti za particionisanje celog skupa instanci u manje podskupove međusobno sličnih instanci. Ovo može biti urađeno na osnovu unapred postojećeg znanja o familijama kojima instance pripadaju ili automatski — klasterovanjem. Na svakom od ovih podskupova moguće je trenirati jedan regresioni model. Zahvaljujući srodnosti ovih instanci, može se očekivati da će regresioni modeli imati značajno veću preciznost, ali se postavlja pitanje koji model treba primeniti za predviđanje vremena izvršavanja za datu instancu. Izbor modela se može izvršiti korišćenjem metoda klasifikacije predloženih u ovom radu.

Eksperimenti vezani za portfolio ne-KNF SAT rešavača su pokazali pozitivne rezultate, ali i dalje ne ubedljive kao u slučaju portfolija KNF SAT rešavača. Osnovni razlog za to je nedovoljna raznovrsnost ne-KNF SAT rešavača u ovom trenutku. Kako se razvoj SAT rešavača kreće prilično brzo, uskoro bi bilo moguće iznova ispitati potencijale ove ideje i, ako je potrebno, doraditi predložene atribute za ne-KNF instance.

Primena mere sličnosti grafova, predložene u svrhe analize sličnosti SAT instanci, je već dovela do značajnih rezultata u analizi studentskih zadataka [96]. Njene dalje primene su moguće i u drugim domenima.

U domenu dokazivanja teorema u koherentnoj logici, osnovni pravac daljeg rada je generisanje čitljivih dokaza, kako u jezicima, kao što je Isar, koji omogućuju mašinsku proveru dokaza, tako i u prirodnom jeziku. Kako bi se to realizovalo, potrebno je naći način da se na osnovu rada rešavača konstruišu dokazi u prirodnom deduktivnom sistemu koherentne logike zasnovanom na rezonovanju uapred, razmatranom u potpoglavlju 4.1.2, ili u nekom sličnom sistemu. Kako je dokaz u koherentnoj rezoluciji već dostupan, prirodno se postavlja pitanje mogućnosti ekstrakcije čitljivog dokaza iz dostupnog rezolucijskog dokaza. Ovde će, na primeru, biti skiciran planirani način na koji bi to moglo biti urađeno. Neka je dat skup aksioma (podrazumeva se da su sve aksiome univerzalno zatvorene):

(Ax1) $p(x,y) \wedge q(x,y) \Rightarrow \bot$
(Ax2) $s(x) \Rightarrow \exists y\ q(x,y)$
(Ax3) $s(x) \vee q(y,y)$

i tvrđenje $(\forall x \forall y\ p(x,y)) \Rightarrow \bot$. U nastavku je dat dokaz tvrđenja pomoću koherentne rezolucije, što je polazni dokaz, i stablo koje opisuje postupak izvođenja činjenica rezonovanjem unapred, što je struktura iz koje se može generisati čitljivi dokaz u prirodnom ili nekom formalnom jeziku. Pri tome se podrazumeva da se prilikom izvođenja svakog zaključka u stablu mogu koristiti sve činjenice izvedene duž grane od korena do trenutnog čvora. Grananja u stablu su rezultat postojanja disjunkcija u desnim stranama aksioma.

$$\dfrac{\dfrac{s(x) \vee q(y,y) \quad p(x,y) \wedge q(x,y) \Rightarrow \bot}{p(x,x) \Rightarrow s(z)} \quad \dfrac{s(x) \Rightarrow \exists y\ q(x,y) \quad p(x,y) \wedge q(x,y) \Rightarrow \bot}{\forall y\ p(x,y) \wedge s(x) \Rightarrow \bot}}{p(x,x) \wedge \forall y\ p(z,y) \Rightarrow \bot}$$

$$\dfrac{\dfrac{\dfrac{\bot}{q(y,y)} \Rightarrow (Ax1) \quad \dfrac{\dfrac{\dfrac{\dfrac{\bot}{q(a,b)} \Rightarrow (Ax1)}{\exists y\ q(a,y)} \exists}{\exists y\ q(x,y)} Inst}{s(x)} \Rightarrow (Ax2)}{\mathcal{AX}, p(x,y)} \vee (Ax3)$$

Izvedeno stablo se dobija iz rezolucijskog dokaza njegovim obilaskom u dubinu s leva nadesno. U korenu stabla su literali iz pretpostavki tvrđenja. Prva aksioma na koju se nailazi prilikom obilaska rezolucijskog dokaza je aksioma Ax3 i stoga se ona prva primenjuje, što dovodi do grananja u dokazu. Predak čvora koji odgovara

Ax3 je rezolventa koja se dobija od Ax3 i Ax1 (što je sledeći list koji se poseću je pri obilasku) po literalu $q(y, y)$, pa je stoga grana stabla u kojoj se generisanje nastavlja grana $q(y, y)$. Kako se vrši rezolviranje sa aksiomom Ax1, to znači da se ta aksioma primenjuje sledeća i izvodi se zaključak $\perp$ koji zatvara granu stabla. Potom se generiše druga grana stabla ukojoj se pretpostavlja da važi $s(x)$. Sledeći list koji se poseću je u obilasku rezolucijskog dokaza je aksioma Ax2 čijom se primenom dobija $\exists y\ q(x, y)$. Prilikom izvođenja egzistencijalno kvantifikovanih atoma, oni se instanciraju ako imaju slobodne promenljive i vrši se eliminacija egzistencijalnog kvantifikatora. Sledeći list koji se posećuje obilaskom rezolucijskog dokaza je aksioma Ax1, čijom primenom se izvodi $\perp$ što zatvara poslednju granu stabla. Generisanje čitljivog dokaza iz ove strukture bi trebalo da bude pravolinijski posao. Formalno opisivanje, dokazivanje korektnosti i implementacija ovog postupka su netrivijalani zadaci koji će biti obrađeni u daljem radu.

Predloženi sistem je dao osnovu za prenošenje heuristika koje su se pokazale uspešnim u rešavanju SAT problema u domen koherentne logike. Najvažnija heuristika je heuristika za izbor literala za pravilo Decide. Moderni SAT rešavači u ove svrhe koriste heuristike zasnovane na aktivnosti koje prate koliko često određeni literali učestvuju u konfliktima, jediničnim propagacijama itd. U slučaju rešavanja SAT problema, skup literala je konačan i poznat na početku rada rešavača. S druge strane, u toku rada dokazivača za koherentnu logiku mogu se pojavljivati literali koji nisu viđeni u ranijem koracima. Stoga se postavlja pitanje kako se mogu donositi heurističke pretpostavke o korisnosti literala za proces rezonovanja na osnovu podataka o prethodnom korišćenju drugih literala sa istim predikatskim simbolom ili literala koji su sadržali iste simbole konstanti. Takođe se postavlja pitanje prenošenja tehnika poput otpočinjanja iznova ili zaboravljanja naučenih lema. Zaboravljanje naučenih lema bi moglo da onemogući generisanje dokaza i stoga je u tom pitanju potrebna posebna pažnja.

Pored proširivanja mogućnosti apstraktnog sistema pravila za koherentnu logiku i unapređivanja dokazivača, u planu su i njegove primene u dokazivanju teorema u matematičkim disciplinama u cilju formalizacije matematičkog znanja. Najpogodnije teorije su geometrijske, zasnovane na aksiomama Hilberta, Tarskog i modernim varijantama u Euklidovih aksioma. Dokazivač `Calypso` se može upotrebiti kao alat za delimičnu automatizaciju napora formalizacije.

Bibliografija

[1] Michael Alekhnovich, Alexander A. Razborov, and Er A. Razborov. Resolution is not automatizable unless w[p] is tractable. In *In 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 210–219, 2001.

[2] J. Alfredsson. The SAT Solver kw. Technical report, Oepir Consulting, 2008.

[3] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Proceedings of the 15th international conference on Principles and practice of constraint programming*, pages 142–157. Springer, 2009.

[4] A. Arbelaez, Y. Hamadi, and M. Sebag. Online Heuristic Selection in Constraint Programming. In *International Symposium on Combinatorial Search - 2009*, 2009.

[5] Gilles Audemard and Laurent Simon. Experimenting a conflict-driven clause learning algorithm. In *14th International Conference on Principles and Practice of Constraint Programming(CP'08)*, pages 630–634. Lecture Notes in Computer Science (LNCS 5202), Springer, sep 2008.

[6] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Implementing the model evolution calculus. *International Journal of Artificial Intelligence Tools*, 15(1):21–52, 2006.

[7] Peter Baumgartner, Björn Pelzer, and Cesare Tinelli. Model evolution with equality – revised and implemented. *Journal of Symbolic Computation*, 47(9):1011–1045, 2012.

[8] Daniel Le Berre and Laurent Simon. The essentials of the sat 2003 competition. In *In Sixth International Conference on Theory and Applications of Satisfiability Testing, volume 2919 of LNCS*, pages 452–467. Springer-Verlag, 2003.

[9] Marc Bezem. Processes, terms and cycles. chapter On the undecidability of coherent logic, pages 6–13. Springer-Verlag, Berlin, Heidelberg, 2005.

[10] Marc Bezem and Thierry Coquand. Automating coherent logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3835 of *Lecture Notes in Computer Science*, pages 246–260. Springer Berlin Heidelberg, 2005.

[11] Marc Bezem and Dimitri Hendriks. On the mechanization of the proof of hessenberg's theorem in coherent logic. *J. Autom. Reason.*, 40(1):61–85, 2008.

[12] A. Biere. Lingeling and Friends at the SAT Competition 2011. Technical report, Institute for Formal Models and Verification, Johannes Kepler University, 2011.

[13] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2009.

[14] Vincent D. Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Review*, 46(4):647–666, 2004.

[15] Franc Brglez, Xiao Yu Li, and Matthias F. Stallmann. On sat instance classes and a method for reliable performance experiments with sat solvers. *Annals of Mathematics and Artificial Intelligence*, 43(1-4):1–34, January 2005.

[16] Franc Brglez and Jason A. Osborne. Performance testing of combinatorial solvers with isomorph class instances. In *Experimental computer science on Experimental computer science*, ecs'07, pages 14–14, Berkeley, CA, USA, 2007. USENIX Association.

[17] B.M. Brown and T.P. Hettmansperger. Kruskalwallis, multiple comparisons and efron dice. *Australian and New Zealand Journal of Statistics*, 44(4):427–438, 2002.

[18] Paul R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA, 1995.

[19] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[20] F. N. David and C. L. Mallows. The Variance of Spearman's Rho in Normal Samples. *Biometrika*, 48, 1961.

[21] S. T. David, M. G. Kendall, and A. Stuart. Some Questions of Distribution in the Theory of Rank Correlation. *Biometrika*, 38, 1951.

[22] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, July 1962.

[23] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960.

[24] Hans de Nivelle and Jia Meng. Geometric resolution: a proof procedure based on finite model search. IJCAR'06, pages 303–317. Springer-Verlag, 2006.

[25] The Coq development team. *The Coq proof assistant reference manual, Version 8.4*. TypiCal Project, 2012.

[26] D. Devlin and B. O'Sullivan. Satisfiability as a Classification Problem. In *19th Irish Conf on Artificial Intelligence and Cognitive Science*, 2008.

[27] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[28] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *Theory and Applications of Satisfiability Testing*, 2004.

[29] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[30] B. Efron and C. Stein. The jackknife estimator of variance. *Annals of Statistics*, 9:586–596, 1981.

[31] Oren Etzioni and Ruth Etzioni. Statistical methods for analyzing speedup learning experiments. *Mach. Learn.*, 14(3):333–347, March 1994.

[32] John Fisher and Marc Bezem. Skolem machines. *Fundam. Inf.*, 91(1):79–103, 2009.

[33] Charles L. Forgy. Expert systems. chapter Rete: a fast algorithm for the many pattern/many object pattern match problem, pages 324–341. IEEE Computer Society Press, 1990.

[34] E. Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1), 1965.

[35] C. Gomes and B. Selman. Algorithm portfolios. *Artif. Intell.*, 126(1-2):43–62, 2001.

[36] Robert J. Grissom and John J. Kim. *Effect Sizes for Research: A Broad Practical Approach*. Lawrence Erlbaum, 1 edition, 2005.

[37] John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[38] Maureen Heymans and Ambuj K. Singh. Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, 19(suppl 1):i138–146, July 2003.

[39] Wassily Hoeffding. A Class of Statistics with Asymptotically Normal Distribution. *The Annals of Mathematical Statistics*, 19:293–325, 1948.

[40] Bjarne Holen, Dag Hovland, and Martin Giese. Efficient rule-matching for automated coherent logic. In *Norsk Informatikkonferanse*, 2012.

[41] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and D. M. Chickering. A bayesian approach to tackling hard computational problems. In *UAI*, pages 235–244, 2001.

[42] H. Hotelling. New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society*, 15(2), 1953.

[43] F. Hutter, H. Hoos, K. Leyton-Brown, and Stützle. Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[44] H. Jain, C. Bartzis, and E. Clarke. Satisfiability Checking of Non-Clausal Formulas Using General Matings. In *Proceedings of Ninth International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2006)*, pages 75–89, 2006.

[45] H. Jain and E. Clarke. Efficient SAT Solving for Non-Clausal Formulas using DPLL, Graphs, and Watched Cuts. In *46th Design Automation Conference (DAC)*, 2009.

[46] Predrag Janičić. *Matematička logika u računarstvu*. Matematički fakultet Univerziteta u Beogradu, 2006.

[47] Predrag Janičić and Stevan Kordić. Euclid - the geometry theorem prover. *FILOMAT*, 3(9), 1995.

[48] Toni Jussila, Armin Biere, Carsten Sinz, Daniel Krning, and Christoph M. Wintersteiger. A first step towards a unified proof checker for qbf. In *Proc. of SAT. To Appear*, 2007.

[49] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. Isac – instance-specific algorithm configuration. In *Proceeding of the 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press, 2010.

[50] M. Kendall. Further contributions to the theory of paired comparisons. *Biometrics*, 11(1), 1955.

[51] Vlado Keselj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING'03*, pages 255–264, Dalhousie University, Halifax, Nova Scotia, Canada, August 2003.

[52] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[53] Sava Krstic and Amit Goel. Architecting Solvers for SAT Modulo Theories: Nelson-Oppen with DPLL. In *Frontiers of Combining Systems*, volume 4720/2007 of *Lecture Notes in Computer Science*, pages 1–27. Springer Berlin / Heidelberg, September 2007.

[54] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[55] Michail G. Lagoudakis and Michael L. Littman. Learning to select branching rules in the dpll procedure for satisfiability. In *In LICS/SAT*, pages 344–359, 2001.

[56] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[57] E. Lehmann. Consistency and unbiasedness of certain nonparametric tests. *The Annals of Mathematical Statistics*, 22(2), 1951.

[58] E. A. Leicht, Petter Holme, and M. E. J. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120+, Feb 2006.

[59] Hans leo Teulings, Lambert R. B. Schomaker, Jan Gerritsen, Hans Drexler, and Marc Albers. An on-line handwriting-recognition system based on unreliable modules. In *Computer Processing of Handwriting*, pages 167–185. World Scientific, 1990.

[60] Lionel Lobjois and Michel Lemaitre. Branch and Bound Algorithm Selection by Performance Prediction. In *In AAAI*, pages 353–358. AAAI, 1998.

[61] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of las vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

[62] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Non-model-based algorithm portfolios for SAT. In *Theory and Applications of Satisfiability Testing (SAT)*, 2011.

[63] N. Mantel. Ranking procedures for arbitrarily restricted observations. *Biometrics*, 23(1), 1967.

[64] Filip Marić. *Formalizacija, implementacija i primene SAT rešavača*. PhD thesis, Matematički fakultet Univerziteta u Beogradu, 2009.

[65] Filip Marić. Formalization and implementation of modern sat solvers. *J. Autom. Reason.*, 43(1):81–119, June 2009.

[66] Filip Marić. Formal verification of a modern sat solver by shallow embedding into isabelle/hol. *Theor. Comput. Sci.*, 411(50):4333–4356, November 2010.

[67] Filip Marić and Predrag Janičić. Formal correctness proof for dpll procedure. *Informatica*, 21(1):57–78, January 2010.

[68] Filip Marić and Predrag Janičić. Formalization of abstract state transition systems for sat. *Logical Methods in Computer Science*, 7(3), 2011.

[69] P. Matos, J. Planes, F. Letombe, and J. Marques-Silva. A MAX-SAT Algorithm Portfolio. In *Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 911–912, 2008.

[70] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, pages 117–128, Washington, DC, 2002. IEEE Computer Society.

[71] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 530–535. ACM, 2001.

[72] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(t). *J. ACM*, 53(6):937–977, November 2006.

[73] M. Nikolić, F. Marić, and P. Janičić. Instance-based selection of policies for sat solvers. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 326–340. Springer, 2009.

[74] Mladen Nikolić. Statistical methodology for comparison of sat solvers. In *SAT*, pages 209–222, 2010.

[75] Mladen Nikolić. Measuring similarity of graph nodes by neighbor matching. *Intelligent Data Analysis*, 16, 2013.

[76] Mladen Nikolić and Predrag Janičić. Cdcl-based abstract state transition system for coherent logic. In *AISC/MKM/Calculemus*, pages 264–279, 2012.

[77] Mladen Nikolić, Filip Marić, and Predrag Janičić. Simple algorithm portfolio for sat. *Artificial Intelligence Review*, pages 1–9, 2012.

[78] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[79] Eugene Nudelman, Kevin L. Brown, Holger H. Hoos, Alex Devkar, and Yoav Shoham. Understanding Random SAT: Beyond the Clauses-to-Variables Ratio. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 438–452. Springer, 2004.

[80] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Proceedings of the 10th international conference on Theory and applications of satisfiability testing*, SAT'07, pages 294–299. Springer-Verlag, 2007.

[81] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning sat solvers with restarts. In *Proceedings of the 15th international conference on Principles and practice of constraint programming*, CP'09, pages 654–668, Berlin, Heidelberg, 2009. Springer-Verlag.

[82] Ruzica Piskac, Leonardo Moura, and Nikolaj Bjørner. Deciding effectively propositional logic using dpll and substitution sets. *J. Autom. Reason.*, 44(4):401–424, 2010.

[83] Andrew Polonsky. *Proofs, Types, and Lambda Calculus*. PhD thesis, University of Bergen, 2012.

[84] L. Pulina and A. Tacchella. A Multi-Engine Solver for Quantified Boolean Formulas. In *Principles and practice of constraint programming*, pages 574–589, 2007.

[85] Luca Pulina. Empirical evaluation of scoring methods. In *Proceedings of the 2006 conference on STAIRS 2006: Proceedings of the Third Starting AI Researchers' Symposium*, pages 108–119, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[86] Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI Commun.*, 15(2,3):91–110, 2002.

[87] Robert Rosenthal. *Meta-Analytic Procedures for Social Research*. Sage, 1991.

[88] Horst Samulowitz and Roland Memisevic. Learning to Solve QBF. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 255–260, Vancouver, British Columbia, Canada, 2007. AAAI Press.

[89] João P. Marques Silva and Karem A. Sakallah. Grasp — a new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, ICCAD '96, pages 220–227, Washington, DC, USA, 1996. IEEE Computer Society.

[90] B. Silverthorn and R. Miikkulainen. Latent Class Models for Algorithm Portfolio Methods. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[91] Carsten Sinz. Visualizing sat instances and runs of the dpll algorithm. *J. Autom. Reasoning*, 39(2):219–243, 2007.

[92] Sana Stojanović, Vesna Pavlović, and Predrag Janičić. A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In Pascal Schreck, Julien Narboux, and Jrgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 201–220. Springer Berlin Heidelberg, 2011.

[93] C. Thiffault, F. Bacchus, and T. Walsh. Solving Non-clausal Formulas with DPLL Search. In *Principles and Practice of Constraint Programming — CP 2004*, pages 663–678, 2004.

[94] Andrija Tomovic, Predrag Janicic, and Vlado Keselj. n-Gram-Based Classification and Unsupervised Hierarchical Clustering of Genome Sequences. *Computer Methods and Programs in Biomedicine*, 81(2):137–153, 2006.

[95] G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic*, 2:115–125, 1968.

[96] Milena Vujošević-Janičić, Mladen Nikolić, Dušan Tošić, and Viktor Kuncak. Software verification and graph similarity for automated evaluation of students assignments. *Information and Software Technology*, 2012.

[97] Markus Wenzel. Isar - a generic interpretative approach to readable formal proof documents. In *Proceedings of the 12th International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '99, pages 167–184. Springer-Verlag, 1999.

[98] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. SATzilla2009: an Automatic Algorithm Portfolio for SAT. In *SAT Competition 2009*, 2009.

[99] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of Artificial Inteligence Research*, 32(1):565–606, 2008.

[100] Yehua Xu, David Stern, and Horst Samulowitz. Learning Adaptation to Solve Constraint Satisfaction Problems. In *LION 3*, 2009.

[101] L. Zager and G. Verghese. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21(1):86–94, January 2008.

Biografija autora

Mladen Nikolić je rođen 3. jula 1981. u Beogradu, gde je završio osnovnu školu i gimnaziju. Studije na smeru Računarstvo i informatika na Matematičkom fakultetu u Beogradu upisao je 2000. godine. Diplomirao je 2005. godine i upisao magistarske studije na istom smeru. Iste godine bio je izabran u zvanje asistenta saradnika na Matematičkom fakultetu. Magistarski rad pod nazivom "Metodologija izbora pogodnih vrednosti parametara SAT rešavača" odbranio je 2008. godine. Izabran je u zvanje asistenta na Matematičkom fakultetu 2009. godine.

Bavi se istraživanjima u oblasti automatskog rezonovanja, pre svega vezano za unapređivanje i proširivanje SAT rešavača, u oblasti istraživanja podataka, pre svega vezano za analizu grafova i socijalnih mreža, kao i za analizu prostornih podataka. Objavio je veći broj radova u vodećim časopisima i na vodećim međunarodnim konferencijama iz oblasti automatskog rezonovanja i istraživanja podataka. Ucestvovao je u radu vise medjunarodnih radionica i letnjih skola posvećenih temama iz automatskog rezonovanja i geostatistike. U istrazivačkim posetama bio je dva puta u Švajcarskoj, na univerzitetima EPFL i UNIL (2010. i 2011. godine), i jednom u Sjedinjenim Američkim Državama, na univerzitetu Temple u Filadelji (četiri meseca tokom 2012. i 2013. godine).

U toku rada na Matematičkom fakultetu držao je vežbe iz 6 predmeta. Sa prof. Predragom Janičićem je koautor skripte za predmet Veštačka inteligencija. Glasovima studenata, izabran je za najboljeg asistenta na Matematičkom fakultetu za 2006/2007. godinu.

Прилог 1.

Изјава о ауторству

Потписани-а _____

број индекса _____

Изјављујем

да је докторска дисертација под насловом

_____

_____

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, _____

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора _____

Број индекса _____

Студијски програм _____

Наслов рада _____

Ментор _____

Потписани/а _____

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

_____

_____

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, _____

_____

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.