UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Viktor Novaković

DIZAJN I IMPLEMENTACIJA DOKAZIVAČA TEOREMA KOREKTNOSTI GEOMETRIJSKIH KONSTRUKCIJA

master rad

Beograd, 2024.

Mentor:

dr Filip Marić, redovan profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Vesna Marinković, docent
Univerzitet u Beogradu, Matematički fakultet

dr Predrag Janičić, redovan profesor
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Dizajn i implementacija dokazivača teorema korektnosti geometrijskih konstrukcija

Rezime: U ovom radu se predstavlja dokazivač prilagođen problemu dokazivanja geometrijskih konstrukcija. ArgoTriCS sistem je sposoban da generiše konstrukcije trouglova za mnoge probleme i koristi dostupne dokazivače kako bi svoje konstrukcije dokazao. Međutim, dobijeni dokazi nisu čitljivi, već samo na pitanje da li je konstrukcija tačna daju odgovor da/ne. Proučava se korišćenje dokazivača koji mogu da generišu čitljiv dokaz, ali je problem previše komplikovan i specifičan za proizvoljne dokazivače, na šta najviše utiče količina lema visokog nivoa koje su deo aksiomatskog sistema. Dokazivač predstavljen u ovom radu je dizajniran baš za ovaj problem. Sposoban je da radi sa proizvoljnim komplikovanim aksiomatskim sistemima i uspeva da efikasno dokazuje teoreme koje je u njima moguće dokazati.

Ključne reči: algoritmi, geometrija, logika, računarstvo

Sadržaj

Glava 1

Uvod

Geometrija je ključni deo matematičkog obrazovanja i donela je mnoge bitne promene u matematici. Euklid je svojim radom popularizovao geometriju, i njegov aksiomatski sistem je značajno uticao na njen razvoj, čineći je vrlo rigoroznom naukom koja zahteva precizno rezonovanje. Geometrija je postala popularna i široko proučena tema i u oblasti automatskog dokazivanja teorema, upravo zbog svoje preciznosti.

Koncept geometrijskih konstrukcija postoji već hiljadama godina, gde je cilj problema konstrukcija da se, sa datim opisom figure, odredi konkretna procedura koju treba preduzeti za njenu izradu, sa dostupnim koracima konstrukcije. Problemi konstrukcije trouglova pomoću lenjira i šestara su jedna od najstarijih i najviše proučavanih oblasti problema konstrukcije. U tim problemima je potrebno da se konstruiše trougao sa datim ograničenjima, koristeći samo lenjir i šestar, gde se podrazumeva da se lenjir ne može koristiti za merenja.

Sistem za automatsko generisanje konstrukcija trouglova, ArgoTriCS [22], uspeva da za mnoge probleme, sa nekim raspoloživim geometrijskim znanjem, generiše validne konstrukcije. Dostupni automatski dokazivači teorema se koriste kako bi se konstrukcije dokazale, ali iako uspešno pokažu da li je konstrukcija ispravna, ne nude razumljiv dokaz. U radu [23] se razmatra korišćenje dokazivača u logici prvog reda, kao i u njenom fragmentu, koherentnoj logici, u nadi da se postignu čitljivi dokazi konstrukcija ArgoTriCS sistema. Primećuje se da je ovaj problem previše specifičan za ovakve dokazivače, i da zahteva leme visokog nivoa kao deo aksiomatskog sistema, što dalje muči dokazivače.

Cilj ovog rada je da se dizajnira i implementira prilagođeni dokazivač ovom problemu, koji može da podrži komplikovane aksiomatske sisteme i da efikasno

pronađe i prikaže čitljiv dokaz za odgovarajuću konstrukciju. Dokazivač radi u još manjem fragmentu logike prvog reda, prilagođenom ovom problemu, dokaz vrši pretragom u širinu i sposoban je da izvede sve informacije koje su moguće u datom aksiomatskom sistemu.

Organizacija rada. U poglavlju 2 se opisuje automatsko dokazivanje geometrijskih teorema i njegovi pristupi, u poglavlju 3 se definiše pojam koherentne logike i opisuje se dokazivanje u njoj. U poglavlju 4 se govori detaljnije o problemu geometrijskih konstrukcija i sistemu ArgoTriCS. U poglavlju 5 se opisuje postupak prilagođenog dokazivača, u poglavlju 6 je detaljnija implementacija algoritma, a u poglavlju 7 su prikazani rezultati nakon testiranja nad datim skupom problema, i prikazani su neki primeri. Konačno, u poglavlju 8 se dovode zaključci i pominje se mogući dalji rad.

Glava 2

Automatsko dokazivanje geometrijskih teorema

Automatsko dokazivanje geometrijskih teorema (ADGT) kao tema postoji već decenijama. Prvi koraci napravljeni su još 1959. godine sa Gelernterovim dokazivačem [12]. Prvi automatski dokazivači geometrijksih teorema, poput Gelernterovog, koristili su opšte pristupe rezonovanja razvijene u oblasti veštačke inteligencije i time su uspeli da automatizuju proces dokazivanja u geometriji. Ove dokazivače zovemo *sintetičkim*. Simuliranjem načina dokazivanja čoveka, rani sintetički dokazivači su generisali čitljive dokaze, ali zbog kombinatorne eksplozije nisu mogli da dokazuju komplikovane teoreme. Da bi se izbegla kombinatorna eksplozija, razvijeni su mnogi heuristički pristupi, ali su dobijeni metodi bili uskog opsega i neefikasni [1]. Kasnije, iz potrebe za moćnijim dokazivačima razvijaju se novi metodi koji koriste *algebarski* pristup pri dokazivanju. Ovi dokazivači prevode geometrijske osobine u jednačine sa koordinatama odgovarajućih tačaka i zatim rade nad tim jednačinama. Smatraju se velikim uspehom u svetu ADGT i uspeli su da dokažu mnoge kompleksne teoreme, ali jedna mana im je to da njihovi dokazi nisu čitljivi. U potrazi za dokazivačima koji imaju čitljiv dokaz, ali koji su takođe dovoljno moćni, verovalo se da je potrebno da se ne koriste koordinate pri dokazivanju. Razvijaju se tzv. *polualgebarske* metode ili metode nezavisne od koordinata (coordinate-free), koje umesto koordinata tačaka koriste određene geometrijske veličine. Takođe, postoje noviji sintetički pristupi koji imaju dobre rezultate u određenim klasama geometrijskih problema [1].

2.1 Algebarski metodi

Algebarski metodi dokazivanja su najuspešniji u oblasti ADGT. Ovi metodi formiraju polinomske jednačine nad koordinatama odgovarajućih tačaka, i zatim pokušavaju da rešavaju dobijene sisteme jednačina. Među najuticajnijim su Vuova metoda i metoda Grebnerovih baza.

Vuova metoda

Kratak pregled Vuove metode (zasnovan na [11]):

- Geometrijska teorema se prevodi u algebarski sistem polinomskih jednačina, time se dobija skup hipoteza $f_1, ..., f_r$ i zaključak g. Polinomi $f_1, ..., f_r, g$ su oblika $p(u_1, ..., u_d, x_1, ..., x_r)$, gde su parametri u_i (za $1 \leq i \leq d$) nezavisne promenljive koje označavaju opšti karakter geometrijske konstrukcije (npr. proizvoljna tačka, proizvoljna prava), a parametri x_j (za $1 \leq j \leq r$) su zavisne promenljive i označavaju geometrijske uslove (npr. normala iz tačke na pravu, presek dveju pravih).

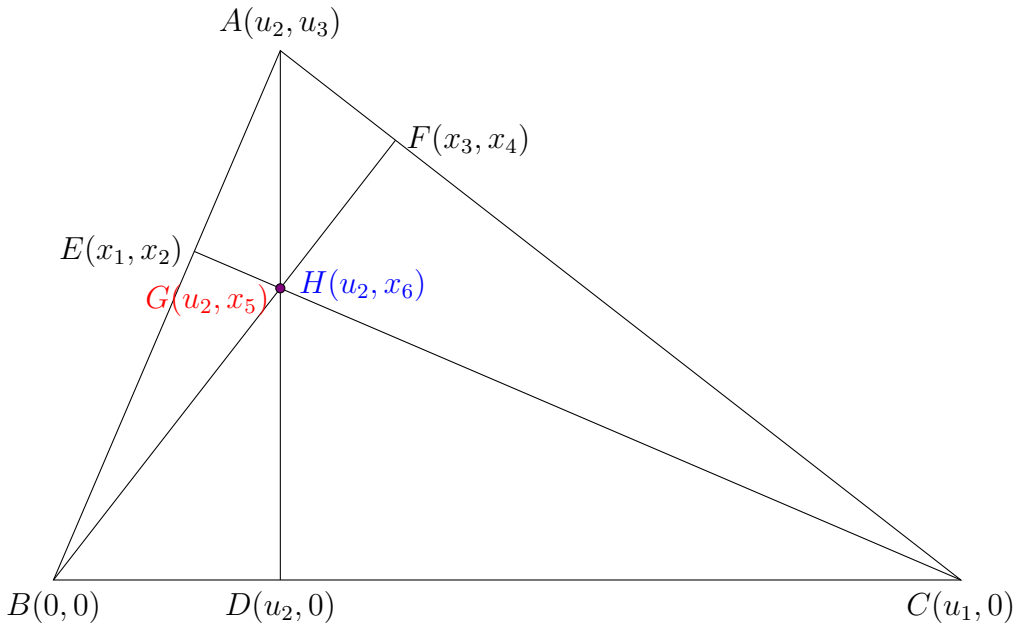
- Sistem jednačina se svodi u trougaonu formu koristeći *pseudo-deljenje*. Pod trougaonom formom se misli da se jednačine mogu zapisati na sledeći način:

$$\begin{aligned} f_1 &= f_1(u_1, ..., u_d, x_1) \\ f_2 &= f_2(u_1, ..., u_d, x_1, x_2) \\ &\vdots \\ f_r &= f_r(u_1, ..., u_d, x_1, ..., x_r) \end{aligned}$$

  Ovaj novi sistem sadrži nerazložive komponente originalnog sistema.

- Izvršavaju se uzastopna pseudo-deljenja nad hipotezama u trougaonoj formi i jednačini zaključka, i dobija se ostatak. Ako je ostatak nula, može se reći da zaključak g sledi iz hipoteza $f_1, ..., f_r$.

- Razmatraju se uslovi nedegenerisanosti koji nastaju tokom triangulacije hipoteza. Zaključuje se da g sledi iz hipoteza $f_1, ..., f_r$, ako važe uslovi nedegenerisanosti. Ovi uslovi su oblika $p \neq 0$, a p je polinom koji nastaje prirodno tokom procesa triangulacije.



Slika 2.1: Dijagram ortocentra

Primer 2.1 *U sledećem primeru (iz [11]) razmatramo algebarsku formulaciju standardne geometrijske teoreme ortocentra, koja glasi da se sve visine trougla* $\triangle ABC$ *seku u jednoj tački* H*, koju nazivamo* ortocentar *(slika 2.1).*

Osnovna ideja je da se data figura postavi u koordinantni sistem, a zatim da se hipoteze teoreme tumače kao iskazi u koordinatnoj geometriji. Počinjemo sa dodeljivanjem koordinata i, bez gubljenja opštosti, tački B *dodeljujemo koordinate* $(0, 0)$*, onda tački* C *možemo da dodelimo koordinate* $(u_1, 0)$*, a tački* A *koordinate* (u_2, u_3)*. Koordinate ostalih tačaka mogu da se dobiju od datih tačaka. Lako je primetiti da ako je duž* $\overline{AD}$ *visina, onda će koordinate tačke* D *biti* $(u_2, 0)$*.*

Neka su $E = (x_1, x_2)$ *i* $F = (x_3, x_4)$ *tačke takve da su* $\overline{BF}$ *i* $\overline{CE}$ *visine iz tačaka* B *i* C *redom. To znači da su tačke* B, E, A *i tačke* C, F, A *kolinearne. Takođe mora da važi* $\overline{CE} \perp \overline{AB}$*,* $\overline{BF} \perp \overline{AC}$*. Ovim pravilima odgovaraju redom naredne hipoteze:*

$$\begin{aligned} x_2u_2 - x_1u_3 &= 0 \\ x_4(u_2 - u_1) - u_3(x_3 - u_1) &= 0 \\ x_2u_3 + u_2(x_1 - u_1) &= 0 \\ x_4u_3 + x_3(u_2 - u_1) &= 0 \end{aligned}$$

i ove polinome obeležavamo h_1*,* h_2*,* h_3 *i* h_4*. Sada treba da se zaključi da se sve*

tri visine seku u jednoj tački. Konstruišemo naredne dve tačke: $G = (u_2, x_5)$ *i* $H = (u_2, x_6)$*. G definišemo kao presek duži* $\overline{AD}$ *i* $\overline{CE}$*, a H kao presek duži* $\overline{AD}$ *i* $\overline{BF}$*. Dakle potrebne su nam dodatne hipoteze da su tačke* G, E, C *i tačke* H, B, F *kolinearne, od čega dobijamo naredne jednačine:*

$$(x_2 - x_5)(x_1 - u_1) - x_2(x_1 - u_2) = 0$$
$$x_6 x_3 - x_4 u_2 = 0$$

koje obeležavamo h_5 *i* h_6*. Konačno, ostaje nam zaključak da su tačke G i H jednake, od toga dobijamo jednačinu:*

$$x_5 - x_6 = 0.$$

Ovaj polinom nazivamo g.

Grebnerove baze

Buhberger (Buchberger) je 1965. godine u svojoj doktorskoj disertaciji [3] uveo pojam Grebnerovih baza i takođe definisao algoritam za njihovo formiranje za dati skup polinoma. Dalje definišemo Grebnerove baze (zasnovano na [20]).

Neka je K polje i $K[y_1, ..., y_m]$ (dalje $K[y]$) prsten polinoma nad K, uzimajući u obzir konačni skup polinoma $F \subset K[y]$, *ideal* generisan od F (koji možemo da zapišemo kao $\langle f_1, ..., f_n \rangle$) je:

$$I_{K[y]}(F) := \{\sum h_i f_i : f_i \in F, h_i \in K[y]\}$$

Nadalje, uzmemo neko „prihvatljivo" uređenje monoma $<$, npr. leksičko uređenje, gde je monom $y_1^{p_1} \cdots y_m^{p_m}$ pre monoma $y_1^{q_1} \cdots y_m^{q_m}$ po uređenju ako postoji $k \leq m$ tako da je $p_i = q_i$ za $i < k$, i $p_k < q_k$. Svaki polinom iz $K[y]$ se može zapisati: $p = c_1 m_1 + c_2 m_2 + ... + c_k m_k$, gde su c_i konstante a m_i monomi uređeni opadajuće, tj. $m_1 > m_2 > ... > m_k$. *Vodeći monom* polinoma $lm(p)$ je monom koji je najveći u poretku, u ovom slučaju m_1, *vodeći koeficijent* $lc(p)$ je koeficijent koji ide uz vodeći monom (c_1), a *vodeći term* $lt(p)$ je term koji sadrži vodeći monom ($c_1 m_1$).

Polinom g može da se *redukuje* polinomom f akko je $lm(f)$ faktor nekog monoma u g. Tada postoji monom u i $b \in K, b \neq 0$, tako da je $u \cdot b \cdot lm(f)$ jednak odgovarajućem termu u g, polinom dobijen *redukovanjem* je $h = g - u \cdot b \cdot f$. Polinom g može da se *redukuje po modulu* F akko postoji $f \in F$ tako da g može da se redukuje sa f. *Normalna forma* polinoma g po modulu F (u oznaci g_F) je

polinom koji je poslednji u nizu redukcija polinoma g po modulu F, tj. polinom koji ne može dalje da se redukuje po modulu F. Pošto proces redukcije ne mora da bude uvek isti, normalne forme polinoma g po modulu F dobijene na različiti način mogu da se razlikuju.

Baza G ideala $I \subset K[y]$ je *Grebnerova baza* ideala I ako se svakom redukcijom polinoma $f \in K[y]$ po modulu G dobija ista normalna forma. Ako je G Grebnerova baza ideala I, polinom g pripada idealu I akko je $g_G = 0$.

Buhbergerov algoritam [3] se koristi za konstrukciju Grebnerovih baza. Sposoban je da za dati konačni skup polinoma F generiše konačan skup polinoma G, tako da je $I(F) = I(G)$ i G je Grebnerova baza.

Dokazivanje teorema korišćenjem Grebnerovih baza: slično kao kod Vuove metode, počinje se sa skupom hipoteza $h_1, ..., h_k$ i zaključkom g. Uslovi hipoteza i zaključka su oblika $h_i = 0$ i $g = 0$, dok su uslovi nedegenerisanosti oblika $s_i \neq 0$. Da bismo dokazali da važi teorema:

$$(\forall x_1, ..., x_n)((h_1 = 0 \wedge ... \wedge h_k = 0 \wedge s_1 \neq 0 \wedge ... \wedge s_l \neq 0) \Rightarrow g = 0)$$

dovoljno je da pokažemo da je Grebnerova baza ideala generisanog od polinoma $\{h_1, ..., h_k, s_1 \cdot z_1 - 1, ..., s_l \cdot z_l - 1, g \cdot z - 1\}$ (gde su $z_1, ..., z_l, z$ nove jedinstvene promenljive koje se ne pojavljuju u polinomima sa kojima se množe, i svaka od njih je različita od $x_1, ..., x_n$) jedinična [19].

Primer 2.2 *Uzmimo primer teoreme ortocentra iz primera 2.1. Imamo hipoteze:*

$$h_1 : x_2u_2 - x_1u_3 = 0$$
$$h_2 : x_4(u_2 - u_1) - u_3(x_3 - u_1) = 0$$
$$h_3 : x_2u_3 + u_2(x_1 - u_1) = 0$$
$$h_4 : x_4u_3 + x_3(u_2 - u_1) = 0$$
$$h_5 : (x_2 - x_5)(x_1 - u_1) - x_2(x_1 - u_2) = 0$$
$$h_6 : x_6x_3 - x_4u_2 = 0$$

i zaključak:

$$g : x_5 - x_6 = 0.$$

Da bismo dokazali da zaključak g sledi iz hipoteza $h_1, ..., h_6$ razmatramo ideal:

$$\langle h_1, h_2, ..., h_6, gz - 1\rangle \subset \mathbb{R}(u_1, u_2, u_3)[x_1, x_2, ..., x_6, z]$$

Buhbergerovim algoritmom se dobija da je baza ovog ideala $\{1\}$, iz čega sledi da je teorema dokazana.

2.2 Polualgebarski metodi

Algebarski metodi, iako su vrlo moćni, uglavnom za neku tvrdnju mogu samo da kažu da li je tačna ili nije. Kod polualgebarskih metoda, da bi se dobio čitljiv dokaz, umesto korišćenja koordinata odgovarajućih tačaka, koriste se određene geometrijske veličine, npr. kod metoda površina [4] se koriste: *odnos paralelnih segmenata*, *orijentisana površina* i *Pitagorina razlika*. Iako polualgebarski metodi pružaju čitljiv dokaz, taj dokaz ne sadrži korake kao kod uobičajenog rezonovanja.

Metod površina

Kratak pregled metoda površina (zasnovan na [18]):

Za formulaciju i dokazivanje tvrdnji, metod površina koristi skup *geometrijskih veličina*:

- *Odnos paralelnih usmerenih segmenata*, predstavljen sa $\overline{AB}/\overline{CD}$. Ako su tačke $A, B, C,$ i D kolinearne, $\overline{AB}/\overline{CD}$ je odnos dužina usmerenih segmenata AB i CD. Ako nisu kolinearne i važi da su AB i CD paralelni, postoji paralelogram $ABPQ$ takav da su tačke P, Q, C i D kolinearne i važi $\overline{AB}/\overline{CD} = \overline{QP}/\overline{CD}$

- *Orijentisana površina* trougla ABC, predstavljena sa $\mathcal{S}_{ABC}$ je površina trougla ABC, koja je negirana ako ABC ima negativnu orijentaciju.

- *Pitagorina razlika*, predstavljena sa $\mathcal{P}_{ABC}$ za tačke A, B, C, je definisana kao $\mathcal{P}_{ABC} = \overline{AB}^2 + \overline{CB}^2 - \overline{AC}^2$

Ideja metoda površina je da se hipoteze teorema predstave korišćenjem nekog skupa početnih („slobodnih") tačaka i skupa konstruktivnih tvrdnji, od kojih svaka uvodi novu tačku, a zaključak se predstavlja korišćenjem jednakosti polinoma geometrijskih veličina (bez upotrebe Dekartovih koordinata). Osnovne aksiome, koje formalno opisuju osobine geometrijskih veličina, čine deduktivnu bazu ovog metoda. Dokaz se razvija uklanjanjem uvedenih tačaka obrnutim redosledom, koristeći odgovarajuće aksiome. Nakon uklanjanja svih uvedenih tačaka, jednakost zaključka se svodi na jednakost između dva racionalna izraza koji sadrže samo slobodne tačke. Ova jednakost može da se dalje pojednostavi tako da uključuje samo nezavisne promenljive. Ako su izrazi sa obe strane jednakosti jednaki, data *pretpostavka* je *teorema*.

Primer 2.3 *Probajmo da dokažemo teoremu ortocentra iz primera 2.1. Za to ćemo koristiti sledeće osnovne aksiome:*

(1) *Pitagorina razlika tri tačke je jednaka Pitagorinoj razlici iste tri tačke u obrnutom redosledu,* $\mathcal{P}_{ABC} = \mathcal{P}_{CBA}$.

(2) *Prava* AB $(A \neq B)$ *je normalna na pravu* CD $(C \neq D)$ *akko* $\mathcal{P}_{ACD} = \mathcal{P}_{BCD}$.

Dokaz. *Definišemo* H *kao presek duži* $\overline{BF}$ *i* $\overline{AD}$*, onda je dovoljno da pokažemo da važi* $AB \perp CH$*, tj.* $\mathcal{P}_{ACH} = \mathcal{P}_{BCH}$ *(2). Iz aksiome (1) imamo* $\mathcal{P}_{ACH} = \mathcal{P}_{HCA}$*;* $\mathcal{P}_{BCH} = \mathcal{P}_{HCB}$*, i iz aksiome (2) imamo* $\mathcal{P}_{HCA} = \mathcal{P}_{BCA}$*;* $\mathcal{P}_{HCB} = \mathcal{P}_{ACB}$*. Ponovo, iz aksiome (1) imamo da važi* $\mathcal{P}_{BCA} = \mathcal{P}_{ACB}$ *iz čega sledi* $\mathcal{P}_{ACH} = \mathcal{P}_{BCH}$.

2.3 Sintetički metodi

Sintetički metodi su generalno slabiji od algebarskih i polualgebarskih, ali imaju i svoje prednosti [5]:

- Dokazi dobijeni sintetičkim metodama su uglavnom najlakši za razumevanje.
- Korišćenjem samo *predikata* se omogućava dostizanje *fiksne tačke*.
- Iako algebarski metodi mogu da dokažu mnogo više teorema, postoje teoreme koje mogu biti elegantno dokazane sintetičkim metodima, a ne mogu biti dokazani algebarskim, jer bi zahtevali previše računarske memorije.

Ovde razmatramo dva značajna pristupa: *Gelernterovu geometrijsku mašinu* [12] i *deduktivnu bazu podataka* [6]. Vredno je napomenuti da se dokazivač koji je predmet ovog rada svrstava u klasu sintetičkih metoda, tačnije u klasu metoda zasnovanih na *koherentnoj logici*, o kojima će biti reči kasnije.

Gelernterova geometrijska mašina

Gelernterova geometrijska mašina koristi *pristup ulančavanja unazad* (backward chaining), tj. rezonuje od zaključka ka hipotezama. Neka su $h_1, ..., h_r$ hipoteze i g zaključak. Da bi se dokazao cilj g, pretražuje se skup *aksioma* dok se ne nađe pravilo oblika:

$$(g_1 \wedge ... \wedge g_t) \Rightarrow g$$

Da bi cilj g bio tačan, dovoljno je dokazati da su *podciljevi* $g_1, ..., g_t$ tačni. Ovaj postupak se ponavlja za svaki od podciljeva rekurzivno sve dok podcilj nije jedan od hipoteza. Ovime se generiše stablo dokaza u kojem, da bi se dokazao bilo koji cilj, potrebno je da se dokaže svaki od njegovih podciljeva. Takođe, neki cilj može da bude generisan pomoću više različitih pravila, pri čemu je dovoljno da se dokaže korišćenjem samo jednog od njih. Drugim rečima, da bi se dokazao cilj u stablu, dovoljno je dokazati jednu njegovu granu.

Primer 2.4 *Za primer dokaza razmotrimo teoremu koja glasi da presek dijagonala paralelograma deli dijagonale na pola (slika 2.2). Prediktom* $coll(A, B, C)$ *se obeležava da su tačke A, B i C kolinearne. Uzimamo tačke A, B i C kao proizvoljne, dok ostale tačke mogu da se dobiju pomoću njih. Da bi se dokazalo da je tačka E na polovini segmenta* $\overline{AD}$*, dovoljno je dokazati* $AE = ED$*. Hipoteze teoreme bi u ovom slučaju bile skup uslova da je ABCD paralelogram, i da je E presek dijagonala. Dakle, teorema glasi:*

$$AB \parallel CD \wedge AC \parallel BD \wedge coll(E, A, D) \wedge coll(E, B, C) \Rightarrow AE = EC$$

Pravila koja možemo da koristimo da bismo dokazali ovu teoremu glase:

$R_1 : AB \parallel CD \Rightarrow \angle ADC = \angle DAB$

$R_2 : \triangle ABC \cong \triangle DEF \Rightarrow AB = DE$

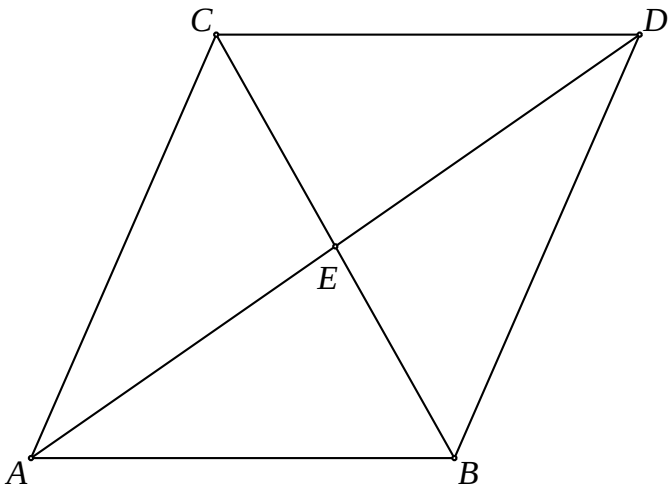
$R_3 : \angle ABC = \angle BAD \wedge coll(A, E, B) \wedge coll(C, E, D) \Rightarrow \angle EBC = \angle EAD$

$R_4 : AB = CD \wedge \angle EAB = \angle FCD \wedge \angle EBA = \angle FDC \Rightarrow \triangle ABE \cong \triangle CDF$

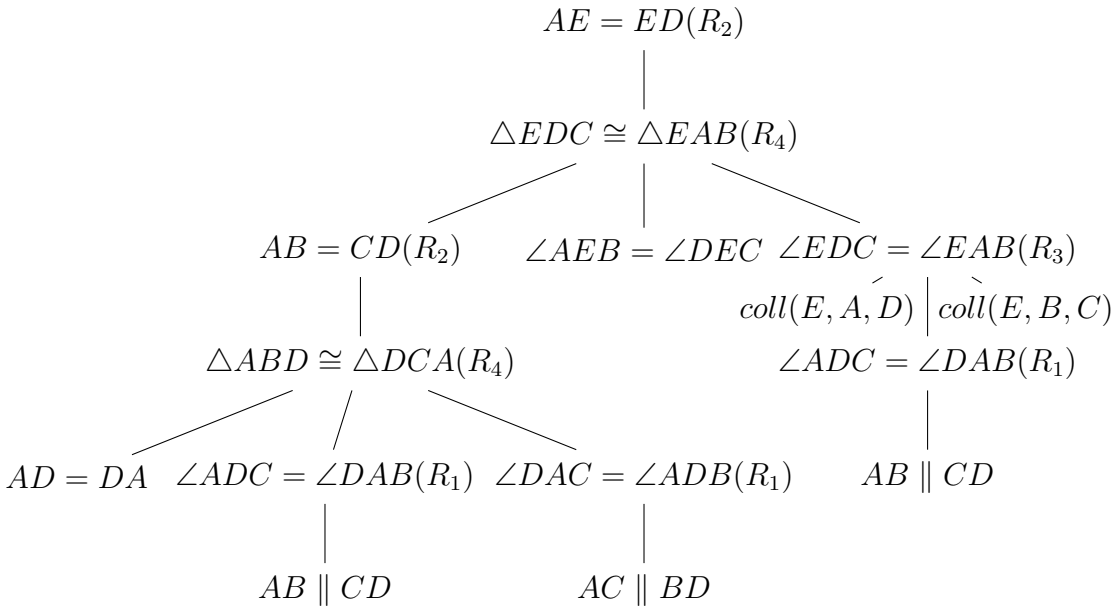
Pojednostavljeno stablo dokaza za ovu teoremu je na slici 2.3.

Deduktivna baza podataka

U radu [6] opisuje se program koji koristi deduktivnu bazu podataka za rešavanje problema dokazivanja geometrijskih teorema. Obrnuto od Gelernterove mašine, koristi se *pristup ulančavanja unapred* (forward chaining), tj. rezonuje se od hipoteza ka zaključku. Program je za datu geometrijsku konfiguraciju sposoban da dostigne *fiksnu tačku*, tj. može naći sve osobine konfiguracije koje se mogu dedukovati koristeći date aksiome. Koristi se *struktuirana deduktivna baza podataka* za koju se pokazuje da smanjuje veličinu baze. Takođe, umesto *strategije pretraživanja po pravilima* (*rule-based search strategy*), koristi se *strategija pretraživanja*



Slika 2.2: Dijagram paralelograma



Slika 2.3: Pojednostavljeno stablo dokaza primera 2.4

po informacijama (*data-based search strategy*). U strategiji pretraživanja po informacijama, čuva se lista „novih informacija" i za svaku od njih sistem pretražuje skup pravila da bi primenio pravilo koristeći ove informacije.

Glava 3

Dokazivanje u koherentnoj logici

Koherentna logika (CL), poznata i kao geometrijska logika, je izražajan fragment logike prvog reda [2]. Formula je *koherentna* ako je oblika [28]:

$$A_1(\boldsymbol{x}) \wedge ... \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y} B_1(\boldsymbol{x}, \boldsymbol{y}) \vee ... \vee B_m(\boldsymbol{x}, \boldsymbol{y})$$

gde se podrazumeva da je univerzalno kvantifikovana i gde: $0 \leq n, 0 \leq m$, $\boldsymbol{x}$ označava niz promenljivih $x_1, x_2, ..., x_k$ $(0 \leq k)$, A_i (za $1 \leq i \leq n$) označava atomičnu formulu, $\boldsymbol{y}$ označava niz promenljivih $y_1, y_2, ..., y_l$ $(0 \leq l)$, i B_j (za $1 \leq j \leq m$) označava konjukciju atomičnih formula. Koherentne formule nemaju funkcionalne simbole arnosti veće od nula. Funkcionalne simbole arnosti 0 zovemo *konstante*. *Svedoci* su nove konstante koje se ne pojavljuju ni u aksiomama ni u pretpostavci koju treba dokazati. *Term* se definiše kao konstanta ili promenljiva. *Atomična formula* je ili $\perp$ ili $p(t_1, ..., t_n)$, gde je p predikatski simbol arnosti n, a t_i $(1 \leq t \leq n)$ je term. Koherentne formule ne sadrže negaciju predikata.

Svaka teorija logike prvog reda može da se prevede u CL, sa mogućim dodatnim predikatskim simbolima [10, 17]. Ovaj proces prevođenja se naziva „koherentizacija" ili „geometrizacija" [10]. S obzirom na to da koherentne formule ne sadrže negaciju, pri prevođenju formule logike prvog reda u CL, potrebno je da se na neki način očuva negacija. Za svaki predikat R (koji se pojavi u negiranom obliku), uvodi se novi simbol $\overline{R}$ koji predstavlja $\neg R$, i uvode se aksiome $\forall \boldsymbol{x}(R(\boldsymbol{x}) \wedge \overline{R}(\boldsymbol{x}) \Rightarrow \perp)$ i $\forall \boldsymbol{x}(R(\boldsymbol{x}) \vee \overline{R}(\boldsymbol{x}))$.

TPTP

Većina dokazivača u koherentnoj logici na ulazu prihvataju probleme u TPTP[1] (Thousands of Problems for Theorem Provers) formatu, preciznije u TPTP FOF (First-Order Form) formatu. TPTP FOF format je standardizovani tekstualni zapis za izražavanje formula logike prvog reda. S obzirom na to da može da predstavi formule logike prvog reda, sposoban je da predstavi i koherentne formule. Primer zapisa aksioma u koherentnoj logici:

```
fof(perp_sym, axiom, ![L1,L2] : (perp(L1,L2) => perp(L2,L1))).
```

koji predstavlja aksiom: $\forall L_1, L_2(perp(L_1, L_2) \Rightarrow perp(L_2, L_1))$

Dokazivač koji je predmet ovog rada takođe prihvata probleme zapisane u TPTP FOF formatu. Važno je napomenuti da je implementacija čitanja TPTP fajlova preuzeta iz Larus dokazivača [17].

3.1 Pravila izvođenja

U CL je dozvoljeno korišćenje narednih pravila izvođenja (iz [21]):

$$\frac{A_1(\vec{a}) \wedge ... \wedge A_n(\vec{a})}{A_1(\vec{a}), ..., A_n(\vec{a})} \wedge E \qquad \frac{A_1 \vee ... \vee A_n \qquad \begin{matrix}[A_1] \\ \vdots \\ B\end{matrix} \quad \ldots \quad \begin{matrix}[A_n] \\ \vdots \\ B\end{matrix}}{B} \vee E \qquad \frac{\bot}{A}\, efq$$

$$\frac{A_1(\vec{a}), ..., A_n(\vec{a}) \qquad A_1(\vec{x}) \wedge ... \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}(\mathcal{B}_1(\vec{x}, \vec{y}) \vee ... \vee \mathcal{B}_m(\vec{x}, \vec{y}))}{\mathcal{B}_1(\vec{a}, \vec{w}) \vee ... \vee \mathcal{B}_m(\vec{a}, \vec{w})}\, ax$$

gde:

- $\vec{a}$ označava vektor konstanti,
- $\vec{w}$ vektor svedoka
- c_i, $1 \leq i \leq n$ niz pravila izvođenja.

[1] https://www.tptp.org

Pravila $(\wedge E)$, $(\vee E)$ i (efq) su standardna pravila eliminacije konjukcije, eliminacije disjunkcije i pravilo *ex falso quodlibet*. Primenom pravila $(\wedge E)$ izvodi se $A_i(\vec{a})$ za svako i, $1 \leq i \leq n$. Pravilo (ax) se primenjuje samo ako već ne postoji vektor konstanti $\vec{w}$ tako da važi: $\mathcal{B}_1(\vec{a}, \vec{w}) \vee \mathcal{B}_2(\vec{a}, \vec{w}) \vee ... \vee \mathcal{B}_m(\vec{a}, \vec{w})$.

Formula $A_1(\vec{x}) \wedge ... \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}(\mathcal{B}_1(\vec{x}, \vec{y}) \vee ... \vee \mathcal{B}_m(\vec{x}, \vec{y}))$ je *teorema u koherentnoj logici* ako se iz premisa $A_1(\vec{a}) \wedge ... \wedge A_n(\vec{a})$ (gde $\vec{a}$ označava niz instanciranih konstanti) mogu izvesti svi konjunkti formule $B_j(\vec{a}, \vec{w})$ za neko j, $1 \leq j \leq m$ i za neki vektor konstanti $\vec{w}$ [21].

3.2 Algoritam dokazivanja

Upoznali smo se sa konceptima ulančavanja unazad (backward chaining) i ulančavanja unapred (forward chaining). Kod koherentnih dokazivača je najčešće korišćen pristup ulančavanja unapred. Ideja ulančavanja unapred je rezonovanje od hipoteza ka zaključku. Pravila se primenjuju na postojeće informacije kako bi se dobile nove, koje se koriste u daljim koracima. Ovaj proces se ponavlja dok se ne dođe do zaključka.

U [25] Nevins je koristio kombinaciju ulančavanja unapred i unazad, sa akcentom na ulančavanje unapred. Još jedan primer dokazivača koji koristi pristup ulančavanja unapred je deduktivna baza podataka [6], već razmotrena u 2.3.

Većina sintetičkih dokazivača u ADGT je ipak koristila pristup ulančavanja unazad (poput npr. Gelernterovog [12]). Postojao je strah da bi se ulančavanjem unapred generisalo previše nepotrebnih tvrdnji, posebno kada bi se radilo sa velikim bazama podataka. Ulančavanje unazad je makar garantovalo da je računarski napor povezan sa ciljem koji treba da se dokaže. Ipak, poređenje nije toliko jednostavno, umesto veličine baze podataka možda veći uticaj imaju njena organizacija i heuristike koje se koriste u pretrazi. Dodavanje novih informacija može da bude od velikog značaja za dalje dokazivanje. Prema Nevinsu *„nemogućnost da se na smislen način iskoristi ulančavanje unapred je znak da prostor problema možda nije dobro struktuiran"* [25].

3.3 Koherentni dokazivači

Od postojećih dokazivača u koherentnoj logici navodimo:

- Euclid - prvi dokazivač teorema za koherentnu logiku. Koristio je fiksan skup aksioma i uspešno je dokazivao osnovne teoreme, ali nije mogao da radi nad proizvoljnom koherentnom teorijom [15].
- ArgoCLP - generički dokazivač, sposoban da radi u proizovljnim aksiomatskim sistemima, primarno fokusiran na geometrijske slučajeve [29].
- Geo - prvi koherentni dokazivač sa ugrađenim mehanizmom učenja lema [8].
- Larus - dokazivač koji dokaz predstavlja kao niz koji sadrži brojeve ili istinitosne vrednosti, sa određenim ograničenjima, i za njegovu pretragu koristi rešavanje ograničenja [17].

U daljem tekstu razmatramo ArgoCLP.

ArgoCLP

ArgoCLP-ov postupak dokazivanja koristi ulančavanje unapred i *iterativno produbljivanje* (iterative deepening). Aksiome se primenjuju korišćenjem pravila zaključivanja (u skladu sa pravilom zaključivanja (ax), datim u poglavlju 3.1), i to na kaskadni način, kada se jedna aksioma primeni, pretraga za narednu primenjivu aksiomu kreće ispočetka. Ovako je implementirana pretraga u dubinu, dok iterativno produbljivanje treba da ograničava dubinu pretrage u trenutnoj iteraciji. Za to se koristi brojač s koji je na početku jednak broju konstanti koje se pojavljuju u premisama tvrdnje koja treba da se dokaže. Sve konstante su numerisane i važi da aksioma može da bude primenjena samo ako su sve njene (univerzalno kvantifikovane) promenljive zamenjene konstantama čiji je indeks manji od s. U slučaju da nema primenjivih aksioma koje ispunjavaju taj uslov, vrednost brojača s se inkermentira. U radu je dokazano da je ovaj postupak potpun i sposoban da dokaže određenu koherentnu formulu, ukoliko je to moguće iz datih aksioma.

U radu su razmatrana dodatna unapređenja koja poboljšavaju efikasnost bez gubljenja potpunosti. Ciljevi unapređenja su kontrolisanje prostora pretraživanja (tj. broja uvedenih svedoka) i delimično smanjenje kombinatorne eksplozije (koju uzrokuju izvedene nepotrebne tvrdnje). Neka od ovih unapređenja navodimo ovde:

- *Uređenje aksioma*: aksiomi su uredjeni po produktivnosti (gde neproduktivni aksiomi nemaju egzistencijalne kvantifikatore, produktivni imaju, a vrlo produktivni nemaju hipoteze), i grananju (gde se aksiomi granaju ako u zaključku imaju više od jedne konjukcije).

- *Rano odsecanje*: kada se proverava primenjivost aksioma, nije potrebno da se instanciraju sve promenljive i da se onda proverava da li su izvedene sve relevantne činjenice. Umesto toga, provera relevantnih činjenica se radi što pre, što omogućava da se aksiome odbiju ranije i time se smanjuje prostor pretrage. Na primer, kada se primenjuje naredni aksiom:
  $\forall x : line\ \forall y : line\ \forall X : point\ \forall Y : point$
  $incident(X, x) \wedge incident(Y, x) \wedge incident(X, y) \wedge incident(Y, y) \wedge X \neq Y$
  $\Rightarrow x = y$
  umesto da se upare x, y, X i Y sa svim mogućim konstantama, prvo se upare x i X i proverava se da li je činjenica instancirana od $incident(X, x)$ već izvedena, u slučaju da nije, nema potrebe za daljim razmatranjem jer znamo da sa ovim uparivanjem primena aksioma nije moguća.

- *Rastavljanje aksioma koji uvode više svedoka*: kod aksioma poput:

  $$\forall x : line(\exists X : point, \exists Y : point(incident(X, x) \wedge incident(Y, x) \wedge X \neq Y))$$

  Ovaj aksiom ne treba da se primeni na neku pravu a (koja instancira x) ako već postoje konstante $A : point$ i $B : point$ za koje već važi: $incident(A, a)$, $incident(B, a)$ i $A \neq B$. Ali u slučaju da postoji samo konstanta A, a ne postoji B, ne bi bilo efikasno da se primeni cela aksioma, jer bi se onda dobile dve nove konstante: C i D. Tada je bolje da se koristi nova varijanta aksiome koja se uvodi:

  $$\forall x : line, \forall X : point(incident(X, x) \Rightarrow \exists Y : point(incident(Y, x) \wedge X \neq Y))$$

  Umesto jedne aksiome bi se onda korstile dve, gde bi opštija imala manji prioritet. Na sličan način se ovaj pristup primenjuje i u slučajevima gde ima više grananja, i tada bi se uvodilo više novih aksima, ali te nove aksiome nisu uvek tačne pa je potrebno da se prvo dokažu unutar sistema.

- *Rad sa jednakostima*: kod teorija koje imaju jednakost, aksiome jednakosti se ne koriste eksplicitno. Umesto toga se koriste klase ekvivalencije za jednakost konstanti. Svaka konstanta u svojoj klasi ima predstavnika, koji je dovoljan za razmatranje.

- *Rad sa simetričnim predikatima*: kod činjenica koje imaju simetrične predikate, umesto da se razmatra svaka činjenica koja ima isto značenje, a samo

zamenjene argumente, razmatra se samo njihov predstavnik. Predstavnik klase činjenica je činjenica sa sortiranim argumentima na simetričnim indeksima.

Primer 3.1 *Razmotrimo primer dokaza ArgoCLP-om (uzet iz [28]). Sa* $bet(A, B, C)$ *obeležavamo da je tačka* B *između tačaka* A *i* C*, a sa* $col(A, B, C)$ *obeležavamo da su tačke* A*,* B *i* C *kolinearne. Teorema onda glasi:*

Teorema 3.1 *Ako važi* $bet(A, B, C)$*,* $AB \cong AD$ *i* $CB \cong CD$*, pokazati da važi* $B = D$*.*

Dokaz:

1. It holds that $bet(B, A, A)$ *(using th 3_ 1).*
2. From the fact(s) $bet(A, B, C)$ *it holds that* $col(C, A, B)$ *(using ax 4_ 10_ 3).*
3. From the fact(s) $AB \cong AD$ *it holds that* $AD \cong AB$ *(using th_ 2_ 2).*
4. It holds that $A = B$ *or* $A \neq B$*.*
5. Assume that: $A = B$*.*
6. From the fact(s) $AD \cong AB$ *and* $A = B$ *it holds that* $AD \cong AA$*.*
7. From the fact(s) $AD \cong AA$ *it holds that* $A = D$ *(using ax_ 3).*
8. From the fact(s) $A = B$ *and* $A = D$ *it holds that* $B = D$*.*
9. The conclusion follows from the fact(s) $B = D$*.*
10. Assume that: $A \neq B$*.*
11. It holds that $A = C$ *or* $A \neq C$*.*
12. Assume that: $A = C$*.*
13. From the fact(s) $bet(A, B, C)$ *and* $A = C$ *it holds that* $bet(A, B, A)$*.*
14. From the fact(s) $bet(A, B, A)$ *and* $bet(B, A, A)$ *it holds that* $A = B$ *(using th_ 3_ 4).*
15. From the fact(s) $A \neq B$ *and* $A = B$ *we get contradiction.*
16. Assume that: $A \neq C$*.*
17. From the fact(s) $A \neq C$ *it holds that* $C \neq A$*.*
18. From the fact(s) $C \neq A$ *and* $col(C, A, B)$ *and* $CB \cong CD$ *and* $AB \cong AD$ *it holds that* $B = D$ *(using th_ 4_ 18).*
19. The conclusion follows from the fact(s) $B = D$*.*
20. The conclusion follows in all cases.
21. The conclusion follows in all cases.

QED

Glava 4

Problem geometrijskih konstrukcija

Koncept geometrijskih konstrukcija je proučavan gotovo hiljadama godina, i još od vremena antičke Grčke (naročito nakon Euklidovih *Elementa* [13]) postao je standardni deo obrazovanja. Rigoroznost geometrijskih konstrukcija smatrana je ključnom za razvoj logičkog mišljenja, što je dovelo do modernizacije geometrije, pri čemu je posebno istaknut doprinos Hilbertovog dela *Grundlagen der Geometrie* [14]. Savremeni pristup klasičnoj sintetičkoj geometriji i dalje se oslanja na Hilbertove ideje, dok su problemi geometrijskih konstrukcija ostali jedan od najrigoroznijih i najprivlačnijih delova geometrije [9].

Problemi konstrukcije trougla su problemi u kojima je potrebno da se pomoću lenjira i šestara konstruiše trougao koji zadovoljava data ograničenja. Podrazumeva se da lenjir nema oznake koje se mogu koristiti za merenja. Konstrukcija pomoću lenjira i šestara je niz specifičnih, primitivnih konstruktivnih koraka, a to su [22]:

- konstrukcija *proizvoljne* tačke (potencijalno različite od nekih datih tačaka),
- konstrukcija (korišćenjem *lenjira*) prave koja prolazi kroz dve date tačke,
- konstrukcija (korišćenjem *šestara*) kruga koji ima centar u nekoj od datih tačaka, tako da mu neka druga data tačka pripada,
- konstrukcija preseka (ako postoji) dva kruga, dve prave ili prave i kruga.

Problemi geometrijske konstrukcije imaju rešenja koja su tradicionalno sačinjena iz četiri faze [9, 22]:

Analiza: Obično se kreće od pretpostavke da određeni geometrijski objekti zadovoljavaju specifikaciju problema Γ i dokazuje se da važe svojstva Λ koja omogućavaju konstrukciju.

Konstrukcija: Konstrukcija je zasnovana na analizi, tj. na svojstvima Λ koji se u okviru nje dokazazuju. U ovoj fazi se opisuje konstrukcija pomoću lenjira i šestara. Konstrukcija je po svojoj prirodi formalna i predstavlja opis postupka, odnosno skup formalnih i apstraktnih koraka (zasnovanih na upotrebi apstraktnih instrumenata kao što su lenjir i šestar), a ne sliku (koja se, na primer, može dobiti na papiru korišćenjem lenjira i šestara). Takva slika može olakšati razumevanje opisa konstrukcije, ali je ni u kom slučaju ne može zameniti.

Dokaz: U ovoj fazi je potrebno dokazati da ako figura dobijena konstrukcijom na osnovu datog opisa (tj. ako zadovoljava uslove Λ), tada zadovoljava i specifikaciju problema (tj. uslove Γ).

Diskusija: U fazi diskusije razmatra se koliko rešenja datog problema postoji i pod kojim uslovima su ta rešenja moguća.

4.1 ArgoTriCS sistem

Automatsko rešavanje konstruktivnih problema bi bilo od koristi milionima učenika koji rešavaju ovu vrstu problema u školama, takođe bi moglo biti od koristi i u nekim industrijskim primenama, npr. u robotici, kada se neki problem može svesti na konstruktivni problem [21]. Ovde razmatramo sistem za automatsko konstruisanje trouglova *ArgoTriCS* (**A**utomated **R**easoning **G**r**O**up **Tri**angle **C**onstruction **S**olver) [22]. ArgoTriCS je alat koji, sa nekim raspoloživim geometrijskim znanjem, rešava problem geometrijske konstrukcije.

Ključna svojstva sistema ArgoTriCS su [21]:

- identifikovanje relevantnog geometrijskog znanja,
- pogodna reprezentacija znanja koja omogućava efikasnu pretragu,
- relativno jednostavan sistem pretrage sa ulančavanjem unapred,
- navođenje kojim se postiže da su svi novi objekti koji se kreiraju tokom konstrukcije potencijalno relevantni za tu konstrukciju,

- mehanizmi za prepoznavanje simetričnih problema, redundantnih problema i problema zavisnih od položaja,
- mehanizmi koji omogućavaju vezu sa automatskim dokazivačima teorema i proveravanje ispravnosti generisanih konstrukcija,
- mehanizmi koji omogućavaju vezu sa interaktivnim dokazivačima teorema i formalno verifikovanje generisanih rešenja.

Rešavanje konstruktivnog problema sistemom ArgoTriCS obuhvata [21]:

- automatsko generisanje neformalnog opisa konstrukcije na prirodnom jeziku,
- automatsko generisanje formalne specifikacije konstrukcije korišćenjem jezika GCLC [16] uz generisanje odgovarajuće ilustracije,
- dokazivanje ispravnosti konstrukcije automatskim dokazivačima (dokazivačem OpenGeoProver [24] i dokazivačima koji postoje u okviru alata GCLC),
- generisanje ulaza za fazu diskusije u kojoj se određuje pri kojim uslovima postoji rešenje problema, kao i koliko rešenja problem ima,
- formalizaciju kompletnog rešenja konstruktivnog problema

ArgoTriCS je uspeo da reši 66 od 74 rešiva problema iz Vernikove liste [30], uspešno je detektovao sve redundantne probleme, probleme zavisne od položaja i simetrične probleme. Prema strožijem razmatranju korpusa, korišćen je za rešavanje svih 560 trojki tačaka iz Vernikove liste, od tih je identifikovao 268 rešivih, 93 zavisnih od položaja i 7 redundantnih, a nije uspeo da reši 192 trojki tačaka. Koristeći algebarske dokazivače dokazano je da su 166 od njih nerešive, dok su ostale rešive, ali je potrebno da se doda neko novo znanje u sistem da bi se rešile.

Iz Kornelijeve liste [7], uspešno je rešio 62 od 73 rešivih problema i, kao kod Vernikove, uspešno je detektovao sve redundantne probleme, probleme zavisne od položaja i simetrične probleme. U strožijem razmatranju korpusa, od 580 problema, identifikovao je 223 rešivih, 84 zavisnih od položaja i 9 redundantnih, dok ostalih 264 nije mogao da reši sa datim znanjem. Bitno je napomenuti da u Konelijevoj listi postoji puno instanci za koje je dokazano da su nerešive ili čiji status još uvek nije utvrdjen.

Primer

ArgoTriCS nakon pronalaska konstrukcije, može da je izvede u nekoliko različitih formata. Podržan je izlaz u prirodnom jeziku (na engleskom jeziku, u LaTeX formatu), kao i izlaz u GCLC jezik [16]. Primer generisane konstrukcije na prirodnom jeziku se može videtu u izlazu programa za sledeći problem (problem 32: A, O, H_a sa Vernikove liste):

Problem 32: Given a point A, a point O, and a point H_a construct the triangle ABC.

Construction:

1. Using the point A and the point H_a construct a line h_a (rule W02);
   % DET: points A and H_a are not the same;
2. Using the point A and the point O construct a circle $k(O, C)$ (rule W06);
   % NDG: points A and O are not the same;
3. Using the point H_a and the line h_a construct a line a (rule W10);
4. Using the circle $k(O, C)$ and the line a construct a point C and a point B (rule W04);
   % NDG: line a and circle $k(O, C)$ intersect.

Non-degenerate conditions: line a and circle $k(O, C)$ intersect; points A and O are not the same.

Determination conditions: points A and H_a are not the same.

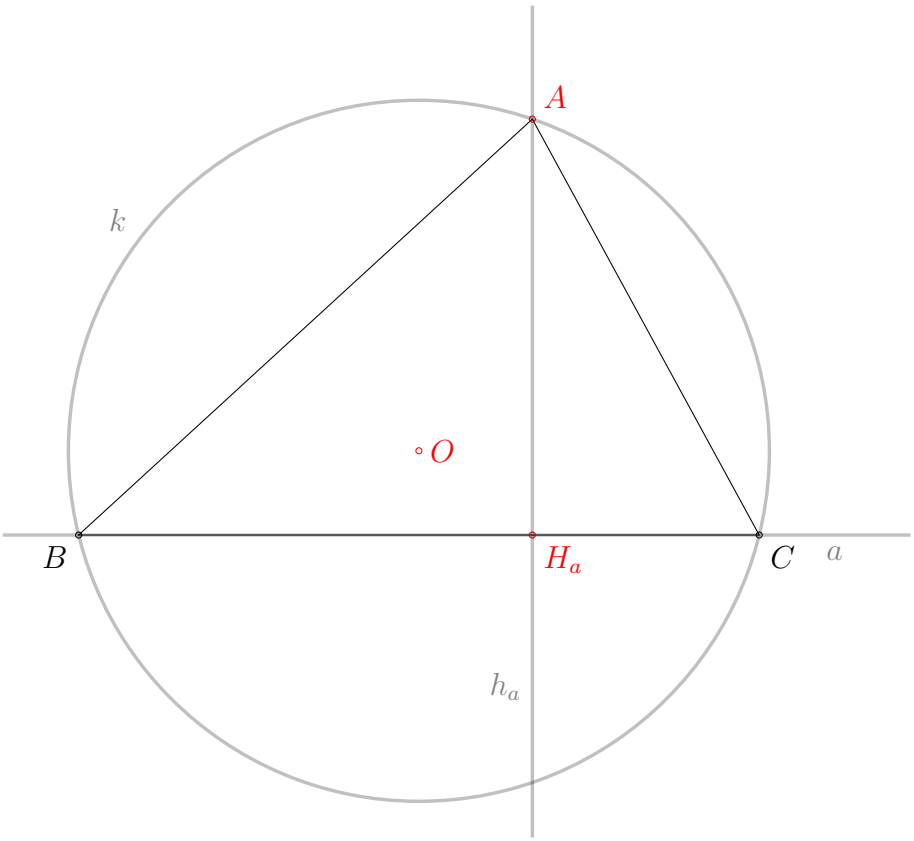
Rules used: [W02, W04, W06, W10a]

Definitions and lemmas used: [D5, D8, D26, GD01, L11, L12]

Pored izlaza na prirodnom jeziku, generiše se i izlaz formulisan korišćenjem geometrijskog jezika GCLC, kojim se zadaje konstrukcija na formalan način. GCLC takođe omogućava vizuelizaciju konstrukcije, i vizuelni prikaz generisane konstrukcije gornjeg primera dat je na slici 4.1

Dokaz korektnosti konstrukcije

ArgoTriCS dokazuje korektnost generisanih konstrukcija pomoću alata GCLC (koji ima podršku za tri metode automatskog dokazivanja geometrijskih teorema:



Slika 4.1: Automatski generisana ilustracija

metoda površina, Vuova metoda i metoda Grebnerovih baza) i dokazivača OpenGeoProver (koji ima implementiranu Vuovu metodu) [22]. Ovi dokazivači uspevaju da dokažu korektnost konstrukcije ali nemaju tradicionalan, čitljiv dokaz. U radu [23] razmatra se korišćenje dokazivača u logici prvog reda i u koherentnoj logici za problem dokazivanja korektnosti konstrukcija. Koristio se dokazivač u logici prvog reda Vampire [27], koji je vrlo efikasan, ali mana mu je to što ne generiše čitljiv dokaz. Takođe se koristio dokazivač u koherentnoj logici Larus [17].

Da bi dokazivači mogli da dokažu date konstrukcije, bilo je potrebno da se dodaju mnoge leme visokog nivoa kao deo aksiomatskog sistema. Ove leme slede iz opšteg geometrijskog znanja, ali su instancirane za bitne tačke, prave i kružnice vezane za trougao ABC. Svaka od lema je predstavljena konstantom (npr. tačke A, B i C su predstavljene konstantama pA, pB i pC redom. Takođe je bilo potrebno da se dokaže tačnost svih korišćenih lema, pa su leme su bile formalno dokazane u Isabelle/HOL [26], koristeći aksiome geometrije.

Dokazivači su bili testirani na podskupu problema iz Vernikovog korpusa. U

podskupu je bilo 35 neizomorfnih problema, za svaki od kojih je pokušano da se dokaže korektnost konstrukcije koju je izveo ArgoTriCS. Vampire dokazivač je uspešno dokazao 31 od ovih problema, a Larusu je bilo potrebno dodatno navođenje, dodavanjem pomoćnih tvrđenja, kako bi uspeo da dokaže 20 problema, a ostale nije mogao da dokaže u datom vremenskom ograničenju

Ovde je evidentno da problem čitljivog dokaza korektnosti ArgoTriCS konstrukcije još uvek nije rešen. Potrebno je napraviti dokazivač koji može da podrži komplikovane aksiomatske sisteme, i da u relativno kratkom vremenu generiše dokaz date tvrdnje, ako je to u datom aksiomatskom sistemu moguće. U sledećoj glavi biće predstavljeno moguće rešenje ovog problema.

Glava 5

Algoritam za automatsko generisanje dokaza

Na osnovu primene prethodnih dokazivača na problem dokazivanja korektnosti konstrukcije, dokazivači ili nisu mogli da generišu čitljiv dokaz, ili su, u pokušaju da to učine, na kompleksnijim primerima bili previše spori. Ovde se opisuje algoritam koji je prilagođen ovom konkretnom problemu kao moguće rešenje.

5.1 Domen problema

Potrebno je razmotriti moguće probleme koji se mogu očekivati. Na ulazu, program treba da primi skup aksioma i teoremu koju treba dokazati, dok na izlazu treba da vrati dokaz teoreme, ako on postoji. Kod konstrukcije trouglova, svi relevantni članovi trougla (npr. tačke A, B i C ili prave a, b ili c) su instancirani kao konstante čak iako ne „postoje" pre konstrukcije. Te konstante postoje da bi odgovarajući članovi trougla mogli da se *definišu* pomoću aksioma. Aksiome definišu osobine odgovarajućih konstanti, ali to ne znači da one postoje pre konstrukcije. Teoreme koje su potrebne za dokaz konstrukcije uvode nove „konstruisane" konstante za koje je potrebno da se dokaže da su jednake već definisanim konstantama.

Primer 5.1 *Razmotrimo konstrukciju iz primera 4.1. Potrebno je konstruisati trougao ABC ako su dati dati: teme A, podnožje visine H_a i centar opisane kružnice O. Konstrukcija koju daje ArgoTriCS glasi:*

1. *Konstruisati pravu $h_a = AH_a$.*

2. *Konstruisati kružnicu k sa centrom O koja sadrži tačku A .*

3. *Konstruisati pravu a tako da je normalna na pravu h_a i sadrži tačku H_a.*

4. *Neka su tačke B i C tačke dobijene presekom kružnice k i prave h_a*

Da bi se ova konstrukcija dokazala, treba pokazati da je A teme konstruisanog trougla (trivijalno), da je H_a podnožje visine ($pHa_1 = pHa$) i da je O ($pO_1 = pOc$) centar opisanog kruga. Hipoteze teoreme koju koristimo za dokaz se dobijaju iz konstrukcije, i glase:

- *$inc\_c(pA, k)$, $inc\_c(pB, k)$ i $inc\_c(pC, k)$ - krug k sadrži sva tri temena A (pA), B (pB) i C (pC),*
- *$center(pO_1, k)$ - centar kruga k je tačka O (pO_1),*
- *$inc(pB, a_1)$ i $inc(pC, a_1)$ - konstruisana prava a (a_1) sadrži tačke B (pB) i C (pC),*
- *$inc(pA, ha_1)$ i $perp(ha_1, a_1)$ - prava ha (ha_1) sadrži tačku A (pA) i normalna je na pravu a (a_1),*
- *$inc(pHa_1, a_1)$ i $inc(pHa_1, ha_1)$ - tačka H_a (pHa_1) pripada pravama a (a_1) i ha (ha_1).*

Iz datih hipoteza i ciljeva, teorema koju treba dokazati je oblika:

Teorema 5.1 $th\_A\_O\_Ha : inc\_c(pA, k) \wedge inc\_c(pB, k) \wedge inc\_c(pC, k) \wedge center(pO_1, k) \wedge inc(pB, a_1) \wedge inc(pC, a_1) \wedge inc(pA, ha_1) \wedge perp(ha_1, a_1) \wedge inc(pHa_1, a_1) \wedge inc(pHa_1, ha_1) \Rightarrow pHa_1 = pHa \wedge pOc = pO_1$

ArgoTriCS prilikom konstrukcije kreira sve potrebne konstante i svedoke. Odatle se vidi da pri dokazu konstrukcije nije potrebno stvaranje novih svedoka, nego se koriste konstante uvedene tokom konstrukcije, pa se *eliminiše potreba za aksiomama sa egzistencijalnim kvantifikatorima*, a osobine uvedenih konstanti se prosleđuju preko hipoteza teoreme. Pošto se u dosadašnjem radu sa dokazivanjem konstrukcija u koherentnoj logici nije bavilo sa uslovima nedegenerisanosti [23], pravi se pretpostavka da je za sada u zaključku dovoljna jedna konjunkcija, tj. da se iz zaključka mogu *eliminisati disjunkcije*, s obzirom na to da je moguće da disjunkcije proizlaze iz uslova nedegenerisanosti.

Dakle, prilagođeni dokazivač radi u fragmentu koherentne logike u kom nema egzistencijalnih kvantifikatora i disjunkcija, gde su formule oblika:

$$A_1(\boldsymbol{x}) \wedge ... \wedge A_n(\boldsymbol{x}) \Rightarrow C_1(\boldsymbol{x}) \wedge ... \wedge C_m(\boldsymbol{x})$$

gde se podrazumeva da su univerzalno kvantifikovane i gde: $0 \leq n$, $1 \leq m$, $\boldsymbol{x}$ označava niz promenljivih $x_1, ..., x_k$ $(0 \leq k)$, i A_i (za $1 \leq i \leq n$) i C_j (za $1 \leq j \leq m$) označavaju atomične formule.

5.2 Opis algoritma

Sve aksiome mogu da se pojednostave pomoću pravila $(\wedge E)$ iz poglavlja 3.1, tako što iz jedne aksiome, koja ima m konjunkta u zaključku, nastanu m aksioma sa istim premisama i po jednim od odgovarajućih konjunkta u zaključku, tj. iz aksioma:

$$A_1(\boldsymbol{x}) \wedge ... \wedge A_n(\boldsymbol{x}) \Rightarrow C_1(\boldsymbol{x}) \wedge ... \wedge C_m(\boldsymbol{x})$$

se izvode aksiome:

$$A_1(\boldsymbol{x}) \wedge ... \wedge A_n(\boldsymbol{x}) \Rightarrow C_j(\boldsymbol{x})$$

za $1 \leq j \leq m$.

Aksiome koje nemaju premise zovemo *činjenice*, jer odmah daju informacije o odgovarajućim konstantama (ili promenljivama), dok se ostale aksiome koriste kako bi se izvele nove činjenice. U daljem tekstu, pod pojmom „aksiome" podrazumevamo isključivo one koje nisu činjenice. Za izvođenje se koristi redukovana verzija pravila (ax) iz poglavlja 3.1:

$$\frac{A_1(\vec{a}, \vec{y}), ..., A_n(\vec{a}, \vec{y}) \qquad A_1(\vec{x}) \wedge ... \wedge A_n(\vec{x}) \Rightarrow \mathcal{C}(\vec{x})}{\mathcal{C}(\vec{a}, \vec{x})}\, rule$$

gde polazne činjenice i činjenica koja se izvodi mogu da sadrže samo konstante, samo promenljive, ili kombinaciju konstanti i promenljivih.

U memoriji se čuva skup svih datih i izvedenih činjenica koji je organizovan tako da se provera da li činjenica pripada skupu vrši efikasno. Jedna moguća implementacija takvog skupa je sortirano binarno stablo činjenica i ono se koristi u implementaciji

Algoritam koristi pristup ulančavanja unapred pomoću pretrage u širinu. Na početku, premise teoreme postaju činjenice i čuvaju se sa ostalim činjenicama, tj. pretpostavlja se da su tačne. Konjunkti zaključka se čuvaju u zasebnom skupu.

U svakoj iteraciji se prolazi kroz listu aksioma, i za svaku aksiomu se proverava primenjivost u skladu sa pravilom *rule*. Ako je aksioma primenjiva instanciranjem datih konstanti i ako rezultujuća činjenica ne postoji u skupu činjenica, može da se doda u taj skup. Ako je nova činjenica jednaka jednom od zaključaka, taj zaključak se briše iz skupa zaključaka. Ovaj postupak se ponavlja sve dok se ne isprazni skup zaključaka, ili dok se ne dodje do maksimalne dozvoljene dubine, koja je konstantna i određena pre pokretanja programa. Postupak može da se zaustavi rano ako se u jednoj iteraciji ne izvede nijedna nova činjenica, ili ako je neka nova izvedena činjenica jednaka $\perp$, jer je tad došlo do kontradikcije. U slučaju kontradikcije, program korisnika obaveštava da je skup aksioma i pretpostavki teoreme nekonzistentan i ispisuje kojim koracima se dolazi do zaključka $\perp$.

Pomoćne aksiome

Pored aksioma koje se učitavaju iz specifikacije problema, na početku se dodaju pomoćne aksiome (iz [17]):

- *Osobine jednakosti (i nejednakosti):* ako se u aksiomatskom sistemu koristi jednakost, potrebno je da se koriste dodatne aksiome koje opisuju njenu simetriju i refleksivnost, kao i simetriju nejednakosti:
  - Simetrija jednakosti: $\forall X, Y\ (X = Y \Rightarrow Y = X)$
  - Simetrija nejednakosti: $\forall X, Y\ (X \neq Y \Rightarrow Y \neq X)$
  - Refleksivnost jednakosti: $\forall X\ (X = X)$

  Umesto dodavanja aksioma za ove slučajeve, efikasnije bi bilo korišćenje klasa ekvivalencije pomoću *union-find* strukture (slično radu sa jednakostima kod ArgoCLP [29] dokazivača, opisanim u poglavlju 3.3).

- *Zamena argumenta:* za svaki predikat u signaturi uvode se aksiome koje omogućavaju da se neki od argumenata zameni nekim novim, ako su oni jednaki. Potrebno je da se za svaki predikat doda broj aksioma jednak njegovoj arnosti. Jedna aksioma odgovara jednom indeksu argumenta koji se zamenjuje. Dakle, za svaki predikat p arnosti n pravi se n aksioma oblika:

$$\forall X, Y_1, ..., Y_n\ (p(Y_1, ..., Y_n) \wedge X = Y_i \Rightarrow p(Y_1, ..., Y_{i-1}, X, Y_{i+1}, ..., Y_n))$$

za $1 \leq i \leq n$. Ako se koriste klase ekvivalencije kao rešenje za slučaj jednakosti, onda se, umesto dodavanja aksioma zamene, problem lako rešava pomoću *kongruentnog zatvorenja* jednakosti i predikatskog simbola.

- *Kontradikcija iz negacije:* u opisu koherentne logike u glavi 3 je rečeno da koherentna logika ne sadrži negaciju predikata, pa se za svaki predikat P uvodi predikat $\overline{P}$ koji predstavlja njegovu negaciju. Ako se za neki predikat p pojavljuje $\overline{p}$ u aksiomama ili teoremi, onda se uvodi aksioma:

$$\forall \vec{x}\ (p(\vec{x}) \wedge \overline{p}(\vec{x}) \Rightarrow \bot)$$

Zasićenje aksioma

Pre pretrage je moguće da se generiše još korisnih aksioma tzv. *zasićenjem* (saturation) (iz [17]). Uzima se lista „jednostavnih" aksioma, tj. aksioma sa samo jednom premisom koje mogu da izvedu nove aksiome (i činjenice) iz postojećih aksioma (i činjenica). Ako jednostavna aksioma može da se primeni na zaključak druge aksiome, tom primenom bi se izvela nova aksioma.

Primer 5.2 *Ako postoje aksiome:*

- $perp\_sym:\ \forall L1, L2\ (perp(L1, L2) \Rightarrow perp(L2, L1))$
- $perp\_para:\ \forall Lba, Lha, A\ (perp(Lha, A) \wedge para(Lba, Lha) \Rightarrow perp(Lba, A))$

primenom jednostavne aksiome $perp\_sym$ *na zaključak aksiome* $perp\_para$ *dobije se* $perp(A, Lba)$ *i izvedena aksioma izgleda kao:*

$$perp\_para1:\ \forall Lba, Lha, A\ (perp(Lha, A) \wedge para(Lba, Lha) \Rightarrow perp(A, Lba)).$$

Aksiome izvedene na ovaj način nisu neophodne za dokaz, dokaz može da se dobije i bez njihovog korišćenja. Ipak, one su korisne jer omogućavaju da se aksioma, koja bi inače bila izvedena u dva koraka (korišćenjem ove dve aksiome iz kojih je izvedena nova), izvede u jednom koraku. Postavlja se pitanje da li se zasićenjem aksioma (oba tipa) poboljšava efikasnost, ili je bolje zasititi samo jednostavne aksiome, ili čak ni jedne ni druge. O tome će biti reči u evaluaciji, u poglavlju 7.1.

Glava 6

Implementacija

Dokazivač je programiran u jeziku C++ i ima ukupno oko 2000 linija koda, organizovanih u 9 izvornih datoteka[1]. Kod je organizovan na sledeći način: u datotekama *common.h* i *common.cpp* se nalaze globalne definisane konstante (poput maksimalne dubine pretrage) i funkcije definisane izvan klasa; u datotekama $formula.h$ i $formula.cpp$ se nalaze klase svih logičkih objekata sa kojima se rukuje (termovi, atomične formule, formule u koherentnoj logici, itd.) i korisne metode nad njima (poput poređenja); u datotekama *theory.h* i *theory.cpp* je klasa teorije, koja čuva i manipuliše aksiome i teoremu (npr. metodom *zasićenja*); a u datotekama *prover.h* i *prover.cpp* je klasa dokazivača koja koristi teoriju i metode pretrage kako bi izvela dokaz.

Signatura, skup aksioma i teorema koja se dokazuje se unose preko fajlova u TPTP formatu (koristeći izmenjenu implementaciju učitavanja iz [17]), gde je signatura implicitno definisana iz unetih aksioma i teoreme. Učitavanje aksioma kao deo problema omogućava dokazivaču da dokazuje u proizvoljnoj teoriji (sve dok je teorija u fragmentu koherentne logike opisanom u poglavlju 5.1).

Činjenice se čuvaju u sortiranom binarnom stablu $facts$, jednostavne aksiome čuvaju se odvojeno od ostalih u nizu *simpleAxioms*, dok se ostale čuvaju u nizu *complexAxioms*. Na početku, posle učitavanja problema, aksiome se prvo pojednostavljuju pomoću pravila ($\wedge E$) (objašnjeno u poglavlju 5.2), pa se unose u odgovarajući niz (ili skup) u zavisnosti od njihovog tipa.

Nakon toga, u teoriju se dodaju pomoćne aksiome (navedene u 5.2), i izvršava se zasićenje aksioma (objašnjeno u 5.2). Pseudokod funkcije zasićenja je napisan u algoritmu 1. Nakon zasićenja se pokreće funkcija dokazivanja, koja deli proces

[1] https://github.com/vita-ride/geo-construction-prover

primenjivanja po tipu aksioma. Pri primenjivanju jednostavnih aksioma nad činjenicama koristi funkciju zasićenja i prosleđuje joj skup $facts$, a pri primenjivanju ostalih aksioma ima novu funkciju $generateFacts$, koja pokušava da primeni svaku od premisa jednu po jednu, i nakon što uspe da primeni jednu (i potencijalno instancira neke promenljive), eliminiše je i nastavlja dalje. Cilj je da eliminiše svaku od premisa, i da dobijeni zaključak izvede kao novu činjenicu ako ona već ne postoji. Pseudokod funkcije $generateFacts$ je napisan u algoritmu 2.

Algoritam 1 Zasićenje - funkcija saturate

```
Input: axioms                                   ▷ Skup aksioma/činjenica
  updated ← false
  do
      updated ← false
      for all ax ∈ simpleAxioms do
          for all ax2 ∈ axioms do
              if canSaturate(ax, ax2) then
                  newAx = saturate(ax, ax2)
                  if newAx ∉ allAxioms then
                      allAxioms.add(newAx)
                      updated ← true
                  end if
              end if
          end for
      end for
  while updated
```

Funkcija $canMerge$ (koja se koristi u algoritmu 2) pokušava da uklopi odabranu premisu aksiome i neku činjenicu sa istim prediktom, i ako je uspešna, na izlazu vraća *tačno* i aksiomu dobijenu njihovom unifikacijom smešta u promenljivu *merged* prosleđenu referencom. Postoji više provera koje se vrše pri unifikaciji kako bi se videlo da li je ona moguća, a i kako bi se osigurale odgovarajuće zamene promenljivih (ili drugim promenljivama ili konstantama). Razmotrimo naredni primer (podrazumeva se da univerzalne promenljive u premisama počinju sa velikim slovom, a konstante sa malim):

Primer 6.1 *Ako postoje*

- *premisa:* $A = B$,
- *činjenica:* $\forall X \ (X = X)$.

Algoritam 2 Generisanje novih činjenica - funkcija generateFacts

Input: $axiom$ ▷ Aksioma koja se primenjuje
Output: $newFacts$ ▷ Skup izvedenih činjenica

```
newFacts ← ∅
while axiom.numPremises > 0 do
    premise ← axiom.nextPremise()
    if ¬premise.hasUnivVariables() ∧ premise ∈ facts then
        axiom.popPremise()
    else
        break
    end if
end while
if axiom.numPremises = 0 then
    if axiom ∉ facts then
        newFacts.add(axiom)
        return newFacts
    end if
end if
premise ← axiom.nextPremise()
validFacts ← facts.findAllWithPredicate(premise.predicate)
for all fact ∈ validFacts do
    if canMerge(axiom, fact, &merged) then
        newFacts ← newFacts ∪ generateFacts(merged)
    end if
end for
return newFacts
```

Unifikacija može da se izvrši, i jedan mogući način je da se univerzalna promenljiva B zameni promenljivom A, pa bi dobijena premisa postala $A = A$, i takođe bi se svako pojavljivanje promenljive B u tekućoj aksiomi zamenilo promenljivom A. Ovaj korak se pravi kako bi se omogućilo dalje grananje u potrazi za primenjivim instanciranjem aksiome.

U narednom primeru dat je slučaj kada instanciranje nije moguće:

Primer 6.2 *Ako postoje*

- *premisa:* $midpoint(pA, pB, P)$,
- *činjenica:* $\forall X\ (midpoint(X, X, X))$.

Vidimo da nije moguće da se primeni (ako $pA \neq pB$), jer činjenica zahteva da sve tri tačke budu jednake, za promenljivu P ovo nije problem jer može da se

instancira u odgovarajuću konstantu, ali konstante pA i pB nisu jednake pa zahtev činjenice ne može da se ispuni.

U poređenju sa *ranim odsecanjem* kod algoritma ArgoCLP [29] (razmotrenim u poglavlju 3.3), kod ranog odsecanja prvo se instanciraju konstante na trenutnoj premisi pa se proverava da li postoji činjenica koja zadovoljava instanciranu premisu. Za razliku od toga, u ovoj implementaciji se prvo traže činjenice koje mogu da se iskoriste i izvodi se instanciranje pomoću njih (ako je to moguće). U ovom postupku, čak i ne moraju sve univerzalne promenljive da se odmah instanciraju nekom konstantom, nego može da se odloži za narednu premisu (ili da se ne instancira uopšte, i ovime se dobijaju činjenice koje imaju univerzalne promenljive, poput $\forall X\ (X = X)$).

Primer 6.3 *Poređenje instanciranja premise* $p(X, Y)$*:*

- Pristup dokazivača ArgoCLP sa ranim odsecanjem*: biraju se dve konstante* x_1 *i* y_1 *(odgovarajućeg tipa, ako je sistem tipiziran), promenljive* X *i* Y *se instanciraju njima, i nakon toga se pretražuje da li instancirana premisa* $p(x_1, y_1)$ *postoji u skupu činjenica.*

- Pristup u ovoj implementaciji*: pretražuje se skup svih činjenica sa prediktom* p*, bira se jedna i sa njom se pokušava unifikacija. Odabrana činjenica može da sadrži univerzalne promenljive, pa u nekim slučajevima promenljive* X *i* Y *se ne bi odmah instancirale, npr. sa činjenicom* $\forall A\ p(A, y_2)$ *bi instancirana premisa bila* $p(X, y_2)$ *i nije potrebna dodatna provera jer je dobijena korišćenjem postojeće činjenice.*

Univerzalne promenljive mogu da postoje i u premisi kojoj se proverava primenjivost i u činjenici sa kojom se vrši unifikacija. Kako bi se osiguralo da $canMerge$ funkcija može da savlada svaki slučaj potrebno je da se održavaju klase ekvivalencije, tj. ako neke dve promenljive predstavljaju isti argument u prediktu, one pripadaju istoj klasi. Ako se neka promenljiva pojavljuje na istom argumentu kao neka konstanta, onda ta konstanta postaje predstavnik klase te promenljive (u klasi nije moguće imati dve konstante). Ako nema konstante u nekoj klasi, za predstavnika se bira prva promenljiva (iz aksiome) u nizu (svaka aksioma čuva niz univerzalnih promenljivih). Klase se stvaraju pomoću *Union-Find* strukture podataka. Pojednostavljen pseudokod funkcije $canMerge$ je napisan u algoritmu 3, gde funkcija *representative* vraća predstavnika klase promenljive koja se prosledi, a

makeUnion spaja klase dve promenljive (ili promenljive i konstante) i vraća $false$ ako su predstavnici klasa konstante koje nisu jednake.

Algoritam 3 Provera primenjivosti - funkcija canMerge

Input: $axiom, fact, \&merged$
Output: $true \vee false$

```
premise ← axiom.nextPremise()
n ← premise.predicate.arity
i ← 0
while i < n do
    arg1 ← premise.argAt(i)
    arg2 ← fact.argAt(i)
    if representative(arg1) ≠ representative(arg2) then
        if ¬makeUnion(arg1, arg2) then
            return false
        end if
    end if
    i ← i + 1
end while
merged ← axiom
merged.popPremise()
for all prem ∈ merged.premises do
    for all arg ∈ prem.args do
        if arg.isUniversal then
            arg ← representative(arg)
        end if
    end for
end for
for all arg ∈ merged.conclusion.args do
    if arg.isUniversal then
        arg ← representative(arg)
    end if
end for
return true
```

U funkciji *prove* se prvo u skup činjenica dodaju premise iz teoreme, a onda se u petlji poziva funkcija generisanja dok se ne dodje do zaključka ili dok se ne dostigne maksimalna dozvoljena dubina. Pored funkcije generisanja, poziva se i funkcija zasićenja nad novo-generisanim činjenicama, ovo je zato što je funkcija zasićenja jednostavnija od funkcije generisanja, pa će se činjenice dobijene primenjivanjem jednostavnih aksioma dobiti brže. Pseudokod funkcije *prove* je napisan u algoritmu 4.

Algoritam 4 Dokaz - funkcija prove

Input: $theorem$
Output: $true \vee false$ ▷ Da li je uspeo da dokaže

```
goals ← ∅
for all premise ∈ theorem.premises do
    facts.add(premise)
end for
for all conclusion ∈ theorem.conclusions do
    goals.add(conclusion)
end for
depth ← 0
updated ← true
while depth < maxDepth ∧ goals ≠ ∅ do
    newFacts ← ∅
    for all axiom ∈ complexAxioms do
        generated ← generateFacts(axiom)
        if false ∈ generated then                    ▷ Kontradikcija
            printContradiction()
            return false
        end if
        newFacts ← newFacts ∪ generated
    end for
    if newFacts = ∅ then
        return false
    end if
    newFacts ← newFacts ∪ saturateFacts(newFacts)
    goals ← goals \ newFacts
    facts ← facts ∪ newFacts
    depth ← depth + 1
end while
if goals = ∅ then
    printProof()
    return true
end if
return false
```

Kako bi se generisao tekst dokaza, u memoriji se čuva usmeren aciklički graf zavisnosti činjenica. Jedan čvor grafa predstavlja jednu izvedenu činjenicu, i u njemu se nalaze informacije o svim činjenicama od kojih zavisi, kao i o aksiomi kojom je izvedena. Graf je implementiran uz pomoć heš tabele, jer je potrebno efikasno pronaći (ili uneti) informacije o proizvoljnoj činjenici u bilo kom trenutku, ne samo tokom iteracije. Informacije o novoj činjenici se formiraju tokom izvođenja, i unose se kada se činjenica doda u teoriju. Ispis dokaza konkretne činjenice se obavlja iteracijom kroz graf zavisnosti u dubinu, bez ponavljanja, gde se činjenice od kojih zavisi ispisuju prvo.

Glava 7

Evaluacija i primeri

U nastavku biće predstavljena uspešnost algoritma na skupu problema, kao i tabele vremena izvršavanja. Takođe, biće dati neki primeri rešenih problema.

7.1 Evaluacija

Dokazivač je bio testiran na istom podskupu problema iz Vernikovog korpusa [30] kao u radu [23]. U podskupu je bilo 34 neizomorfnih problema, i za svaki je pokušano da se dokaže korektnost konstrukcije koju je izveo ArgoTriCS. Dokazivač je uspešno dokazao 20 od ovih problema (i to istih 20 koje je rešio Larus [17]), a kod ostalih 14 je u kratkom vremenu prestao da izvodi nove činjenice, za razliku od Larusa, koji je za te probleme dostizao vremensko ograničenje. Uspešne dokaze je izveo potpuno autonomno, a Larusu je bilo potrebno dodatno navođenje, dodavanjem pomoćnih tvrđenja.

Iako dokazivač radi u redukovanoj logici (opisanoj u poglavlju 5.1) u poredjenju sa Larusom koji radi u CL, dobija slične rezultate. To daje nadu da je prelaz na redukovanu logiku dobra odluka. Eliminacija disjunkcija trenutno verovatno ne pravi razliku, jer se radi na dokazima konstrukcija bez uslova nedegenerisanosti. Iako je odluka da se eliminišu egzistencijalni kvantifikatori bila opravdana, pre testiranja nije moglo da se zna da li će to uticati na sposobnost dokazivača da reši sve relevantne probleme. Razlog zašto Larusi ovaj dokazivač imaju ovakve rezulate je najverovatnije zato što aksiomatski sistemi možda još uvek nisu potpuni.

Dalje razmatramo tabele vremena izvršavanja. Na tabeli 7.1 su prikazana vremena izvršavanja programa. U prvoj koloni se nalaze imena problema, u drugoj vremena izvršavanja, a treća prikazuje da li je dokaz uspešan.

Problem	Vreme	Uspeh	Problem	Vreme	Uspeh
A_B_Ma	1ms	Da	O_Ha_H	893ms	Da
A_B_G	3ms	Da	O_Ma_G	216ms	Da
A_B_H	8ms	Da	Ma_Hb_H	643ms	Da
A_O_Ha	8ms	Da	A_Hb_Ma	112ms	Ne
Ma_Mb_G	6ms	Da	O_Ma_Mb	421ms	Ne
A_Mb_H	10ms	Da	O_Ma_Hb	505ms	Ne
A_Ha_Hb	8ms	Da	Ma_Mb_Ha	215ms	Ne
A_Hb_Hc	9ms	Da	Ma_Mb_Hc	103ms	Ne
Ha_Hb_H	10ms	Da	Ma_Ha_Hb	359ms	Ne
A_Ma_Mb	18ms	Da	A_O_G	209ms	Ne
A_Mb_G	27ms	Da	A_G_Hb	419ms	Ne
A_O_Ma	112ms	Da	A_G_H	406ms	Ne
A_Mb_Mc	16ms	Da	Ma_G_Hb	295ms	Ne
Ma_G_H	227ms	Da	G_Ha_H	647ms	Ne
A_O_Hb	184ms	Da	A_O_H	234ms	Ne
O_G_Ha	837ms	Da	A_Ma_H	205ms	Ne
O_Ma_H	279ms	Da	Ma_Mb_Mc	269ms	Ne

Tabela 7.1: Vreme izvršavanja svih testiranih problema

Proveravamo i koji je najbolji pristup zasićenja (opisano u 5.2). Na tabeli 7.2 razmatraju se performanse za tri različita načina zasićenja aksioma. U prvoj koloni su imena problema sa uspešnim dokazom, u drugoj su vremena izvršavanja kada su zasićene samo jednostavne aksiome, u trećoj kada su zasićene sve aksiome, a u četvrtoj kada nijedna aksioma nije zasićena. Iz rezultata vidimo da se najbolji rezultati postižu kada su zasićene samo jednostavne aksiome, mada slučaj kada nijedna aksioma nije zasićena se ne razlikuje značajno. S druge strane, slučaj kada su sve aksiome zasićene pokazuje drastično lošije performanse.

7.2 Primeri

Primer 7.1 *Razmotrimo dokaz teoreme 5.1, sa nazivom $A\_O\_H_a$, objašnjenu u primeru 5.1. Formulacija teoreme glasi:*

Teorema 7.1 $th\_A\_O\_Ha : inc\_c(pA, k) \wedge inc\_c(pB, k) \wedge inc\_c(pC, k) \wedge center(pO_1, k) \wedge inc(pB, a_1) \wedge inc(pC, a_1) \wedge inc(pA, ha_1) \wedge perp(ha_1, a_1) \wedge inc(pHa_1, a_1) \wedge inc(pHa_1, ha_1) \Rightarrow pHa_1 = pHa \wedge pOc = pO_1$

Problem	Samo jednostavni	Svi aksiomi	Nijedni
A_B_Ma	1ms	2ms	1ms
A_B_G	3ms	5ms	3ms
A_B_H	8ms	14ms	9ms
A_O_Ha	8ms	12ms	8ms
Ma_Mb_G	6ms	9ms	6ms
A_Mb_H	10ms	16ms	11ms
A_Ha_Hb	8ms	15ms	10ms
A_Hb_Hc	9ms	15ms	9ms
Ha_Hb_H	10ms	16ms	11ms
A_Ma_Mb	18ms	41ms	17ms
A_Mb_G	27ms	59ms	29ms
A_O_Ma	112ms	231ms	123ms
A_Mb_Mc	16ms	36ms	17ms
Ma_G_H	227ms	530ms	258ms
A_O_Hb	184ms	385ms	205ms
O_G_Ha	837ms	1754ms	920ms
O_Ma_H	279ms	528ms	301ms
O_Ha_H	893ms	1758ms	922ms
O_Ma_G	216ms	501ms	232ms
Ma_Hb_H	643ms	1311ms	687ms

Tabela 7.2: Vreme izvršavanja za različite načine zasićenja

Potrebne aksiome:

- *Početne aksiome:*

$$
\begin{aligned}
(pHa\_def) &: \forall H_1\ (inc(H_1, ha) \wedge inc(H_1, bc) \Rightarrow H_1 = pHa) \\
(haA) &: \forall H\ (perp(H, bc) \wedge inc(pA, H) \Rightarrow ha = H) \\
(bc\_unique) &: \forall L\ (inc(pB, L) \wedge inc(pC, L) \Rightarrow L = bc) \\
(cc\_unique) &: \forall C\ (inc\_c(pA, C) \wedge inc\_c(pB, C) \wedge inc\_c(pC, C) \\
&\quad \Rightarrow C = cc) \\
(center\_unique) &: \forall C, C_1, C_2\ (center(C_1, C) \wedge center(C_2, C) \\
&\quad \Rightarrow C_1 = C_2)
\end{aligned}
$$

- *Izvedene pomoćne aksiome:*

$$(eq\_sym) : \forall A, B\ (A = B \Rightarrow B = A)$$
$$(incEqSub_1) : \forall A, B, X\ (inc(A, B) \wedge B = X \Rightarrow inc(A, X))$$
$$(perpEqSub_1) : \forall A, B, X\ (perp(A, B) \wedge B = X \Rightarrow perp(A, X))$$
$$(centerEqSub_1) : \forall A, B, X\ (center(A, B) \wedge B = X \Rightarrow center(A, X))$$

Potrebne činjenice:

- $center(pOc, cc)$

Dokaz dobijen iz izlaza dokazivača:

Theorem to prove:
$th\_A\_O\_Ha\ :\ inc\_c(pA, k) \wedge inc\_c(pB, k) \wedge inc\_c(pC, k) \wedge center(pO_1, k) \wedge inc(pB, a_1) \wedge inc(pC, a_1) \wedge inc(pA, ha_1) \wedge perp(ha_1, a_1) \wedge inc(pHa_1, a_1) \wedge inc(pHa_1, ha_1) \Rightarrow pHa_1 = pHa \wedge pOc = pO_1$
Assumptions:

$inc\_c(pA, k)$

$inc\_c(pB, k)$

$inc\_c(pC, k)$

$center(pO_1, k)$

$inc(pB, a_1)$

$inc(pC, a_1)$

$inc(pA, ha_1)$

$perp(ha_1, a_1)$

$inc(pHa_1, a_1)$

$inc(pHa_1, ha_1)$

It should be proved that:

$pHa = pHa_1$

$pOc = pO_1$

Proof found:

1. $k = cc$ *(from* $inc\_c(pC, k)$, $inc\_c(pB, k)$, $inc\_c(pA, k)$ *using* (cc_unique)*; instantiation:* $C \to k$*)*
2. $center(pO_1, cc)$ *(from* $k = cc$, $center(pO_1, k)$ *using* $(centerEqSub_1)$*; instantiation:* $A \to pO_1$, $B \to k$, $X \to cc$*)*
3. $pOc = pO_1$ *(from* $center(pO_1, cc)$, $center(pOc, cc)$ *using* $(center\_unique)$*; instantiation:* $C_1 \to pOc$, $C \to cc$, $C_2 \to pO_1$*)* ***(Goal proved)***
4. $a_1 = bc$ *(from* $inc(pC, a_1)$, $inc(pB, a_1)$ *using* (bc_unique)*; instantiation:* $L \to a_1$*)*
5. $inc(pHa_1, bc)$ *(from* $a_1 = bc$, $inc(pHa_1, a_1)$ *using* $(incEqSub_1)$*; instantiation:* $A \to pHa_1$, $B \to a_1$, $X \to bc$*)*
6. $perp(ha_1, bc)$ *(from* $a_1 = bc$, $perp(ha_1, a_1)$ *using* $(perpEqSub_1)$*; instantiation:* $A \to ha_1$, $B \to a_1$, $X \to bc$*)*
7. $ha = ha_1$ *(from* $inc(pA, ha_1)$, $perp(ha_1, bc)$ *using* (haA)*; instantiation:* $H \to ha_1$*)*
8. $inc(pHa_1, ha)$ *(from* $ha_1 = ha$, $inc(pHa_1, ha_1)$ *using* $(incEqSub_1)$*; instantiation:* $A \to pHa_1$, $B \to ha_1$, $X \to ha$*)*
9. $pHa_1 = pHa$ *(from* $inc(pHa_1, bc)$, $inc(pHa_1, ha)$ *using* (pHa_def)*; instantiation:* $H_1 \to pHa_1$*)*
10. $pHa = pHa_1$ *(from* $pHa_1 = pHa$ *using* (eq_sym)*; instantiation:* $A \to pHa_1$, $B \to pHa$*)* ***(Goal proved)***

Objašnjenje dokaza:

(1) ako krug k sadrži sva tri temena pA, pB i pC, onda je on opisan krug trougla ABC (jer je opisan krug trougla jedinstven),

(2,3) ako je k opisan krug trougla ABC, i pO_1 *njegov centar, onda je* pO_1 *centar opisanog kruga trougla ABC (jer je centar kruga jedinstven),* ovde se dokazuje cilj $pOc = pO_1$,

(4) ako prava a_1 sadrži tačke pB i pC onda je jednaka pravoj bc (jer je strana trougla ABC koja sadrži tačke B i C jedinstvena),

(6,7) ako prava ha_1 sadrži tačku pA i normalna je na pravu $bc = a_1$, onda mora da bude jednaka pravoj ha (jer u trouglu ABC postoji jedinstvena visina iz tačke A),

(5,8,9,10) ako tačka pHa_1 pripada pravama $ha = ha_1$ i $bc = a_1$, onda mora da bude jednaka tački pHa (jer je presek pravih jedinstven), ovde se dokazuje cilj $pHa = pHa_1$.

Nakon što su svi ciljevi dokazani, time je dokazana i teorema.

Primer 7.2 *Razmotrimo problem* 342 : $M_a\_M_b\_G$ *sa Vernikove liste [30]. Potrebno je konstruisati trougao ABC ako su dati: središte M_a, središte M_b i težište G. Konstrukcija koju daje ArgoTriCS glasi:*

1. *Konstruisati tačku A tako da je $\overrightarrow{AG} : \overrightarrow{AM_a} = 2 : 3$*
2. *Konstruisati tačku B tako da je $\overrightarrow{BG} : \overrightarrow{BM_b} = 2 : 3$*
3. *Konstruisati tačku C tako da je M_a središte duži $\overline{BC}$*

Vizuelni prikaz konstrukcije je dat na slici 7.1.

Da bi se konstrukcija dokazala, treba pokazati da su u konstruisanom trouglu ABC tačke M_a i M_b središta ($pMa = pMa_1$ i $pMb = pMb_1$), i da je tačka G težište ($pG = pG_1$). Hipoteze teoreme dobijene iz konstrukcije glase:

- $ratio23(pA, pG_1, pA, pMa_1)$ *- odnos rastojanja tačaka A (pA) i G (pG_1) sa rastojanjem tačaka A (pA) i M_a (pMa_1) je jednak* $2 : 3$,
- $ratio23(pB, pG_1, pB, pMb_1)$ *- odnos rastojanja tačaka B (pB) i G (pG_1) sa rastojanjem tačaka B (pB) i M_b (pMb_1) je jednak* $2 : 3$,
- $midpoint(pMa_1, pB, pC)$ *- tačka M_a pMa_1 se nalazi na sredini između tačaka B (pB) i C (pCA).*

Iz hipoteza i ciljeva dobija se teorema oblika:

Teorema 7.2 $th\_Ma\_Mb\_G : ratio23(pA, pG_1, pA, pMa_1) \wedge ratio23(pB, pG_1, pB, pMb_1) \wedge midpoint(pMa_1, pB, pC) \Rightarrow pMa = pMa_1 \wedge pMb = pMb_1 \wedge pG = pG_1$

Dalje se razmatra dokaz teoreme, prvo navodimo potrebne aksiome:

- *Početne aksiome:*

$$(midpoint\_unique) : \forall M_1, M_2, P_1, P_2\ (midpoint(M_1, P_1, P_2) \wedge midpoint(M_2, P_1, P_2) \Rightarrow M_1 = M_2)).$$

- *Izvedene pomoćne aksiome:*

$$(ratio23EqSub_1) : \forall A, B, C, D, X\ (ratio23(A, B, C, D) \wedge B = X \Rightarrow ratio23(A, X, C, D))$$
$$(ratio23EqSub_3) : \forall A, B, C, D, X\ (ratio23(A, B, C, D) \wedge D = X \Rightarrow ratio23(A, B, C, X))$$
$$(unique\_ratio23_1) : \forall A, B, X, Y\ (ratio23(A, X, A, B) \wedge ratio23(A, Y, A, B) \Rightarrow X = Y)$$
$$(unique\_ratio23_2) : \forall A, B, X, Y (ratio23(A, B, A, X) \wedge ratio23(A, B, A, Y) \Rightarrow X = Y)$$

Potrebne činjenice:

- $midpoint(pMa, pB, pC)$
- $ratio23(pA, pG, pA, pMa)$
- $ratio23(pB, pG, pB, pMb)$

Dokaz dobijen iz izlaza dokazivača:

Theorem to prove:
$th\_Ma\_Mb\_G : ratio23(pA, pG1, pA, pMa1) \wedge ratio23(pB, pG1, pB, pMb1) \wedge midpoint(pMa1, pB, pC) \Rightarrow pMa = pMa_1 \wedge pMb = pMb_1 \wedge pG = pG_1$
Assumptions:

$midpoint(pMa_1, pB, pC)$

$ratio23(pA, pG_1, pA, pMa_1)$

$ratio23(pB, pG_1, pB, pMb_1)$

It should be proved that:

$pMa = pMa_1$

$pMb = pMb_1$

$pG = pG_1$

Proof found:

1. $pMa = pMa_1$ *(from* $midpoint(pMa_1, pB, pC)$, $midpoint(pMa, pB, pC)$ *using* $(midpoint\_unique)$*; instantiation:* $M_1 -> pMa$, $M_2 -> pMa_1$, $P_1 -> pB$, $P_2 -> pC$*) (**Goal proved**)*

2. $pMa_1 = pMa$ *(from* $midpoint(pMa, pB, pC)$, $midpoint(pMa_1, pB, pC)$ *using* $(midpoint\_unique)$*; instantiation:* $M_1 -> pMa_1$, $M_2 -> pMa$, $P_1 -> pB$, $P_2 -> pC$*)*

3. $ratio23(pA, pG_1, pA, pMa)$ *(from* $pMa_1 = pMa$, $ratio23(pA, pG_1, pA, pMa_1)$ *using* $(ratio23EqSub_3)$*; instantiation:* $A -> pA$, $C -> pA$, $D -> pMa_1$, $B -> pG_1$, $X -> pMa$*)*

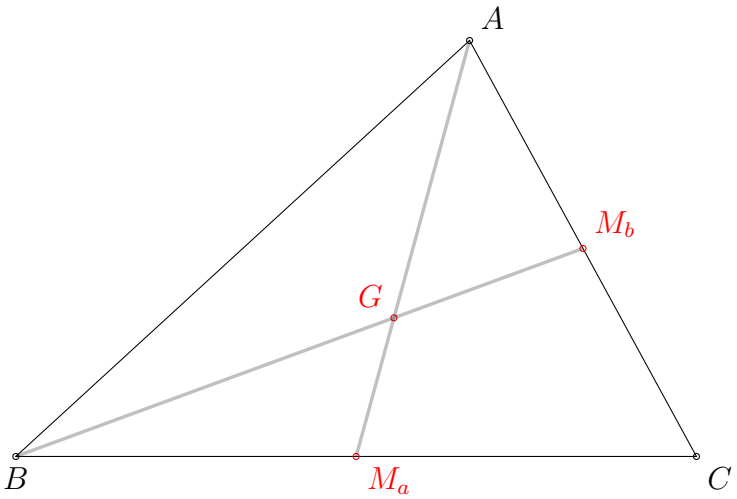
4. $pG = pG_1$ *(from* $ratio23(pA, pG_1, pA, pMa)$, $ratio23(pA, pG, pA, pMa)$ *using* $unique\_ratio23_1$*; instantiation:* $A -> pA$, $X -> pG$, $B -> pMa$, $Y -> pG_1$*) (**Goal proved**)*

5. $pG_1 = pG$ *(from* $ratio23(pA, pG, pA, pMa)$, $ratio23(pA, pG_1, pA, pMa)$ *using* $unique\_ratio23_1$*; instantiation:* $A -> pA$, $X -> pG_1$, $B -> pMa$, $Y -> pG$*)*

6. $ratio23(pB, pG, pB, pMb_1)$ *(from* $pG_1 = pG$, $ratio23(pB, pG_1, pB, pMb_1)$ *using* $ratio23EqSub_1$*; instantiation:* $A -> pB$, $C -> pB$, $D -> pMb_1$, $B -> pG_1$, $X -> pG$*)*

7. $pMb = pMb_1$ *(from* $ratio23(pB, pG, pB, pMb_1)$, $ratio23(pB, pG, pB, pMb)$ *using* $unique\_ratio23_2$*; instantiation:* $A -> pB$, $X -> pMb$, $B -> pG$, $Y -> pMb_1$*) (**Goal proved**)*

Objašnjenje dokaza:



Slika 7.1: Automatski generisana ilustracija konstrukcije problema $M_a\_M_b\_G$

(1,2) ako je tačka pMa_1 *na sredini između tačaka* pB *i* pC*, onda je ona središte (jer je središte bilo koje strane trougla* ABC *jedinstveno),* ovde se dokazuje cilj $pMa = pMa_1$,

(3,4,5) ako je odnos rastojanja tačaka pA *i* pG_1 *i rastojanja tačaka* pA *i* $pMa = pMa_1$ *jednak* $2:3$*, onda je tačka* pG_1 *težište trougla (jer je težište trougla jedinstveno),* ovde se dokazuje cilj $pG = pG_1$

(6,7) ako je odnos rastojanja tačaka pB *i* $pG = pG_1$ *i rastojanja tačaka* pB *i* pMb_1 *jednak* $2:3$*, onda je tačka* pMb_1 *središte (jer je središte bilo koje strane jedinstveno),* ovde se dokazuje cilj $pMb = pMb_1$

Nakon što su svi ciljevi dokazani, time je dokazana i teorema.

Glava 8

Zaključak

Dokazivanje korektnosti konstrukcije korišćenjem čitljivih dokaza je i dalje relevantan i nerešen problem. Čitljivi dokazi konstrukcija su izuzetno korisni u kontekstu matematičkog obrazovanja, gde je studentima potrebno da razumeju zašto je neka geometrijska tvrdnja tačna.

Razvijen je specijalizovan dokazivač za ovaj specifičan problem. Dokazivač koristi pristup ulančavanja unapred sa pretragom u širinu kako bi pronašao što kraće dokaze.

Dokazivač je testiran i pokazao se uspešnim na velikom broju problema, za koje efikasno pronalazi čitljiv dokaz. Uspeva da radi sa proizvoljnim komplikovanim aksiomatskim sistemima i sposoban je da izvede sve informacije koje su moguće u datom sistemu. Razmatranje konkretnih problema sa neuspešnim dokazima ukazuje na to da dostupni aksiomatski sistemi ne sadrže sve potrebne leme za njihovo rešavanje.

Problem dokazivanja korektnosti konstrukcija se generičkim dokazivačima za koherentnu logiku, poput Larusa [17], pokazao preteškim. Predstavljeni dokazivač prevazilazi generičke dokazivače u ovom problemu, jer je posebno prilagođen za njega.

U daljem radu je moguće poboljšati efikasnost ovog dokazivača mnogim unapređenjima, poput uvođenja pametnih heuristika u pretrazi i rada sa klasama ekvivalencije jednakosti. Dokazivač dodatno može da se proširi dodavanjem podrške za disjunkcije, što bi potecnijalno omogućilo rad sa uslovima nedegenerisanosti. Pored poboljšanja efikasnosti i proširenja dokazivača, potreban je dalji rad na formiranju širih aksiomatskih sistema novim lemama, koje bi naknadno morale biti formalno dokazane.

Literatura

[1] Nuno Baeta and Pedro Quaresma. Towards ranking geometric automated theorem provers. *Electronic Proceedings in Theoretical Computer Science*, 290:30–37, April 2019.

[2] Marc Bezem and Thierry Coquand. *Automating Coherent Logic*, page 246–260. Springer Berlin Heidelberg, 2005.

[3] Bruno Buchberger. Bruno buchberger's phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3–4):475–511, March 2006.

[4] S.-C. Chou, X.-S. Gao, and J.-Z. Zhang. Automated production of traditional proofs for constructive geometry theorems. In *[1993] Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 48--56, 1993.

[5] Shang-Ching Chou and Xiao-Shan Gao. *Automated Reasoning in Geometry*, page 707–749. Elsevier, 2001.

[6] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3):219–246, 2000.

[7] Harold Connelly. An extension of triangle constructions from located points. *Forum Geometricorum [electronic only]*, 9, 01 2009.

[8] Hans de Nivelle and Jia Meng. *Geometric Resolution: A Proof Procedure Based on Finite Model Search*, page 303–317. Springer Berlin Heidelberg, 2006.

[9] M. Djoric and P. Janičić. Constructions, instructions, interactions. *Teaching Mathematics and its Applications*, 23(2):69–88, June 2004.

[10] Roy Dyckhoff and Sara Negri. Geometrisation of first-order logic. *The Bulletin of Symbolic Logic*, 21(2):123–163, June 2015.

[11] Joran Elias. Automated geometric theorem proving: Wu's method. *The Mathematics Enthusiast*, 3(1):3–50, February 2006.

[12] Herbert L. Gelernter. Realization of a geometry theorem proving machine. In *IFIP Congress*, 1995.

[13] T. Heath, editor. *The thirteen books of Euclid's elements (3 vols)*. Dover, 2 edition, 1956.

[14] David Hilbert. *Grundlagen der Geometrie*. Teubner, fourth edition, 1913.

[15] Predrag Janičić and Stevan Kordić. Euclid - the geometry theorems prover. *Filomat*, 9(3):723--732, 1995.

[16] Predrag Janičić. Geometry constructions language. *Journal of Automated Reasoning*, 44(1–2):3–24, June 2009.

[17] Predrag Janičić and Julien Narboux. Theorem proving as constraint solving with coherent logic. *Journal of Automated Reasoning*, 66(4):689–746, May 2022.

[18] Predrag Janičić, Julien Narboux, and Pedro Quaresma. The area method: A recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, November 2010.

[19] Deepak Kapur. Using gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, December 1986.

[20] B. Kutzler and S. Stifter. Automated geometry theorem proving using buchberger's algorithm. In *Proceedings of the fifth ACM symposium on Symbolic and algebraic computation - SYMSAC '86*, SYMSAC '86, page 209–214. ACM Press, 1986.

[21] Vesna Marinković. *Automated Solving of Construction Problems in Geometry*. PhD thesis, University of Belgrade, 2015.

[22] Vesna Marinković. ArgoTriCS – automated triangle construction solver. *Journal of Experimental & Theoretical Artificial Intelligence*, 29(2):247–271, January 2016.

[23] Vesna Marinković, Tijana Šukilović, and Filip Marić. Towards automated readable proofs of ruler and compass constructions. *Electronic Proceedings in Theoretical Computer Science*, 398:11–20, January 2024.

[24] Filip Marić, Ivan Petrović, Danijela Petrović, and Predrag Janičić. Formalization and implementation of algebraic methods in geometry. *Electronic Proceedings in Theoretical Computer Science*, 79:63–81, February 2012.

[25] Arthur J. Nevins. Plane geometry theorem proving using forward chaining. *Artificial Intelligence*, 6(1):1--23, 1975.

[26] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

[27] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2-3):91--110, 2002.

[28] Sana Stojanović, Julien Narboux, Marc Bezem, and Predrag Janičić. *A Vernacular for Coherent Logic*, page 388–403. Springer International Publishing, 2014.

[29] Sana Stojanović, Vesna Pavlović, and Predrag Janičić. *A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs*, page 201–220. Springer Berlin Heidelberg, 2011.

[30] William Wernick. Triangle constructions with three located points. *Mathematics Magazine*, 55(4):227–230, September 1982.