УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Сана Н. Стојановић

ФОРМАЛИЗАЦИЈА И АУТОМАТСКО ДОКАЗИВАЊЕ ТЕОРЕМА ЕУКЛИДСКЕ ГЕОМЕТРИЈЕ

докторска дисертација

Београд, 2016.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Sana N. Stojanović

FORMALIZATION AND AUTOMATION OF EUCLIDEAN GEOMETRY

Doctoral Dissertation

Belgrade, 2016.

Ментор:

др Предраг Јаничић, редовни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Зоран Лучић, ванредни професор
Универзитет у Београду, Математички факултет

др Филип Марић, доцент
Универзитет у Београду, Математички факултет

др Мирјана Борисављевић, редовни професор
Универзитет у Београду, Саобраћајни факултет

др Жилиен Нарбу, професор
Универзитет у Стразбуру

Датум одбране: _____

Наслов дисертације: Формализација и аутоматско доказивање теорема еуклидске геометрије

Резиме: Напредак геометрије кроз векове се може разматрати кроз развој различитих аксиоматских система који је описују. Употреба аксиоматских система започиње са Еуклидом, наставља се са Хилбертом и Тарским али се ту не завршава. Чак и данас се развијају нови аксиоматски системи за рад са еуклидском геометријом. Аксиоматски систем Авигада се враћа на саме почетке и прецизно описује основна извођења представљена Еуклидовим „Елементима".

Записивање аксиоматског система у формату погодном за рачунарско доказивање теорема често је изазов сам по себи. Непрецизна формулација аксиома, која се јавља у књигама, долази до изражаја тек када их треба записати у облику погодан за рачунар. Формализација различитих аксиоматских система и доказивање уз помоћ рачунара у оквиру њима описаних теорија је главни мотив ове тезе.

Програми за доказивање теорема постоје још од осамдесетих година прошлог века и данас представљају колекцију веома моћних алата. У овој тези биће представљен систем за аутоматско и формално доказивање теорема који користи моћи резолуцијских доказивача теорема, кохерентни доказивач, али и интерактивне доказиваче теорема за верификовање генерисаних доказа. Кохерентни доказивач ArgoCLP је један од доприноса ове тезе. Развијен је и дијалект кохерентне логике, базиран на природној дедукцији, који омогућава једноставно трансформисање генерисаних доказа у доказе записане на језицима интерактивних доказивача Isabelle и Coq и у доказе записане на природном језику, енглеском и српском.

Систем за доказивање теорема ће бити примењен на три аксиоматска система еуклидске геометрије, илуструјући своју применљивост како приликом доказивања великих математичких теорија, тако и приликом верификовања неформалних доказа из математичких уџбеника.

Кључне речи: кохерентна логика, формализација геометрије, аутоматско доказивање теорема, интерактивно доказивање теорема, аутоматско генерисање читљивих доказа

Научна област: рачунарство

Ужа научна област: вештачка интелигенција

УДК број: [004.832.3:510.66]:[514.112/.113](043.3)

Dissertation title: Formalization and automation of Euclidean geometry

Abstract: The advance of geometry over the centuries can be observed through the development of different axiomatic systems that describe it. The use of axiomatic systems begins with Euclid, continues with Hilbert and Tarski, but it doesn't end there. Even today, new axiomatic systems for Euclidean geometry are developed. Avigad's axiomatic system goes back to the beginnings and precisely describes basic derivations presented in Euclid's „Elements".

Writing an axiomatic system in a format suitable for computer theorem proving is a challenge in itself. Imprecise formulations of axioms which appear in books get exposed only when they need to be written in a format suitable for computers. The formalization of different axiomatic systems and computer-assisted proofs within theories described by them is the main motif of this thesis.

The programs for theorem proving have existed since the eighties and today they present a collection of very powerful tools. This thesis presents a system for automated and formal theorem proving which uses the power of resolution theorem provers, a coherent prover, as well as interactive theorem provers for verifying the generated proofs. Coherent prover ArgoCLP is one of the contributions of the thesis. Additionally, the thesis develops a dialect of coherent logic based on natural deduction which enables simple transformation of generated proofs into proofs written in languages of interactive provers Isabelle and Coq as well as in natural languages, English and Serbian.

The system for theorem proving is applied to three axiomatic systems of Euclidean geometry, thus illustrating its applicability to both proving the large mathematical theories and verification of informal proofs from mathematical textbooks.

Keywords: coherent logic, formalization of geometry, automated theorem proving, interactive theorem proving, automated generation of readable proofs

кохерентна логика, формализација геометрије, аутоматско доказивање теорема, интерактивно доказивање теорема

Research area: Computer Science

Research sub-area:

UDC number: [004.832.3:510.66]:[514.112/.113](043.3)

Захвалница

Геометрија је област која ме је привукла још у средњој школи. Начин резоновања, дијаграми који је прате и извођење доказа у њој увек су ми привлачили пажњу. Након ње дуго нисам наилазила на област која ме је једнако занимала, све док нисам почела да слушам предмет Математичка логика у рачунарству код професора Предрага Јаничића. Математичка логика и ја смо се одмах некако нашли јер смо обоје превише детаљни и тешки за разумевање. Када сам, после завршених магистарских студија, почела да сарађујем поново са професором Предрагом Јаничићем, мој ентузијазам за науку је знатно порастао када сам открила да се он управо бави спојем моје две омиљене области. Од тада ми је професор био идеалан водич у мом путу ка освајању геометрије и разних аксиоматских система који је описују. Моју захвалност посебно дугујем њему, нарочито у последњим месецима израде ове тезе. Деловало је да његово стрпљење и помоћ коју ми је пружао некад нису имали границе. Упутила бих своју захвалност и Филипу Марићу који је у току мојих докторских студија мени био као други ментор. Без његове помоћи многи алати и радови који су описани у овој тези не би могли да буду развијени на исти начин. Такође изразиту захвалност бих упутила и члановима комисије, посебно професорки Мирјани Борисављевић која је несебично улагала своје време у то да моја теза постане још прецизнија, јаснија и боље написана.

Приликом писања ове тезе срела сам се са много више изазова него што сам мислила да је физички могуће поднети. Захваљујући мојој породици, првенствено мом супругу Жељку и нашој девојчици Вањи, који су веровали у мене у сваком тренутку и били моја неизмерна подршка, нашла сам снагу да све то успешно савладам. Њима ову тезу посебно посвећујем.

Садржај

1 Увод **1**

1.1 Introduction — Summary 4

2 Преглед основних појмова **7**

2.1 Заснивање геометрије 7

2.2 Аутоматско и интерактивно доказивање теорема 8

2.2.1 Аутоматско доказивање теорема 9

2.2.2 Интерактивно доказивање теорема 9

2.2.3 Интеграција интерактивног и аутоматског доказивања теорема 11

2.3 Доказивање теорема у геометрији уз помоћ рачунара 12

2.4 Кохерентна логика и аутоматско доказивање у кохерентној логици 14

2.4.1 Језик кохерентне логике и извођење доказа у кохерентној логици 14

2.4.2 Читљиви докази 16

2.5 Background — Summary 18

3 Доказивач ArgoCLP и проширења **20**

3.1 Опис доказивача ArgoCLP 21

3.1.1 Основна процедура претраживања 23

3.1.2 Напредна процедура претраживања 24

3.1.3 Технике које не чувају својство потпуности 28

3.2 Имплементација доказивача и начин коришћења 29

3.3 Евалуација 36

3.4 Трансформисање система аксиома 38

3.4.1 Симетрични предикатски симболи 39

3.4.2 Аксиоме које уводе више сведока у закључку 41

3.4.3 Имплементација и евалуација 43

3.5 Сродни системи и технике убрзавања процеса доказивања 47

3.5.1 Сродни системи . . . 48
3.5.2 Технике за убрзање рада аутоматских доказивача . . . 48
3.6 ArgoCLP Prover and its Extensions — Summary . . . 50

4 Дијалект кохерентне логике 52
4.1 Математички дијалект . . . 52
4.2 Репрезентација доказа у кохерентној логици . . . 53
4.3 Алат за подршку дијалекту кохерентне логике . . . 56
4.4 Пример генерисаних доказа . . . 60
4.5 Coherent Logic Vernacular — Summary . . . 65

5 Систем за аутоматску и интерактивну формализацију геометрије 68
5.1 Систем за аутоматску формализацију . . . 68
5.1.1 Доказивање једног тврђења . . . 69
5.1.2 Доказивање више тврђења теорије . . . 70
5.2 Систем за проверавање неформалних доказа . . . 71
5.3 Поређење са сродним системима . . . 74
5.4 Framework for Formalization of Geometry - Summary . . . 76

6 Примене система за аутоматску и интерактивну формализацију геометрије 78
6.1 Геометрија Тарског . . . 78
6.1.1 Аксиоматски систем . . . 80
6.1.2 Запис геометрије Тарског у кохерентној логици . . . 82
6.1.3 Доследност формализације . . . 87
6.1.4 Преглед постојећих рачунарских формализација књиге Тарског . . . 88
6.1.5 Упаривање Coq формализације са кохерентном формализацијом . . . 90
6.1.6 Процес аутоматизације и извођење експеримената . . . 93
6.2 Геометрија Хилберта . . . 99
6.2.1 Аксиоматски систем . . . 99
6.2.2 Примена система за проверавање неформалних доказа . . . 100
6.3 Геометрија Авигада . . . 105
6.3.1 Аксиоматски систем . . . 106
6.3.2 Примена система за проверавање неформалних доказа . . . 107
6.4 Applications of the Proving Framework — Summary . . . 113

7 Закључци и даљи рад **115**

7.1 Conclusions and Future Work — Summary 118

A Аксиоматски систем Тарског **120**

B Хилбертов аксиоматски систем **129**

C Авигадов аксиоматски систем **131**

Литература **150**

Биографија аутора **151**

1

Увод

Развој геометрије кроз векове покренуо је неколико револуција у математици. Геометрија је увек имала веома битну улогу и у математичком образовању због начина резоновања које захтева. Из ових разлога, геометрија деценијама представља изазован домен за поље рачунарског доказивања теорема, при чему је највише пажње посвећено еуклидској геометрији.

Још половином двадесетог века (а и раније) јавља се идеја креирања рачунарских система који би се могли користити приликом решавања свакодневних математичких проблема. Први приступи аутоматском доказивању теорема из области геометрије се јављају 1950-их али прави успех јавља се тек последњих година двадесетог века. Поред програма за аутоматско доказивање теорема, крајем 1960-тих година јављају се и први програми за интерактивно доказивање теорема. Основна корист ових програма је у проверавању исправности доказа креираних од стране човека, али и (у случају аутоматских доказивача теорема) у генерисању доказа техничких, али ипак нетривијалних, лема и теорема које се користе приликом развоја великих математичких теорија.

У последњих неколико деценија, велики напредак је направљен у пољу интерактивног и аутоматског доказивања теорема и данас се такви програми користе у многим пољима математике и рачунарства. Ипак, упркос великом прогресу у развоју програма за рачунарско доказивање теорема, математичари их и даље не користе у свакодневној пракси. Даљи развој доказивача теорема мотивисан је у великој мери жељом да се они приближе потребама и очекивањима математичара који би их користили као средство у свом свакодневном раду.

Део логике првог реда погодан за записивање стандардних математичких теорија (посебно геометрије) је кохерентна логика. Кохерентна логика обезбеђује

релативно лако креирање природних, интуитивних и читљивих доказа (у стилу природне дедукције и уланчавања унапред). Кохерентна логика се може разматрати и као проширење резолуцијске логике, али за разлику од доказивања у резолуцијској логици, тврђење које се доказује се не трансформише и доказује се директно (побијање, Сколемизација и транформисање у клаузалну форму се не користе). Докази у оквиру кохерентне логике се лако могу превести у језике различитих интерактивних доказивача теорема и у природни језик тако да су погодни за процес формализације геометрије. У оквиру ове тезе креиран је аутоматски доказивач теорема за кохерентну логику ArgoCLP (креиран у сарадњи са Весном Маринковић, рођеном Павловић, и Предрагом Јаничићем). Поред тога развијен је и приказан дијалект за кохерентну логику и репрезентација доказа која га описује. Дијалект за кохерентну логику је заснован на природној дедукцији. Репрезентација доказа је задата у XML формату, веома је једноставна али и довољно изражајна да се у њој могу записати разне математичке теорије. Може се користити као излазни формат за кохерентне доказиваче теорема (али и доказиваче општег типа) и доказивач ArgoCLP генерише доказе у XML формату који припада овом дијалекту. Креирана је и подршка у виду колекције XSL алата за трансформисање трага доказа из XML формата у различите језике. Тренутно постоје алати за превођење доказа у рачунарски провериве доказе записане на језицима интерактивних доказивача теорема Isabelle и Coq, као и алати за превођење доказа у доказе записане на природном језику, енглеском и српском (записане у форматима LaTeXi HTML).

Ова теза се бави формализацијом математичког знања (математичког наслеђа, уџбеника и слично) уз помоћ програма за интерактивно и аутоматско доказивање теорема. Биће приказане предности и мане тих програма, као и веза која се може успоставити међу њима. Систем за аутоматску и интерактивну формализацију који је направљен и описан у овој тези има могућност доказивања теорема и проверавања неформалних доказа теорема при чему генерише машински провериве доказе записане у форми читљивих доказа налик на доказе из математичких уџбеника. Систем се базира на кохерентној логици. Комбинује неколико различитих приступа и програма који укључују резолуцијске доказиваче теорема, доказивач теорема за кохерентну логику, интерактивне доказиваче теорема, као и скуп XSL алата за кохерентну логику (који се користи приликом превођења доказа у машински провериве доказе, као и доказе записане на природном језику). Систем ће бити коришћен за доказивање теорема из првог дела књиге о заснивању геометрије *Metamath-*

ematische Methoden in der Geometrie користећи аксиоматски систем Тарског. Биће илустрована употреба система за проверавање неформалних доказа на доказима из математичких уџбеника за први разред средње школе користећи Хилбертов аксиоматски систем. Систем креиран у овој тези ће бити употребљен и за проверавање неформалних доказа Еуклидових *„Елемената"* користећи аксиоматски систем Цереми Авигада.

Доприноси овог рада су следећи:

- Због чињенице да је кохерентна логика погодан домен за записивање аксиома и теорема из области геометрије, креиран је аутоматски доказивач теорема ArgoCLP који се базира на кохерентној логици. Доказивач може да ради са произвољним скупом аксиома и генерише доказ који се може формално проверити уз помоћ рачунара, као и читљив доказ записан на природном језику. Рад доказивача је анализиран са неколико различитих аксиоматских система и приказан на конференцији *Automated Deduction in Geometry*, а након тога и објављен у Springer-овој серији *Lecture Notes in Computer Science* [83].

- Представљене су две технике за трансформацију аксиоматских система. Коришћење модификованог аксиоматског система, у току процеса аутоматског доказивања теорема, генерише знатно мањи број тривијалних чињеница и директно утиче на убрзање доказивача. Рад који описује те технике и њихов утицај на ефикасност доказивача и на изглед добијених доказа је објављен у Springer-овој серији *Lecture Notes in Computer Science* [81] форума *Automated Deduction in Geometry*.

- Дефинисан је дијалект за кохерентну логику и репрезентација доказа која га описује. Тако креиран формат доказа може бити дељен између различитих аутоматских доказивача теорема и може се лако трансформисати у језике различитих интерактивних доказивача теорема, као и у природне језике. Подржан је у оквиру доказивача ArgoCLP и коришћен у процесу формализације различитих аксиоматских система еуклидске геометрије. Представљен је на конференцији *Intelligent Computer Mathematics - CICM* и објављен у Springer-овој серији *Lecture Notes in Computer Science* [82].

- Формализован је аксиоматски систем Тарског који је представљен у књизи *Metamathematische Methoden in der Geometrie*. Аксиоме и теореме које припадају геометрији равни, из првих дванаест поглавља те књиге, су анализиране, трансформисане у кохерентну форму и искоришћене за детаљну анализу постојеће рачунарске формализације те књиге.

- Дизајниран је систем за аутоматско доказивање теорема који комбинује моћи различитих аутоматских и интерактивних доказивача теорема. Систем се може користити за доказивање појединачне теореме али и за аутоматско доказивање целе теорије. Примењен је на теореме из првих дванаест поглавља књиге Тарског (које припадају геометрији равни) и резултати су објављени у часопису *Annals of Mathematics and Artificial Intelligence* [86].

- Креиран је систем за аутоматску верификацију неформалних доказа. Примењен је над доказима теорема и решењима задатака које се налазе у математичким уџбеницима за први разред средње школе. Његова употреба наликује интерактивном доказивању теорема са далеко једноставнијим приступом доказивању теорема него што тренутно постоји у интерактивним доказивачима. Рад на ту тему прихваћен је за објављивање у часопису *InfoM*. Аутоматска верификација неформалних доказа теорема је извршена и над доказима неколико почетних теорема Еуклидових „Елемената" и њихових последица.

1.1 Introduction — Summary

Geometry always had a very important role in mathematical education because of paradigmatic reasoning that it requires. For decades it has been a challenging domain for computer theorem proving, with most attention paid to Euclidean geometry. The idea of using computer systems to solve problems in mathematics originates back to the mid 20th century. The first useful interactive programs that assist in theorem proving were created in 1960s. Today, these programs can be used to verify human-generated proofs and also to generate original profs for technical, non-trivial lemmas used in the development of large-scale mathematical theorems. Despite the ever-increasing sophistication of these systems, mathematicians still rarely use them. Ongoing work in this field is focused on making automated theorem provers robust enough so that they can become an integral part of a daily work-flow of an average mainstream mathematician.

Two main directions in computer theorem proving, interactive theorem proving and automated theorem proving, today have similar goals. Theirs development is motivated by the need for building a corpus of verified mathematical knowledge, by the use within different applications in education (e.g., within dynamic geometry software) or in industry (when it is more important to know that a certain conjecture is valid than to have its proof) and more often motivated by the ease of use.

The main goal of this thesis is formalization of mathematical knowledge with the use of programs for automated and interactive theorem proving and coherent logic. The idea is to develop a system for automated theorem proving, that can generate formal, machine verifiable geometry proofs, if possible efficiently and in the traditional geometry manner. The system can be used for generating a proof of a theorem, as well as for verification of an informal proof of a theorem. The system generates formal, computer verifiable proofs along with a proof written in Serbian and English.

This thesis makes the following contributions:

- Given that coherent logic is a domain suitable for writing geometry axioms and theorems, ArgoCLP, an automated theorem prover based on coherent logic has been created. The prover is able to work on an arbitrary set of axioms to generate both proofs that can be formally verified using a computer as well as readable proofs written in natural language. The operation of the prover has been analysed against several different axiomatic systems and presented at the *Automated Deduction in Geometry* conference and later published in Springer's *Lecture Notes in Computer Science* series [83].

- Two techniques for transformation of axiomatic systems are presented. The use of modified axiomatic system during the process of automated theorem proving generates significantly smaller number of trivial facts and directly affects the speed increase of the prover. A paper describing these techniques and their effect on prover efficiency and the structure of the obtained proofs has been published in Springer's *Lecture Notes in Computer Science* series [81] for the *Automated Deduction in Geometry* forum.

- A dialect for coherent logic is defined along with a representation of the proof that describes it. The defined format can be shared between various automated theorem provers and can easily be transformed into languages of various interactive theorem provers as well as into natural languages. The format is supported by ArgoCLP prover and has also been used in the process of formalization of various axiomatic systems for Euclidean geometry. It was presented at the *Intelligent Computer Mathematics - CICM* conference and published in Springer's *Lecture Notes in Computer Science* series [82].

- The thesis formalizes Tarski's axiomatic system, as presented in book "Metamathematische Methoden in der Geometrie". The axioms and theorems that

refer to the geometry of plane, described in the first twelve chapters of that book, are analysed, transformed into coherent form and used for detailed analysis of the existing computer formalization of the book.

- A system for automated theorem proving, combining the power of various automated and interactive theorem provers is designed. The system can be used not only for proving a single theorem, but also for automated proving of a whole theory. It has been applied on theorems from the first twelve chapters of Tarski's book (that refer to the geometry of plane), and the results have been published in the *Annals of Mathematics and Artificial Intelligence* journal [86].

- A system for automated verification of informal proofs has been created. The system has been applied on proofs of theorems and solutions of problems found in mathematical textbooks for the first grade of high school. Its use is similar to interactive theorem proving with significantly simpler approach to theorem proving compared to the approach that's currently used by interactive theorem provers. A paper on this subject has been accepted for publication in the *InfoM* journal [85]. Automated verification of informal theorem proofs has also been performed on proofs of several opening theorems in Euclid's "Elements" and their implications.

2

Преглед основних појмова

У овој глави су представљени основни појмови коришћени у овој тези. Дат је кратак опис аксиоматских система еуклидске геометрије, преглед неколико доказивача за аутоматско и интерактивно доказивање теорема, као и опис кохерентне логике (дела логике првог реда) на коју се ослањају сви алати описани у оквиру ове тезе.

2.1 Заснивање геометрије

Еуклид ($Ευκλειδης$), са својом књигом „Елементи", сматра се зачетником употребе аксиоматских система у математици. Он је у „Елементима" [44] засновао геометрију као један аксиоматски систем са својим аксиомама и постулатима. Коришћењем логичких правила, из тих аксиома и постулата, извео је многа геометријска својства која су била позната и давно пре њега. Његов аксиоматски систем био је коришћен вековима, иако се из данашње перспективе може сматрати донекле непрецизним.

Дејвид Хилберт (David Hilbert) је 1899, у својој књизи *Der Grundlagen der Geometrie*, представио нови аксиоматски систем за еуклидску геометрију који је исправио многе мане и слабости Еуклидовог система [45]. Овај аксиоматски систем представља један од најзнаменитијих резултата математике у XX веку. Хилбертов аксиоматски систем ипак има недоречености које отежавају верну рачунарску формулацију.[1] Хилбертов аксиоматски систем користи три врсте

[1] На пример, у ранијим верзијама аксиоматског система Хилберт је експлицитно наводио услове различитости тачака у аксиомама, у каснијим верзијама ти услови су избачени али је остала општа напомена за сва тврђења да помињање „двеју, трију,..." тачака подразумева различите тачке. Испоставља се да ова напомена често уноси забуну приликом тумачења Хилбертових аксиома и поставља се питање да ли је различитост објеката у појединим аксиомама неопходна или не.

објеката: тачке, праве и равни. Скуп аксиома подељен је у пет група (аксиоме припадања, аксиоме редоследа, аксиоме подударности, аксиома паралелности и аксиома непрекидности). Свака група аксиома пропраћена је основним теоремама које се могу доказати употребом претходних аксиома. Модернија верзија Хилбертовог аксиоматског система су представили Карол Борсук (Karol Borsuk) и Ванда Шмилев (Wanda Szmielev) [14].

Средином двадесетог века, Тарски је изградио нови аксиоматски систем за еуклидску геометрију (дао је неколико варијанти тог аксиоматског система) записан у логици првог реда. Осим аксиоматског система (са својствима непрекидности која су слабија у односу на Хилбертов аксиоматски систем), Тарски је представио и процедуру одлучивања за ту теорију [91, 79]. Тај аксиоматски систем је прилично једноставан: користи само једну врсту објеката — тачке, користи само два примитивна предиката (*cong* арности 4 и *bet* арности 3, који представљају релацију подударно и релацију између) и само једанаест аксиома.

Један од новијих аксиоматских система за Еуклидску геометрију, систем E, осмислили су Џереми Авигад, Едвард Дин (Edward Dean) и Џон Мума (John Mumma) [3]. Аксиоматски систем E је креиран тако да прецизно покрива основне идеје и методе закључивања које су коришћене у Еуклидовим „Елементима". Заснива се, као и Еуклидов аксиоматски систем, на коришћењу илустрација али на ригорозан, формални начин. Развијен је након великог броја примедби упућених коришћењу илустрација у Еуклидовим доказима и запажања да се извођења у доказима више служе интуицијом него строгом применом унапред задатих правила[2].

2.2 Аутоматско и интерактивно доказивање теорема

Доказивање теорема уз помоћ рачунара данас има два основна правца: аутоматско доказивање теорема и интерактивно доказивање теорема. Основна разлика између ова два правца се огледа у количини помоћи које крајњи корисник добија од рачунара као и у информацијама које рачунар на крају процеса доказивања пружа. У случају аутоматског доказивања теорема корисник се у потпуности ослања на рачунар и на крају процеса доказивања може само да има увид у добијени излаз (који у неким случајевима може садржати сам доказ тврђења, а у неким само информацију да ли је дато

[2] http://www.phil.cmu.edu/~avigad/formal/paris2_merged.pdf

тврђење теорема или не). У интерактивном доказивању теорема корисник се првенствено ослања на сопствене идеје и има много већу слободу у току самог процеса доказивања. Корисник доказује теорему уз помоћ рачунара који првенствено служи томе да провери да ли је корисников доказ формално исправан, при чему корисник има приступ и одређеном нивоу аутоматизације који може користити у случајевима када је аутоматизација погодна.

2.2.1 Аутоматско доказивање теорема

Аутоматски доказивачи теорема деле се на доказиваче опште намене (на пример, доказивачи засновани на методу резолуције [76]), и на доказиваче намењене конкретним теоријама првог реда. Обе врсте аутоматских доказивача су достигле висок ниво успешности и могу се ефикасно користити за доказивање (или оповргавање) веома тешких или обимних тврђења (која могу да имају стотине или хиљаде променљивих) која се могу добити из различитих области математике или индустрије.

Иако се најчешће користе за проверавање да ли је одређено тврђење теорема, аутоматски доказивачи теорема могу бити коришћени и за откривање скупа аксиома и лема које се користе у процесу доказивања одређеног тврђења. Осим тога, у случају да аутоматски доказивач није у стању да докаже да је неко тврђење теорема, човек као крајњи корисник, познајући детаље проблема који се решава, може обезбедити помоћ аутоматском доказивачу теорема у виду формулисања додатних лема за које се може показати да су од кључног значаја за процес доказивања одређеног тврђења.

2.2.2 Интерактивно доказивање теорема

Са великим бројем грешака у доказима теорема у уџбеницима и у објављеним радовима, како у области математике тако и у области рачунарства, машински провериви докази (формални докази записани на објектном нивоу — у терминима аксиома и правила извођења) добијају све већи и већи значај. Интерактивни доказивачи теорема су програми који се користе за конструисање и проверавање формалних доказа од стране човека (у складу са логиком на којој се базирају). У потпуности су поуздани захваљујући томе што су засновани на малом скупу правила која се могу ручно проверити. Међу најпопуларнијим интерактивним доказивачима теорема налазе се Isabelle[3] [68],

[3] http://www.cl.cam.ac.uk/research/hvg/Isabelle/

Coq[4] [93], Mizar[5] [94], и HOL-light[6] [43]. Многа комплексна тврђења везана за верификацију софтвера и неке тешке математичке теореме су доказане уз помоћ интерактивних доказивача теорема [42, 39].

У последњих неколико година популарност интерактивних доказивача расте, а формални докази добијају централну улогу у чувању математичког знања (на пример приликом дигитализације математичког наслеђа), у образовању, али и у индустрији у случајевима када је коректност неког алгоритма или израчунавања од пресудне важности. Постоји све веће интересовање за креирање колекција доказаних теорема и формирање корпуса формализованог математичког знања.

Традиционални математички докази, када се разматрају кроз сферу машински проверивих доказа, тешко се могу назвати доказима већ више подсећају на скицу доказа. У таквим доказима често има прескочених делова доказа, недовољно формалних и прецизних аргумената и слично.

Писање машински проверивих доказа се највећим делом изводи ручно (са ограниченом аутоматизацијом), али се докази проверавају уз помоћ рачунара. Отуда је интерактивно доказивање теорема често врло захтеван и дуготрајан процес, што зависи од искуства корисника. Осим тога, постојеће доказе није лако поново искористити. Чак и у случају мањих промена у оквиру теорије у којој се доказује тврђење, с обзиром на органичен ниво аутоматизације, доказ једне исте теореме (у промењеној теорији) неће обавезно бити идентичан и често може бити потребно расписивање целог доказа од почетка. Због тога, и поред све већег броја успешних резултата, употреба интерактивних доказивача теорема још увек није стандардна пракса у процесу доказивања теорема у свим областима математике. Стога се све више пажње посвећује томе да интерактивни доказивачи постану пријатнији и лакши за коришћење из перспективе крајњег корисника (чак и почетника).

Када се пореде са традиционалним доказима, формални докази су у већини случајева много дужи од традиционалних доказа[7]. Напредак који се јавља у интерактивним доказивачима теорема може се пратити кроз чињеницу да формални докази постају све краћи, али да и даље садрже довољну количину информација да би систем био у стању да конструише и верификује комплетан

[4] http://coq.inria.fr/

[5] http://www.mizar.org

[6] http://www.cl.cam.ac.uk/~jrh13/hol-light/

[7] Однос у дужини формалног доказа и неформалног доказа се назива *de Bruijn* фактор [4]. Разликује се за различите делове математике и за различите доказиваче, а често је око 4. У случајевима када се интензивно користи аутоматизација, тај фактор може бити чак и испод 1. Такође може бити и већи од 10, у случајевима када је неформални доказ недовољно прецизан.

формални доказ.

Докази у интерактивним доказивачима теорема могу бити написани на три начина: коришћењем декларативног стила, процедуралног стила или коришћењем аутоматског навођења (*the guided automated style*) [99]. Декларативни стил доказа подразумева навођење непосредних последица у сваком кораку доказа коришћењем језика који подсећа на стандардан текст који се јавља у математичким уџбеницима. Процедурални стил доказа одликује интензивна употреба аутоматизације и докази који се састоје од низа команди које мењају дрво доказа. Доказивање аутоматским навођењем подразумева навођење низа тврђења које доказивач покушава да докаже (у неким случајевима се могу проследити додатне информације доказивачу да би се олакшало доказивање одређеног тврђења).

Декларативни и процедурални стил доказа се могу користити у доказивачима HOL-Light, Isabelle [96] и Coq, док Mizar [99] обезбеђује само декларативни стил писања доказа. Процедурални стил писања доказа се чешће користи у Coq заједници. Доказивање аутоматским навођењем се може користити за доказивање теорије формулисане преко низа тврђења – лема, при чему се текућа лема доказује уз помоћ скупа претходних лема. Користи се у доказивачима ACL2 [52] и Theorema [20] и од свих приступа највише има сличности са приступом описаним у овој тези.

2.2.3 Интеграција интерактивног и аутоматског доказивања теорема

Интеграција интерактивних и аутоматских доказивача теорема може бити корисна како приликом интерактивног, тако и приликом аутоматског доказивања теорема. Са једне стране аутоматски доказивачи теорема се не могу сматрати апсолутно поузданим из разлога што може постојати грешка у имплементацији доказивача. Један начин да се овај проблем превазиђе је да се аутоматски доказивач користи заједно са интерактивним доказивачем теорема који ће верификовати излаз који аутоматски доказивач генерише. Са друге стране, интерактивни доказивачи теорема раде са теоријом коју корисник дефинише. Може се десити да корисник направи грешку у куцању или да погрешно формулише нека тврђења теорије. У таквим случајевима аутоматски доказивачи теорема могу бити коришћени за откривање неконзистентности аксиоматског система који се користи у процесу доказивања. Из тих разлога последњих година се интензивно ради на комбиновању алата за аутоматско и интерактивно доказивање теорема. Данашњи интерактивни доказивачи

теорема имају могућност коришћења SAT доказивача, SMT доказивача, метода резолуције и слично.

Комбиновање тих алата отвара нова поља примене аутоматског доказивања теорема како у верификацији софтвера и хардвера тако и у формализацији математичког знања и у свакодневној примени математике.

Интерактивни доказивач теорема Isabelle располаже одређеним алгоритмима за аутоматско доказивање теорема међу којима се налазе доказивач који се заснива на класичном резоновању, симплификатор који врши једнакосно резоновање, специјализоване процедуре одлучивања за линеарну аритметику, итд.

Најуспешнији пример комбиновања аутоматског доказивања теорема и интерактивног доказивања теорема је систем *Sledgehammer* [11, 12]. Sledgehammer је систем у оквиру интерактивног доказивача теорема Isabelle који укључује употребу екстерних аутоматских доказивача теорема базираних на резолуцији као што су *Vampire* [75], *E* [78] и *SPASS* [95] и употребу локалног (резолуцијског) доказивача теорема *Metis* (који је, када се користи самостално, знатно слабији од екстерних доказивача). У случају да је један од екстерних доказивача успешно доказао теорему, идентификује се листа аксиома (и лема) коришћених у доказу и та листа се прослеђује доказивачу Metis који најчешће (мада не увек) успева да реконструише верификовани доказ за дату теорему.

2.3 Доказивање теорема у геометрији уз помоћ рачунара

У овој тези највише пажње ће бити усмерено ка еуклидској геометрији и аксиоматским системима који се у оквиру ње користе. Најпознатији аксиоматски системи за еуклидску геометрију су Еуклидов, Хилбертов и аксиоматски систем Тарског. Формализовање тих система на рачунару често захтева њихову веома детаљну анализу [72, 60, 65]. Прављење нових аксиоматских система је и даље актуелна област претежно захваљујући програмима за интерактивно доказивање теорема и формализацији математике која се са њима појавила.

Аутоматско доказивање теорема у геометрији актуелно је више од педесет година [25]. Херберт Гелертнер (Herbert Gelernter) је 1959-те креирао доказивач геометријских теорема који је био у стању да докаже велики број проблема геометрије равни из средњошколских уџбеника [37]. Највећи успеси у аутоматском доказивању теорема у геометрији су постигнути уз помоћ аутоматских

доказивача теорема који се заснивају на алгебарским методама као што су *Вуов метод* [100, 24] и *метод Гребнерових база* [19, 51, 33]. Мана ових метода је што као излаз дају само да/не одговор пропраћен алгебарским аргументима. На тај начин се не могу добити традиционални читљиви геометријски докази. У одређеној мери је овај проблем решен појавом метода "слободних координата", као што је метод површина [26] и метод потпуних углова [27, 23], али не у потпуности. Арт Куаиф (Art Quaife) је радио на генерисању аутоматских доказа за геометрију Тарског применом резолуцијског доказивача [74]. Успео је да докаже нека комплексна тврђења али се није бавио генерисањем формалних нити читљивих доказа.

Формализација геометрије, уз помоћ интерактивних доказивача теорема, је област на којој је интензивно рађено последњих година. Кристоф Делинџер (Christophe Dehlinger), Жан-Франсоа Дуфо (Jean-Francois Dufourd) и Паскал Шрек (Pascal Schreck) су формализовали прве две групе Хилбертове књиге *Grundlagen der Geometrie* у оквиру алата за интерактивно доказивање теорема Coq. Користили су интуиционистички приступ [30] и приметили да се многе Хилбертове теореме не могу доказати без правила искључења трећег. Испоставља се да је резоновање по случајевима неопходно за доказивање Хилбертових теорема па су аутори прво формулисали и доказали слабије варијанте теорема које су онда искористили у доказу оригиналних теорема. Докази слабијих варијанти теорема су потпуно интуиционистички, док докази оригиналних теорема као први корак користе правила искључења трећег (која су додата у скуп аксиома за појединачне предикате и не користе се у општем облику). Лаура Мејкл (Laura Meikle) и Жак Флорио (Jacques Fleuriot) [60] су формализовали прве три групе Хилбертових аксиома у алату за интерактивно доказивање теорема Isabelle/Isar. Показали су да се докази Хилбертових теорема често ослањају на неке имплицитне претпоставке (које нису наглашене у доказу већ се базирају на илустрацији којом је представљен проблем) и показали су неопходност постојања формално проверивих доказа. Жилиен Нарбу (Julien Narboux) [66] је формализовао, у оквиру алата за интерактивно доказивање теорема Coq, иницијално првих осам поглавља књиге Тарског [79], да би касније у сарадњи са осталим колегама формализовао и остатак те књиге [15]. Та формализација је показала да је геометрија Тарског погодна за формализацију због своје једноставности и чињенице да захтева разматрање мањег броја дегенерисаних случајева.

Осим ових формализација, у Coq-у су развијене и наредне формализације: Жил Канова (Gilles Kahn) формализација Жан Вон Платове (Jan von Plato)

конструктивне геометрије [71, 47], Фредерик Гиљоова (Frederique Guilhot) формализација средњошколске геометрије [40], Жан Дупраова (Jean Duprat) формализација аксиоматског система за геометрију компаса и шестара [32], формализација пројективне геометрије коју су направили Николас Магауд (Nicolas Magaud), Нарбу и Шрек [55, 56], итд. Све поменуте формализације су урађене ручно, практично без икакве спољне аутоматизације.

2.4 Кохерентна логика и аутоматско доказивање у кохерентној логици

У овом поглављу ће бити описана кохерентна логика и процес доказивања у оквиру кохерентне логике. Више различитих аутора независно је издвојило кохерентну логику (или сличне делове логике првог реда) као логику погодну за изражавање значајних делова стандардне математике (посебно геометрије). На пример, Џереми Авигад (Jeremy Avigad) и коаутори, приликом креирања новог аксиоматског система за Еуклидску геометрију [3], Мохан Гансалингам (Mohan Ganesalingam) и Вилијам Тимоти Гауерс (William Timothy Gowers) приликом аутоматског креирања читљивих доказа [35], и Стивен Гивант (Steven Givant) и Алфред Тарски (Alfred Tarski) у контексту развијања нове аксиоматизације за геометрију [92, 79].

2.4.1 Језик кохерентне логике и извођење доказа у кохерентној логици

Кохерентна логика је део логике првог реда погодан за аутоматско доказивање теорема и једноставно генерисање читљивих доказа. Иницијално ју је дефинисао Торалф Сколем (Thoralf Skolem), а у последње време кохерентна логика је интензивно коришћена од стране различитих аутора [8, 34, 9].

Сигнатуру кохерентне логике чине предикатски симболи и функцијски симболи арности нула, које ћемо звати константама. Не постоје функцијски симболи арности веће од нула. У изградњи формула користе се бинарни логички везници $\wedge$, $\vee$, $\Rightarrow$, нуларни везник тј. логичка константа $\bot$ и квантификатори, универзални $\forall$ и егзистенцијални $\exists$. Терм је константа или променљива. Атомичка формула је $\bot$ или $p(t_1, \ldots, t_n)$, при чему је p предикатски симбол арности n и t_i $(1 \leq i \leq n)$ су терми. Чињеница је атомичка формула у којој се не појављују променљиве. Не користи се логички везник $\neg$. Негација атомичких формула симулира се увођењем додатних предикатских симбола. За сваки

предикатски симбол R, уводи се нови симбол $\overline{R}$ који ће се користити уместо $\neg R$, и уводе се следеће две аксиоме [73]: $\forall\vec{x}(R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot)$, $\forall\vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$. Дефиниција кохерентне логике је преузета из рада Марка Безема [9]. Формуле кохерентне логике су имплицитно универзално квантификоване формуле следећег облика:

$$A_1 \wedge \ldots \wedge A_n \Rightarrow \exists\vec{x_1}B_1 \vee \ldots \vee \exists\vec{x_m}B_m \tag{2.1}$$

где је $0 \leq n$, $0 \leq m$, и $\vec{x}$ означава низ променљивих $x_1, x_2, \ldots, x_k$ $(0 \leq k)$, A_i (за $1 \leq i \leq n$) означава атомичке формуле и B_j (за $1 \leq j \leq m$) означава конјункцију атомичких формула .

Током доказивања неке теореме сигнатура теорије којој припада се мења додавањем нових симбола константи — сведока. Под *константом* ћемо сматрати и константе које су део сигнатуре и сведоке.

У кохерентној теорији T, скуп $\Delta^T(X \vdash F)$ извођења чињенице F из скупа чињеница X се дефинише индуктивно користећи наредна два правила [9]:

$$\frac{X}{F}\ F \in X \qquad \frac{X \quad \mathbf{A} \Rightarrow \mathbf{D} \quad \delta_1 \ldots \delta_m}{F}\ \mathbf{A} \subseteq X$$

Базни корак се примењује када се циљ F налази у скупу чињеница X. Индуктивни корак се примењује када је $\mathbf{A} \Rightarrow \mathbf{D}$ затворена инстанца формула из T чије премисе су задовољене скупом чињеница X, што је означено са $\mathbf{A} \subseteq X$. Број подизвођења δ_i је једнак броју дисјунката дисјункције $\mathbf{D}$. Ако је $\mathbf{D} = \exists\vec{x_1}B_1 \vee \ldots \vee \exists\vec{x_m}B_m$, свако подизвођење δ_i је део $\Delta^T(X, \overline{B_i} \vdash F)$ $(1 \leq i \leq m)$. У овој формули $X, \overline{B_i}$ означава скуп чињеница X проширен атомима из B_i у којима су променљиве $\vec{x_i}$ замењене свежим константама. Ако је $\mathbf{D}$ једнако $\bot$, тада нема подизвођења.

У кохерентној логици се обично не користе типови и уместо њих уводе се додатни предикатски симболи који ће се користити за представљање типова објеката. На пример, у случају геометрије биће уведени унарни предикатски симболи *point*, *line* и *plane* за представљање објеката који припадају типовима тачка, права и раван. Тако ће формула тачке A и B су различите бити записана на следећи начин: $point(A) \wedge point(B) \wedge A \neq B$.

Сваки доказ у оквиру кохерентне логике који може бити изведен наведеним правилима, може бити изведен у оквиру интуиционистичке логике. Разне теорије и теореме могу бити директно изражене у оквиру кохерентне логике. Може се доказати да се свака формула првог реда може превести у скуп

формула кохерентне логике (са различитом сигнатуром од почетне) исте задовољивости [73] (међутим, ово превођење се ослања на кораке који укључују класичну логику). Кохерентна логика је полуодлучива и постоји неколико већ имплементираних процедура полуодлучивања за њу [8]. Мана доказивача за кохерентну логику је њихова ефикасност тако да најчешће нису погодни за доказивање тешких математичких теорема.

Разне геометријске теорије природно се могу формулисати у оквиру кохерентне логике. За њих постоји процедура заснована на претрази у ширину која је сагласна и потпуна: кохерентна формула F је доказива ако и само ако је F тачна у сваком моделу Тарског (са непразним доменом) скупа аксиома и чињеницама $A_1(\vec{a}), \ldots, A_n(\vec{a})$ [8].

2.4.2 Читљиви докази

Читљивост доказа је битан аспект математике, како у неформалном облику, тако и приликом формализације математике. Иако се формализација претежно бави питањима исправности доказа (која је гарантована интерактивним доказивачима теорема) и питањима аутоматизације процеса доказивања (која се добија интеракцијом са аутоматским доказивачима теорема), читљивост доказа је аспект на коме се последњих година све више ради. Постојање читљивих доказа неће бити једнако битно у свим пољима (на пример приликом верификације софтвера) али они могу бити веома корисни и у току самог процеса доказивања. Такође, математичарима често није циљ добијање само *да/не* одговора на питање да ли је неко тврђење теорема већ им је циљ добијање јасног и интуитивног доказа те теореме.

Кохерентна логика допушта коришћење егзистенцијалних квантификатора у закључку формуле, па се може сматрати проширењем (надградњом) резолуцијске логике. За разлику од доказивања методом резолуције, тврђење које се доказује се не мења и директно се доказује (побијање, Сколемизација и превођење у клаузалну форму се не користи). Отуда су докази у оквиру кохерентне логике природни и интуитивни, резоновање је интуиционистичко, а читљиви и формални докази (у стилу уланчавања унапред и природне дедукције) се једноставно добијају. Генерисање читљивих доказа у оквиру кохерентне логике ће бити детаљно описани у поглављу 4.2.

Доказивач који су направили Гансалингам и Гауерс [35] генерише формат излаза доказа који је читљив за човека. Правила извођења предложена у њиховом раду веома личе на правила коришћена у систему за доказивање

који се заснива на кохерентној логици (који ће бити описан у наставку тезе).[8] Иако њихов систем ради са логиком првог реда са идејом да укључе и нека својства логике другог реда (што би такође било могуће урадити и за кохерентну логику), на основу доприноса њиховог рада делује да њихов досадашњи рад такође припада фрагменту кохерентне логике, с обзиром на то да су докази коришћењем контрапозиције и контрадикције планирани за будућност.

Када је реч о добијању читљивих доказа, алтернатива коришћењу доказивача за кохерентну логику (или сличних доказивача прилагођених читљивим доказима) би била коришћење моћнијих аутоматских доказивача и реконструкција и рефакторисање доказа [80, 10, 50]. Међутим, процес реконструисања доказа на основу јако ефикасног и оптимизованог доказа аутоматског доказивача је веома тежак. Један од проблематичних корака би била десколемизација, тј. реконструисање доказа насталог из сколемизоване верзије почетног проблема. У доказивању базираном на методу резолуције, трансформисање улазне формуле у конјунктивну нормалну форму и сколемизација је први корак у процесу доказивања. Иако резолуцијски доказивачи могу доказати много већи скуп теорема, докази које они дају могу бити веома нечитљиви. Међутим, у циљу добијања читљивих доказа, аутоматски доказивачи теорема могу бити корисни у виду предпроцесора када се користе као помоћ доказивачу за кохерентну логику.

Важност постојања читљивих доказа се види и кроз измене настале у оквиру језика за интерактивно доказивање теорема. Већина интерактивних доказивача теорема користе скрипте које експлицитно наводе списак свих аксиома и правила извођења која се користе у сваком кораку доказивања што знатно олакшава праћење самог тока доказа. У оквиру доказивача Isabelle развијен је Isar (*The Intelligible semiautomated reasoning*) [96], декларативни језик који представља алтернативу традиционалним скриптовима и тактикама и обезбеђује писање доказа који су разумљиви и погодни за читање како рачунарима тако и људима. Његова појава знатно олакшава употребу интерактивног доказивача теорема и смањује јаз који постоји између интерних појмова доказа који се користе од стране интерактивног доказивача теорема и одговарајућег нивоа апстракције који је потребан за удобан рад крајњег корисника.

[8] На пример, њихово правило *splitDisjunctiveHypothesis* одговара нашем правилу *case split*, правило *deleteDoneDisjunct* одговара нашем правилу *as*, правило *removeTarget* одговара *as* (са дужином $\vec{\mathbf{y}}$ већом од 0), *forwardsReasoning* одговара правилу *mp*. Видети поглавље 4.2.

2.5 Background — Summary

In this section, different axiomatic systems for Euclidean geometry are presented. Starting with well-known systems given by Euclid, Hilbert and Tarski and going through to the modern axiomatic system developed by Jeremy Avigad. Euclid was the first to use axiomatic methods in mathematics in his book "Elements" [44]. Later, Hilbert introduced axiom system for elementary geometry that fixed many flaws and weaknesses of Euclid's system [45] in his seminal book "Der Grundlagen der Geometrie". In mid-twentieth century, Tarski presented a new, first-order axiomatization for elementary geometry along with a decision procedure for that theory [91, 79]. Development of different axiomatic systems is still an ongoing process as Jeremy Avigad, Edward Dean, and John Mumma presented in 2009 a new axiomatic system E for Euclid's Elements [3]. All these axiomatic systems are used within this thesis.

An overview of interactive theorem provers and automated theorem provers and their use in geometry is given. Interactive theorem provers (or proof assistants) are used for proving mathematical theorems in a rigorous and machine verifiable way. They have wide range of application, starting from proving extremely complex conjectures about software correctness [42, 39] to different projects for formalization of mathematics. The most popular theorem proving assistants nowadays are Isabelle [68], Coq [93], and Mizar [99]. Automated theorem provers have reached high levels of maturity and can be used to prove (or disprove) extremely difficult as well as huge conjectures (involving hundreds of thousands of variables) coming from various areas of mathematics and industrial applications. Proof arguments of automated theorem provers are not fully trusted and they are, in the last few years, more frequently used in a conjunction with an interactive theorem prover (which verifies the output of automated theorem prover). In this conjunction automated theorem provers can be used to handle technical conjectures and also to reveal what axioms and lemmas are sufficient for proving a theorem.

Coherent logic is the underlying logic behind all contributions of this thesis. Coherent logic is essentially the modified Horn fragment which allows having a whole disjunction of existentially quantified conjunctions of atoms within the conclusion. Coherent logic can be considered as an extension of resolution logic, but in contrast to the resolution-based proving, the conjecture being proved is left unchanged and is proved directly. Thus it is suitable for expressing many mathematical theories, especially geometry[9], while allowing for the construction of natural, intuitive, hu-

[9]It is used, for instance, by Avigad et.al. in the context of a new axiomatic foundations of Euclidean geometry [3], by Ganesalingam and Gowers in the context of importance of automated

man readable proofs (in the style of forward reasoning and as a variant of natural deduction) [8].

Formulas of the coherent logic are implicitly universally quantified formulas of the following form:

$$A_1 \wedge \ldots \wedge A_n \Rightarrow \exists \vec{x_1} B_1 \vee \ldots \vee \exists \vec{x_m} B_m \tag{2.2}$$

where $0 \leq n$, $0 \leq m$, $\vec{x}$ denotes a sequence of variables $x_1, x_2, \ldots, x_k$ $(0 \leq k)$, A_i $(1 \leq i \leq n)$ denotes an atomic formula, and B_j (за $1 \leq j \leq m$) denotes conjunction of atomic formulae.

In projects for formalizing mathematical knowledge, apart from the issue of trusted proofs (guaranteed by proof assistants) and the issue of automation (provided by automated theorem provers), there is also an issue of readable proofs. Having readable proofs is often not important in fields such as software verification, but they are essential in everyday mathematical practice. For mathematicians and in education, the main goal is often a clear and intuitive proof of a theorem, that serves not only as a justification, but more importantly as an explanation.

generation of readable proofs [35], by Tarski in the context of geometry [79], etc.

3

Доказивач ArgoCLP и проширења

Интерактивно и аутоматско доказивање теорема имају различите примене које усмеравају њихов развој. Интерактивно доказивање теорема се првенствено користи приликом прављења корпуса верификованог математичког знања, док аутоматско доказивање теорема има разне примене у образовању и индустрији (за генерисање доказа у оквиру динамичких софтвера за геометрију, или у случајевима када је потребно проверити да ли је неко тврђење теорема, без постојања читљивог доказа). Иако су ти начини коришћења доста различити постоје области у којима би било корисно имати и једно и друго и јавља се идеја креирања ефикасног система који би аутоматски генерисао формалне, машински провериве доказе налик традиционалним доказима који се могу наћи у уџбеницима.

Један од доприноса ове тезе је креирање аутоматског доказивача базираног на кохерентној логици — *ArgoCLP (Automated Reasoning GrOup Coherent Logic Prover*) који генерише и традиционалне (читљиве) доказе и формалне (машински провериве) доказе [83]. Генерисани докази (који се састоје од детаљних корака извођења) веома личе на доказе који се налазе у стандардним уџбеницима.

Доказивач је првенствено намењен за доказивање у геометрији (у оквиру различитих аксиоматских система), али није ограничен на домен геометрије. Теореме које овај доказивач може да докаже су најчешће релативно једноставне теореме или теореме које су настале као модификација одређених аксиома (и у том случају би се доказивач могао користити да оправда увођење таквих модификација). Осим тога, доказивач може да служи као асистент у процесу доказивања, тако што би се користио за доказивање тежих теорема након њиховог разбијања на неколико једноставнијих, помоћних лема.

3.1 Опис доказивача ArgoCLP

Доказивач ArgoCLP је генерички[1] аутоматски доказивач теорема који се базира на кохерентној логици и може радити са произвољним скупом кохерентних аксиома. Његов алгоритам се заснива на једноставној процедури уланчавања унапред и користи итеративно повећање простора претраге (*forward chaining and iterative deepening*). Улаз за доказивач се може задати у TPTP формату[2] (или у формату који је специјално креиран за запис формула у доказивачу ArgoCLP) и даје излаз у одговарајућем XML формату (као и директно добијене доказе записане на језику интерактивног доказивача теорема Isabelle и доказе записане на енглеском језику). Излазни XML формат се даље преводи коришћењем XSLT алата, и може бити преведен у доказе записане у језику доказивача за интерактивно доказивање теорема Isabelle/Isar и Coq и у доказе записане на природном језику[3] (Енглеском) форматиране у LaTeX-у или форматиране у HTML-у [82].

Једна од теорема из књиге Тарског и њен доказ записан на енглеском језику су дати у наставку текста:

[1] Скуп аксиома који се користи у оквиру доказивача није чврсто уграђен у сам доказивач већ се уноси кроз засебне датотеке.

[2] http://www.cs.miami.edu/~tptp/

[3] Докази на природном језику су записани користећи мали скуп реченица фиксиране конструкције.

Teorema 3.1 (th_4_19) *Assuming that* $bet(A, B, C)$ *and* $AB \cong AD$ *and* $CB \cong CD$ *it holds that* $B = D$.

Proof:

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

3.1.1 Основна процедура претраживања

Основна процедура претраживања је једноставна процедура уланчавања унапред са итеративним повећавањем простора претраге. Корисник задаје доказивачу теорију са којом ради: језик теорије, њену сигнатуру, аксиоме теорије и тврђење које се доказује. Додатне дефиниције које корисник жели да користи се формулишу и користе у форми додатних аксиома.

За све предикатске симболе, који се јављају у сигнатури теорије, генеришу се гранајуће аксиоме облика $R(\vec{x}) \vee \overline{R}(\vec{x})$. Скуп предикатских симбола се одређује у зависности од тога који формат се користи за задавање теорије. У случају да је теорија (скуп дефиниција, аксиома и тврђење које се доказује) задата у TPTP формату, скуп предиката се задаје имплицитно кроз скуп тврђења који се наводи. У случају да се теорија задаје кроз неколико различитих датотека постојаће посебна датотека која ће садржати скуп предикатских симбола теорије.

Аксиоме се примењују према правилу извођења кохерентне логике, описаном у поглављу 2.4.1. Редослед којим се примењују аксиоме се може описати принципом водопада — доказивач покушава да примени једну од аксиома редоследом којим су оне задате у улазној датотеци. Када се једна од аксиома успешно примени, претрага за следећом аксиомом почиње од почетка, од прве аксиоме.

Све константе уведене премисама тврђења које се доказује или применом аксиома се нумеришу у току извршавања програма. У оквиру доказивача се користи вредност s специјално намењена контролисању примене аксиома. Аксиома се примењује само ако су све њене (универзално квантификоване) променљиве упарене са константама чији је редни број мањи од s (овом приликом се врши инстанцирање променљивих на све могуће начине до вредности s). На почетку процеса доказивања, s је једнако броју константи које се појављују у премиси тврђења које се доказује. Његова вредност се повећава само у случају да ни једна аксиома не може бити примењена над тренутним скупом дозвољених константи (дозвољене константе су оне чији је редни број мањи од s).

У случају да се закључци тврђења које се доказује могу инстанцирати тако да инстанциране чињенице припадају текућем скупу чињеница, кажемо да је тврђење доказано. У случају да се у доказу теореме појавило гранање, кажемо да је теорема доказана ако је свака грана затворена или закључком теореме или контрадикцијом. У случајевима када тврђење које се доказује није теорема може се десити да се процес доказивања никад не заврши или да доказивач не може да примени више ни једну аксиому на текући скуп чињеница.

Може се показати да је ова процедура претраге сагласна и потпуна у односу на скуп правила представљен у поглављу 2.4.1, тј. формула F је доказива ако и само ако је F теорема кохерентне логике. И поред постојања својства потпуности, доказивање комплексних тврђења одређених теорија је најчешће немогуће са основном процедуром претраге (тачније немогуће кад се узму у обзир просторна и временска ограничења).

3.1.2 Напредна процедура претраживања

Основна процедура претраживања није ефикасна. Један од разлога је потенцијално велики број сведока који се уводе приликом примене нових аксиома, а други је генерисање великог броја ирелевантних чињеница. Чињеница се сматра ирелевантном ако се њеним елиминисањем не нарушава доказ тврђења. Ефикасност основне процедуре претраживања се може поправити без угрожавања својства потпуности. У овом поглављу ће бити наведено неколико начина који су првенствено усмерени на смањење простора претраге (тј. на број сведока уведених у току процедуре доказивања) и контролисање броја уведених чињеница (који може бити експоненцијално велики у односу на број почетних чињеница).

Редослед аксиома. Један од начина смањења броја ирелевантних чињеница јесте увођење мањег броја сведока. Како се сведоци уводе продуктивним аксиомама, њихова што касније примена у току процеса доказивања може утицати на ефикасност доказивача. У основној процедури претраживања продуктивне аксиоме се примењују на исти начин као и остале аксиоме али се показује да је боље форсирати њихову што каснију примену (након исцрпне примене свих непродуктивних аксиома).

Аксиоме које се задају доказивачу су формуле кохерентне логике и припадају једној од наредних група. Претпоставља се да је $n \geq 0$ (осим за трећу и четврту групу где се претпоставља да је $n > 0$), $m > 1$, $\vec{x}$ означава низ променљивих $x_1, x_2, \ldots, x_k$ $(0 \leq k)$, A_i (за $1 \leq i \leq n$) означава атомичке формуле (у којима се појављује нула или више променљивих из $\vec{x}$), $\vec{y}$ означава низ променљивих $y_1, y_2, \ldots, y_l$ $(0 \leq l)$, и B_j (за $1 \leq j \leq m$) означава конјункцију атомичких формула (у којима се појављује нула или више променљивих из $\vec{x}$ и $\vec{y}$).

непродуктивне негранајуће аксиоме: $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B(\vec{x})$

непродуктивне гранајуће аксиоме: $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B_1(\vec{x}) \vee \ldots \vee B_m(\vec{x})$

продуктивне негранајуће аксиоме: $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}\ B(\vec{x}, \vec{y})$

продуктивне гранајуће аксиоме: $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1\ B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{x}, \vec{y}_m)$

јако продуктивне негранајуће аксиоме: $\exists \vec{y}\ B(\vec{y})$

јако продуктивне гранајуће аксиоме: $\exists \vec{y}_1\ B_1(\vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{y}_m)$

На основу самог записа аксиоме може се јединствено одредити њен тип, односно може се одредити којој групи аксиома припада. Свака група аксиома има свој приоритет који одговара редоследу којим су наведене групе аксиома у претходној листи. Унутар сваке групе редослед аксиома неће бити мењан у односу на редослед који је иницијално задат (иако промена редоследа аксиома унутар групе може утицати на ефикасност доказивача).

Рано одсецање. Приликом провере да ли је нека аксиома применљива, није увек неопходно инстанцирати све њене променљиве пре провере да ли су све чињенице из закључка већ изведене. Уместо тога провера постојања релативних чињеница може се померити најраније могуће, са циљем одбацивања одређених инстанцирања аксиома и убрзања процеса претраге. Релевантне чињенице, у оквиру примене аксиоме, ће бити провераване чим су сви аргументи тих чињеница инстанцирани.

На пример, приликом примене наредне аксиоме [4]:

$point(A) \wedge point(B) \wedge line(L1) \wedge line(L2) \wedge inc\_po\_l(A, L1) \wedge A \neq B \wedge inc\_po\_l(B, L1) \wedge inc\_po\_l(A, L2) \wedge inc\_po\_l(B, L2) \Rightarrow L1 = L2$

уместо провере да ли је аксиома применљива након инстанцирања свих променљивих A, B, $L1$ и $L2$ са целокупним скупом дозвољених константи, провера применљивости ове аксиоме се може извршити чим се променљиве A и $L1$ инстанцирају и у случају да чињеница $inc\_po\_l(A, L1)$ није још увек изведена може се закључити да то инстанцирање није успешно и може се покушати са другим инстанцирањем.

Разбијање аксиома које уводе више сведока. На основу правила описаних у поглављу 2.4.1, аксиома:

[4] Предикатски симбол $inc\_po\_l(A, L)$ означава да тачка A припада правој L. Целокупан списак предикатских симбола коришћен приликом записивања аксиоматског система који је коришћен за евалуацију доказивача се налази у додатку B.

$line(L1) \Rightarrow \exists A \exists B(point(A) \wedge point(B) \wedge inc\_po\_l(A, L1) \wedge inc\_po\_l(B, L1) \wedge A \neq B)$

неће бити примењена на конкретну праву P (када инстанцирамо променљиву $L1$) у случају да већ постоје константе A и B такве да важи $point(A) \wedge point(B) \wedge inc\_po\_l(A, P) \wedge inc\_po\_l(B, P) \wedge A \neq B$. Међутим, поставља се питање да ли та аксиома треба бити примењена у случају када постоји константа A таква да важи $point(A) \wedge inc\_po\_l(A, P)$, али не постоји константа B таква да важи $point(B) \wedge inc\_po\_l(B, P) \wedge A \neq B$. У математичким уџбеницима, примена те аксиоме се у том случају мало модификује. Не примењује се *у потпуности* дата аксиома, већ се најчешће уводи само једна нова константа B и чињенице $point(B) \wedge inc\_po\_l(B, P) \wedge A \neq B$. Како би, стриктно гледано, примена ове аксиоме увела нове две константе C и D, потребно је у аксиоматски систем додати донекле измењену варијанту почетне аксиоме која би оправдала овакав корак:

$line(L1) \wedge point(A) \wedge inc\_po\_l(A, L1) \Rightarrow \exists B(point(B) \wedge inc\_po\_l(B, L1) \wedge A \neq B)$

Овакво тврђење је корисно и из разлога ефикасности јер уводи мање константи од почетне аксиоме. Стога ћемо, осим коришћења почетне аксиоме, у скуп *аксиома* додати и помоћну лему. Ова лема би требала да има већи приоритет од почетне аксиоме (да би доказивач прво покушао њу да примени).

Сличан механизам извођења нових лема може бити примењен на све аксиоме које у закључку имају више од једног егзистенцијалног квантификатора. Поступак разбијања таквих аксиома на нове леме није увек тривијалан као у претходном случају и биће детаљније објашњен у поглављу 3.4.2.

Теорија једнакости. Доказивач ArgoCLP пружа подршку за теорију једнакости, ако корисник жели да је користи. Тада се аксиоме једнакости не наводе експлицитно. Овај механизам је подржан класама једнакости над константама. У сваком тренутку у току рада програма уместо коришћења целе класе објеката, користи се само канонски представник те класе. На почетку процеса доказивања свака константа представља јединствену класу и класе се одржавају уз помоћ Тарјанове (Robert Tarjan) *union-find* структуре [90].

На пример, ако за константе A, P, Q и R важе чињенице $point(A)$, $line(P)$,

$line(Q)$, $plane(R)$, $inc\_po\_l(A, P)$, $inc\_po\_l(Q, R)$, $inc\_l\_pl(Q, R)$ и $P = Q$, наредна аксиома може бити примењена:

$point(X) \wedge line(L) \wedge plane(W) \wedge inc\_po\_l(X, L) \wedge inc\_l\_pl(L, W) \Rightarrow inc\_po\_pl(X, W)$

над константама $X = A$, $L = P$, $W = R$, и чињеница $inc\_po\_pl(A, R)$ ће бити изведена. Наиме, у случају таквог инстанцирања приликом провере да ли важи чињеница $inc\_po\_l(P, R)$, прво се одређују канонски представници за P и R — нека су то на пример, Q и R — и како чињеница $inc\_po\_l(Q, R)$ важи, аксиома може бити примењена.

Аксиоме једнакости нису коришћене експлицитно током процедуре доказивања, али ће бити генерисане и коришћене приликом креирања потпуног (машински проверивог) доказа. Ови кораци би се могли формално проверити и без експлицитног навођења конкретне аксиоме једнакости (пошто интерактивни доказивачи теорема имају уграђену аксиоматизацију логике са једнакошћу), али је информација о конкретној аксиоми која се користи ипак сачувана у нацрту доказа који генерише доказивач ArgoCLP.

Рад са симетричним предикатским симболима. За предикатски симбол R кажемо да је *симетричан* ако је следеће (универзално квантификовано) тврђење тачно за свако $1 \leq i, j \leq n$:

$R(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) \Leftrightarrow R(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$

Приликом рада са симетричним предикатским симболима довољно је разматрати само *представнике* чињеница. На пример, уместо чувања двеју чињеница $col(A, B, C)$ и $col(C, B, A)$, довољно је чувати само чињеницу $col(A, B, C)$ (где col представља релацију колинеарно). Представник класе чињеница може бити одређен на следећи начин: захваљујући нумерацији константи, аргументи предикатских симбола се могу сортирати тако да се минимална пермутација изабере као представник. Представник класе се одређује само приликом рада са симетричним предикатским симболима. Овај механизам се може комбиновати са подршком за једнакост објеката да би се још више смањио број непотребно изведених чињеница. Слично као у случају аксиома једнакости, леме које се тичу симетричности предикатских симбола се не користе током процеса доказивања, већ само на крају процеса доказивања приликом креирања потпуног (машински проверивог) доказа.

На пример, у случају постојања константи A, B, C и D (за које важе чињенице $point(A)$, $point(B)$, $point(C)$ и $point(D)$) са нумерацијом којој одговара уређење $A < B < C < D$, и чињеницама $ncol(C, B, D)$ и $col(A, D, C)$, након извођења чињенице $A = B$ класе еквиваленције та два објекта ће бити спојене и биће откривена контрадикција. Наиме, како је A представник класе еквиваленције којој припадају константе A и B, представник чињенице $ncol(C, B, D)$ биће $ncol(C, A, D)$. Захваљујући својству симетрије представник те чињенице ће бити $ncol(A, C, D)$. Са друге стране, захваљујући својству симетрије представник чињенице $col(A, D, C)$ ће бити $col(A, C, D)$, па се из чињеница $ncol(A, C, D)$ и $col(A, C, D)$ може извести контрадикција.

Информација да ли је неки предикатски симбол симетричан или не, се може проверити аутоматски (у фази претпроцесирања) што ће бити детаљније објашњено у поглављу 3.4.1.

Коришћење овде описаних техника и даље неће гарантовати налажење доказа тешких теорема (у разумном временском року). Осим тога, могуће је да ће генерисани докази и даље садржати неке ирелевантне чињенице.

3.1.3 Технике које не чувају својство потпуности

У циљу побољшања ефикасности доказивача, приликом доказивања одређених теорема, могу бити коришћене и неке технике које нарушавају потпуност процедуре доказивања.

Ограничавање гранајућих аксиома. У основној процедури претраге гранајуће аксиоме облика $R(\vec{x}) \vee \overline{R}(\vec{x})$ ће бити генерисане и коришћене за све предикатске симболе задате унутар теорије са којом се ради. Уместо коришћења оваквих аксиома за све предикатске симболе могуће је користити гранајуће аксиоме само за примитивне предикатске симболе (који нису уведени дефиницијама у оквиру теорије). Осим тога, могуће је доказати да се избацивањем гранајућих аксиома за неке дефинисане предикатске симболе не нарушава потпуност процедуре одлучивања [16].

Ограничење примене аксиома. У току процедуре претраживања може се увести ограничење на примену само оних аксиома које се састоје само од предикатских симбола који се појављују у тврђењу које се доказује. Може се увести и блаже ограничење и увести ограничење на примену

само оних аксиома које у себи садрже макар један предикатски симбол који се појављује у тврђењу које се доказује.

Ова ограничења, иако нарушавају својство потпуности доказивача не нарушавају сагласност и могу се показати као пресудна у случају доказивања одређених теорема. Дешава се да тек са додатним ограничењима доказивач постане довољно моћан да докаже те теореме и са практичне стране нарушавање потпуности на овај начин нема негативних последица.

3.2 Имплементација доказивача и начин коришћења

Доказивач ArgoCLP је имплементиран у програмском језику C++. Састоји се од око 5000 линија кода који је организован у 23 класе. Аксиоматски систем који се користи, као и теорема која се доказује се задају програму кроз посебне датотеке, тако да се доказивач може користити за разне теорије кохерентне логике.

Постоје два начина задавања теорије, коришћењем TPTP формата[5] [87] или задавањем теорије кроз неколико различитих датотека у формату који ће бити описан у наставку текста. И у једном и у другом формату подразумева се да су све формуле записане у кохерентној форми, односно да су универзално квантификоване, да се у премисама налази конјункција атомичких формула, а у закључку се налази егзистенцијално квантификована дисјунктивна нормална форма (при чему егзистенцијални квантификатор у закључку није обавезан).

Пример задавања формула у TPTP формату. Илустрације ради наведена је само једна аксиома и једна дефиниција, а не целокупан скуп коришћен приликом доказивања наведеног тврђења. У датотеци се може наћи произвољан број аксиома и дефиниција и само једно тврђење које се доказује. Запис тврђења почиње кључном речју *fof*. Приликом навођења аксиома и дефиниција прво се наводи име аксиоме или дефиниције па затим кључна реч *axiom*, док се тврђење које се доказује наводи са кључним речима *goal* и *conjecture*. Након тога се наводи формула записана у кохерентном формату.

```
fof(ax_I1, axiom, (![A,B] : ((point(A) & point(B) & A!=B) =>
                  (?[L] : (line(L) & inc_po_l(A,L) & inc_po_l(B,L)))))).
fof(ax_D1, axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C)
```

[5] http://www.cs.miami.edu/~tptp/

```
                & line(L) & inc_po_l(A,L) & inc_po_l(B,L)
                & inc_po_l(C,L)) => (col(A,B,C))))).
fof(goal, conjecture, (![A,B,C,L] : ((point(A) & point(B) & point(C)
                & line(L) & inc_po_l(A,L) & inc_po_l(B,L)
                & bet(A,B,C)) => (inc_po_l(C,L))))).
```

Пример задавања формула у ArgoCLP формату. У наставку текста је приказан запис истог скупа тврђења кроз неколико датотека које се користе. Те датотеке су:

Сигнатура теорије: у оквиру које се наводе имена типова објеката, након кључне речи *types*, на пример:

```
types point line plane
```

иза које следи списак предикатских симбола заједно са листом типова аргумената за сваки предикатски симбол. На пример, предикатски симбол *inc_po_l* (тачка припада правој) се задаје у облику:

```
datatype inc_po_l point line
```

Осим тога, предикатски симбол *eq_type* ће се користити за означавање једнакости између два објекта типа *type* и њега корисник не мора експлицитно наводити.

Скуп аксиома: аксиоме се задају у облику који је илустрован наредним примером:

```
premises
point(1)
point(2)
~eq_point(1,2)

conclusions
line(3)
inc_po_l(1,3)
inc_po_l(2,3)
```

Променљиве су представљене природним бројевима. Универзална квантификација се подразумева, као и конјункција и у премисама и у закључцима (дисјункција мора бити експлицитно наведена симболом | и

дозвољена је само у закључку). У наведеном примеру променљиве 1 и 2, које су уведене унарним предикатом који представља тип *тачка*, су универзално квантификоване. Променљива 3 се налази у закључку и она је егзистенцијално квантификована.

Скуп дефиниција: у одређеним теоријама се користи јако велики скуп дефиниција којима се уводе нови предикатски симболи. Дефиниције се користе због повећане читљивости и наводе се у истом формату као аксиоме:

```
premises
point(1)
point(2)
point(3)
line(4)
inc_po_l(1,4)
inc_po_l(2,4)
inc_po_l(3,4)

conclusions
col(1,2,3)
```

Тврђење које се доказује: задаје се у истом формату као аксиоме:

```
premises
point(1)
point(2)
point(3)
line(4)
inc_po_l(1,4)
inc_po_l(2,4)
bet(1,2,3)

conclusions
inc_po_l(3,4)
```

Технике описане у поглављу 3.1.2 које се увек користе у доказивачу ArgoCLP су: груписање аксиома и додељивање приоритета, рано одсецање и подршка за теорију једнакости. Разбијање аксиома које уводе више сведока, као и коришћење симетричних предикатских симбола може бити обрађено у фази претпроцесирања на начин који ће бити накнадно објашњен у поглављу 3.4.

Осим задавања теорије у којој се ради и теореме која се доказује, корисник има могућност задавања и датотеке за конфигурацију доказивача. Уз помоћ ове датотеке корисник има могућност избора које од техника из поглавља 3.1.2 и 3.1.3 ће бити коришћене. Скуп опција који је понуђен кориснику је следећи:

equality опција, ако је укључена биће коришћено једнакосно резоновање унутар доказивача.

excluded_middle опција, ако је укључена, аксиоме искључења трећег ће бити коришћене.

опције за технике које не очувавају потпуност у овом случају у питању је неколико различитих параметара па корисник може изабрати: да ли жели да користи само аксиоме које су исказане над предикатским симболима из тврђења које се доказује; да ли жели да користи само аксиоме које садрже бар један од предикатских симбола из тврђења које се доказује; да ли ће се бројач s који контролише примену аксиома увећавати пре примене јако продуктивних аксиома или не.

Током процеса доказивања, ArgoCLP генерише нацрт доказа који у себи садржи све релевантне информације. На основу тог нацрта доказа формира се излазна XML датотека о којој ће бити више речи у поглављу 4.2. Пре генерисања излазне датотеке, у оквиру доказивача се позива додатни механизам за елиминисање ирелевантних корака доказа [59] (укључујући и кораке гранања), чиме се добија оптимизован „чист" (често знатно краћи) нацрт доказа. Корак гранања је релевантан само ако обе гране користе претпоставке направљене гранањем, иначе се гранајући корак може заменити граном која не користи претпоставку гранања.

Пример 3.1 *Размотримо наредно тврђење: Нека су дате три праве p, q и r, и раван α која их садржи. Ако важи да су праве p и q различите, и праве q и r различите, и праве p и q се не секу, и праве q и r се не секу и ако постоји тачка A која припада равни α и правама p и r, онда су праве p и r једнаке.*

Записано у Argo формату, ово тврђење изгледа овако:

```
premises
# TH_8
% for three lines and a plane which contains them all holds that
% if first and second are distinct and second and third are distinct
% and first and second do not intersect and second and third do not
```

```
% intersect and if there exists a point which belongs to the plane
% and to the first and third line, then first and third line are equal

line(1)
line(2)
line(3)
plane(4)
~eq_line(1,2)
~eq_line(2,3)
~int_l_l(1,2)
~int_l_l(2,3)
inc_l_pl(1,4)
inc_l_pl(2,4)
inc_l_pl(3,4)
point(5)
inc_po_pl(5,4)
inc_po_l(5,1)
inc_po_l(5,3)

conclusions

eq_line(1,3)
```

Кључни део генерисаног (оптимизованог) Isabelle/Isar доказа је приказан испод:

```
...
lemma  TH_8:
assumes "LI1 ~= LI2"
and "LI2 ~= LI3"
and "\<not>int_l_l LI1 LI2"
and "\<not>int_l_l LI2 LI3"
and "inc_l_pl LI1 PL1"
and "inc_l_pl LI2 PL1"
and "inc_l_pl LI3 PL1"
and "inc_po_pl PO1 PL1"
and "inc_po_l PO1 LI1"
and "inc_po_l PO1 LI3"
shows "LI1 = LI3"
proof -

(*1*)
have  "LI1 = LI3 \<or> LI1 ~= LI3"
using ax_g_ex_mid_3 [of "LI1" "LI3"]
by auto
(*2*)  moreover
{ assume "LI1 = LI3"
```

```
(*3*)
from this
have ?thesis
by auto
} note note1 = this
(*4*)  moreover
{ assume "LI1 ~= LI3"
(*5*)  moreover
have  "inc_po_l PO1 LI2 \<or> \<not>inc_po_l PO1 LI2"
using ax_g_ex_mid_7 [of "PO1" "LI2"]
by auto
(*6*)  moreover
{ assume "inc_po_l PO1 LI2"
(*7*)  moreover
from ‘LI1 ~= LI2‘ and ‘inc_po_l PO1 LI1‘ and ‘inc_po_l PO1 LI2‘
have  "int_l_l LI1 LI2"
using ax_D5 [of "LI1" "LI2" "PO1"]
by auto
(*8*)  moreover
from ‘int_l_l LI1 LI2‘ and ‘\<not>int_l_l LI1 LI2‘
have False
by auto
(*9*)
ultimately
have False
by auto
} note note2 = this
(*10*)  moreover
{ assume "\<not>inc_po_l PO1 LI2"
(*11*)  moreover
from ‘\<not>int_l_l LI1 LI2‘
have  "\<not>int_l_l LI2 LI1"
using ax_nint_l_l_21 [of "LI1" "LI2"]
by auto
(*12*)  moreover
from ‘\<not>inc_po_l PO1 LI2‘ and ‘inc_po_pl PO1 PL1‘ and ‘inc_l_pl LI2 PL1‘
and ‘inc_po_l PO1 LI1‘ and ‘inc_l_pl LI1 PL1‘ and ‘\<not>int_l_l LI2 LI1‘
and ‘inc_po_l PO1 LI3‘ and ‘inc_l_pl LI3 PL1‘ and ‘\<not>int_l_l LI2 LI3‘
have  "LI1 = LI3"
using ax_E2 [of "PO1" "LI2" "PL1" "LI1" "LI3"]
by auto
(*13*)  moreover
from ‘LI1 = LI3‘ and ‘LI1 ~= LI3‘
have False
by auto
```

```
(*14*)
ultimately
have False
by auto
} note note3 = this
(*15*)  from note2 and note3 and ‘inc_po_l PO1 LI2 | \<not>inc_po_l PO1 LI2‘
have False
by auto
(*16*)
ultimately
have False
by auto
} note note4 = this
(*17*)  from note1 and note4 and ‘LI1 = LI3 | LI1 ~= LI3‘
have ?thesis
by auto
ultimately
show ?thesis
by auto
qed
```

Доказ генерисан на природном језику (након процеса оптимизације) се налази у наставку текста. Да би докази били више налик доказима у математичким уџбеницима, могу се направити додатне трансформације приликом исписа и докази могу бити записани у форми коришћења контрадикције (*reductio ad absurdum* [59]).

Theorem TH_8:

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.

   Contradiction.

Therefore, it holds that $p = r$.

This proves the conjecture.

Theorem proved in 9 steps and in 0.02 s.

3.3 Евалуација

За евалуацију доказивача ArgoCLP, коришћена су четири аксиоматска система за еуклидску геометрију равни и доказивач је примењен над десетак теорема из стандардних универзитетских уџбеника за геометрију.

Приликом рада са различитим аксиоматским системима нису мењана подешавања доказивача. Аксиоматски системи који су коришћени су: Хилбертов аксиоматски систем [45], систем Тарског [91, 79], систем који су дефинисали Борсук и Шмилев [14] и аксиоматски систем доказивача Еуклид (*EUCLID* [46]). Коришћена је унија свих типова објеката и предикатских симбола који су коришћени у појединачним аксиоматским системима. Ограничење које се намеће је природно, а то је да аксиоматски систем који не садржи одређене предикатске симболе не може бити коришћен за доказивање својстава тих предикатских симбола (нпр., аксиоматски систем Тарског, који користи само тачке, не може бити коришћен за доказивање својстава релације

инциденције између тачке и праве). Цео скуп аксиома ових аксиоматских система је коришћен у овом експерименту, осим аксиома непрекидности (због њене сложености). Ово ограничење није од пресудног значаја из разлога што се велики део геометрије може доказати и без њих. У неким случајевима било је потребно преформулисати запис аксиоме, али је вођено рачуна о томе да се задржи оригинално значење аксиоме.

Процедура записивања аксиома за потребе аутоматског доказивања није увек тривијална. Оригиналне формулације аксиома које се налазе у књигама су често непрецизне и неки услови који недостају често се подразумевају. На пример, у запису својих аксиома, Хилберт приликом употребе фразе „две тачке" претпоставља да су те две тачке различите али не наводи ту информацију експлицитно приликом записивања аксиома и теорема. Лаура Мејкл и Жак Флорио су први истакли овај проблем [60]. Приликом решавања проблема тог типа требало би водити рачуна о томе да ли различите модификације аксиоматског система мењају скуп теорема доказивих у том систему. Овај проблем неће бити разматран у овој тези.

Како је већ показано [30], докази многих тврђења у оквиру Хилбертовске геометрије користе правило искључења трећег, па се (у складу са правилима кохерентне логике) уводе аксиоме облика $R(\vec{x}) \vee \overline{R}(\vec{x})$ (за све предикатске симболе R). Са овим додатком аксиоматском систему и даље остајемо у оквиру интуиционистичког приступа и не губимо потпуност процедуре претраге и доказивања.

Скуп теорема над којима је тестиран доказивач ArgoCLP је добијен из различитих уџбеника за геометрију и наведен је у наставку текста. Састоји се од 14 теорема и у њему се налазе како теореме које је доказивач успео да докаже, тако и теореме које доказивач није успео да докаже (у оквиру временског ограничења од 30 секунди).

Теорема 1: За две праве које се секу важи да постоји раван која их садржи.

Теорема 2: Ако две различите праве имају заједничку тачку, онда је она јединствена.

Теорема 3: За раван и праву која не лежи у тој равни, ако имају заједничку тачку онда важи да је та тачка јединствена.

Теорема 4: За три неколинеарне тачке важи да су у паровима различите.

Теорема 5: За три неколинеарне тачке важи да постоји раван која их садржи.

Теорема 6: За праву и тачку која не припада тој правој важи да постоји раван која их садржи.

Теорема 7: За четири компланарне тачке важи да ако три од њих нису

колинеарне и припадају некој равни, онда и четврта тачка припада тој равни.
Теорема 8: За три праве p, q и r од којих су p и q, и q and r различите и раван α која их садржи важи да ако се праве p и q не секу и праве q и r не секу и ако постоји тачка A која припада равни α и правама p и r, онда се праве p и r поклапају.
Теорема 9: Ако тачке A и B припадају правој p и постоји тачка C тако да је тачка B између тачака A и C онда и тачка C припада правој p.
Теорема 10: За три тачке A, B и C такве да тачка B лежи између тачака A и C важи да тачка C не лежи између тачака A и B.
Теорема 11: За две различите тачке постоји тачка која лежи између њих.
Теорема 12: Релација конгруенције је рефлексивна.
Теорема 13: Релација конгруенције је симетрична.
Теорема 14: Релација конгруенције је симетрична по паровима.

У оквиру овог експеримента нису коришћена додатна подешавања доказивача прилагођена одређеном аксиоматском систему, већ су коришћена иста подешавања за сва четири аксиоматска система и укључују: коришћење само оних аксиома које се састоје од предикатских симбола који се налазе у теореми која се доказује, и правило искључења трећег само над примитивним (не и над дефинисаним) предикатским симболима. Као што је и очекивано, доказивост одређеног тврђења (и време доказивања) зависи од аксиоматског система који се користи. У табели 3.1 су приказани резултати евалуације доказивача.

У овом експерименту, приликом доказивања сваке теореме, користе се само аксиоме одговарајућег аксиоматског система, а не и леме или претходно доказане теореме, што знатно отежава проналажење доказа. Иако није у стању да докаже све задате теореме, може се видети потенцијал примене кохерентних доказивача.

3.4 Трансформисање система аксиома

Као што је раније речено, у току рада кохерентног доказивача као што је ArgoCLP број генерисаних чињеница може бити веома велики. Знатан број тих чињеница је тривијалан када се доказ разматра са становишта доказа који се налазе у уџбеницима. Показује се да се захваљујући симетричности одређених предикатских симбола, као и увођењем помоћних лема (у скуп аксиома које се користе у оквиру рада доказивача) тај број чињеница може смањити [81]. У овом поглављу ће бити представљене две технике за убрзање рада доказивача,

Табела 3.1: *Резултати примене доказивача; формат резултата је у облику време/n_1/n_2, при чему је n_1 број коришћених аксиома у доказу, а n_2 број коришћених аксиома у „чистом" доказу (из ког су елиминисани сви непотребни кораци) записаном у природном језику; '-' означава да је истекло временско ограничење, NA означава да теорему није могуће доказати у оквиру датог аксиоматског система; експеримент је покренут на PC Core 2Quad 2.4GHz са 4GB RAM, под Linux-ом.*

#	Еуклид	Тарски	Борсук	Хилберт
1	-	NA	-	-
2	0.01/5/3	NA	0.01/5/3	0.01/5/3
3	0.01/5/3	NA	0.01/5/3	0.01/5/3
4	-	NA	-	-
5	0.01/27/1	NA	0.03/28/1	-
6	-	NA	16.07/524/59	-
7	11.08/125/4	NA	8.09/119/4	-
8	0.01/12/9	NA	0.01/12/9	0.01/12/9
9	-	NA	-	-
10	0.01/2/1	-	0.01/2/1	-
11	-	-	0.07/71/8	-
12	0.01/5/2	0.01/6/2	0.01/6/2	-
13	0.25/13/3	0.16/24/3	0.22/24/3	-
14	1.26/26/7	0.52/30/7	0.57/30/7	-

а затим и резултати њихове примене.

3.4.1 Симетрични предикатски симболи

Коришћење својства симетричности предикатских симбола може утицати на смањење величине простора претраге и увођење мањег броја тривијалних корака. Препознавање симетричних предикатских симбола и начин њиховог коришћења ће бити обрађен кроз додатну технику за претпроцесирање аксиоматског система. Са том техником, докази које генерише кохерентни доказивач ArgoCLP постају значајно краћи, читљивији, и више налик доказима који се могу наћи у математичким уџбеницима.

За процес аутоматског откривања симетричних предиката потребно је разматрати саму дефиницију симетричних предиката, као и теорему која следи која ће се показати кључна за ефикасан процес откривања симетричних предиката.

Дефиниција 3.1 *(Симетричан предикат) Предикат R арности n је симетричан, по свим својим аргументима, ако је наредно (универзално*

квантификовано) тврђење теорема за сваку пермутацију σ*:*

$$R(x_1, \ldots, x_n) \Leftrightarrow R(x_{\sigma(1)}, \ldots, x_{\sigma(n)}).$$

Теорема 3.1 *Предикат* R *арности* n *је симетричан ако и само ако су наредна два тврђења теореме:*

$$R(x_1, x_2, x_3, \ldots, x_n) \Leftrightarrow R(x_2, x_1, x_3 \ldots, x_n) \tag{3.1}$$

$$R(x_1, x_2, x_3 \ldots, x_n) \Leftrightarrow R(x_2, x_3, \ldots, x_n, x_1) \tag{3.2}$$

У случају да су ове две формуле теореме, може се показати да су и све друге формуле које изражавају симетрију над осталим пермутацијама такође теореме. Тако одређен скуп теорема са својим доказима је потребан у процесу аутоматског доказивања теорема и приликом генерисања формалних, машински проверивих доказа. Тај скуп теорема ћемо звати *симетричне леме*.

Приликом рада са симетричним предикатима издвајају се наредне три фазе:

Фаза претпроцесирања. Тврђења облика (3.1) и (3.2) могу бити аутоматски генерисана на основу скупа предикатских симбола који се користе у оквиру аксиоматског система са којим се доказује тврђење. Након тога, аутоматски доказивач може бити коришћен у циљу проверавања да ли су та тврђења теореме (како неће сва тврђења бити теореме, потребно је поставити временско ограничење). За оне предикатске симболе за које се покаже да су симетрични, скуп симетричних лема такође може бити аутоматски генерисан и касније може бити коришћен приликом генерисања формалног доказа. Информација о симетричности предикатских симбола се користи у оквиру самог рада доказивача на начин описан у наставку текста.

Фаза доказивања. Због ефикаснијег рада са симетричним предикатским симболима, све пермутације аргумената симетричног предикатског симбола ће бити представљене јединственом пермутацијом — на пример минималном пермутацијом (у односу на лексикографски поредак). Ако користимо предикатски симбол *col* и константе A, B, C, обе чињенице $col(B, A, C)$ и $col(A, C, B)$ ће бити представљене у бази знања чињеницом $col(A, B, C)$ и због тога ћемо их сматрати идентичним приликом фазе доказивања.

Фаза реконструкције доказа на објектном нивоу. У току фазе реконструкције доказа изједначавање чињеница $col(B, A, C)$ и $col(A, C, B)$ ће бити оправдано коришћењем наредне две леме:

$col(B, A, C) \Rightarrow col(A, B, C)$

$col(A, B, C) \Rightarrow col(A, C, B)$

и формални, комплетан, доказ ће бити генерисан (под претпоставком да је те две леме већ доказао доказивач).

3.4.2 Аксиоме које уводе више сведока у закључку

Анализирањем доказа теорема у системима налик Хилбертовом, може се приметити да се неке аксиоме ретко примењују у свом оригиналном облику. На пример, аксиома *I3: Права садржи две различите тачке* се, у случају да постоји већ једна тачка која припада правој, употребљава на начин описан наредним тврђењем:

I3a: Ако тачка A припада правој p, онда постоји још једна тачка B различита од A која припада p.

Овакав начин примене аксиоме, иако присутан у математичким уџбеницима, није прецизан и не би прошао формалну верификацију доказа. Оно што би требало урадити је увести две нове тачке B и C, са својствима: B и C су различите и припадају p, и доказати да се једна од њих мора разликовати од тачке A. Без обзира на то, испоставља се да је ова пракса оправдана из разлога што се формула $I3a$ може доказати као теорема, што би свакако био неопходан корак приликом формалног доказивања.

Овакво (неформално) модификовање аксиоматског система је могуће због тога што аксиома $I3$ уводи два објеката у закључку. Сличан принцип се може применити и у општем случају када аксиома уводи више објеката у закључку.

Разматрајмо специјалан случај формуле кохерентне логике:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y} B(\vec{x}, \vec{y}) \tag{3.3}$$

за $n \geq 0$, $\vec{x} = \{x_1, x_2, \ldots, x_k\}$ $(0 \leq k)$, $\vec{y} = \{y_1, y_2, \ldots, y_l\}$ и $l \geq 2$ (где l означава број уведених сведока). Приликом процеса доказивања, у контексту уланчавања унапред, примена аксиоме овог типа *уводи l сведока*: [6]

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists y_1 \exists y_2 \ldots \exists y_l B(\vec{x}, y_1, \ldots, y_l).$$

[6] Преименовала сам k у l да би се слагало са претходним формулама. Нисам означавала ову промену у наредном пасусу.

У случају када је $l = 1$ кохерентни доказивач теорема неће применити аксиому облика $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists y_1 B(\vec{x}, y_1)$ ако постоји константа a за коју важи $B(\vec{x}, a)$, али у случају да је $l > 1$ (и да постоји l егзистенцијалних квантификатора) проверавање постојање свих l сведока. Ова аксиома *неће* бити примењена *само у случају* да сви сведоци (који инстанцирају $y_1, \ldots, y_l$) већ постоје, али ако макар један од њих не постоји аксиома ће бити примењена и примена те аксиоме ће увести нових l сведока. Поставља се питање да ли се могу прецизно дефинисати услови који обезбеђују да, у случају постојања неких сведока који задовољавају аксиому, примена аксиоме уведе мање од l сведока.

Претпоставимо да се у тврђењу облика (3.3) формула B може записати као конјункција атомичких формула $B = B_1 \wedge B_2$, где је B_1 конјункција свих атомичких формула у којима се појављују само променљиве из $\vec{x}$ и y_1 (ако такве атомичке формуле не постоје, B_1 је $\top$), а B_2 је непразна конјункција свих осталих атомичких формула из формуле B:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists y_1 \ldots \exists y_l (B_1(\vec{x}, y_1) \wedge B_2(\vec{x}, y_1, \ldots, y_l))$$

Разматрамо транформисану верзију претходног тврђења:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \wedge B_1(\vec{x}, y_1) \Rightarrow \exists y_2 \ldots \exists y_l B_2(\vec{x}, y_1, \ldots, y_l) \tag{3.4}$$

Ово тврђење уводи мање сведока од полазног тврђења. У општем случају, такво тврђење није последица полазног аксиоматског система, и није доказиво унутар теорије са којом се ради. Међутим, у случајевима када је то тврђење теорема, оно може бити коришћено приликом процеса доказивања уместо оригиналне аксиоме, ако су испуњени одговарајући услови.

Приликом рада са аксиомама које уводе више сведока у закључку издвајају се наредне две фазе:

Фаза претпроцесирања. Пролазећи кроз скуп аксиома теорије, аутоматском претрагом могу бити препознате аксиоме облика (3.3) и трансформисана тврђења могу бити генерисана. Аутоматски доказивач затим може бити коришћен за проверавање да ли је генерисано тврђење теорема (с обзиром да неће сва генерисана тврђења бити теореме, мора се поставити временско ограничење). У случају да је генерисано тврђење теорема, над њим се поново може спровести иста трансформација (ако задовољава потребне услове). Овај процес се понавља све док се не добије празна формула B_2. У случају да генерисано тврђење није теорема, корисник га може модификовати и поново покренути

процес провере да ли је такво тврђење теорема.

Фаза аутоматског доказивања. Леме генерисане на овај начин се приликом доказивања користе као аксиоме. Али, да би коришћење таквих лема имало ефекта на рад доказивача, у случајевима када су и оригинална аксиома и изведена лема применљиве, лема мора бити употребљена пре оригиналне аксиоме чиме се уводи мање константи у току процеса доказивања.

Пример 3.2 *Аксиома I*3: Права садржи две различите тачке.

Аутоматским трансформисањем ове аксиоме добија се управо тврђење са почетка овог поглавља:
*I*3*a: Ако тачка A припада правој p, онда постоји још једна тачка B различита од A која припада p.*
Трансформисано тврђење *I*3*a* јесте теорема и може се користити уместо оригиналне аксиоме у случајевима када је то могуће. То не важи у општем случају, и да неће увек сва трансформисана тврђења бити теореме што се може видети у наредном примеру.

Пример 3.3 *Аксиома* I8: Постоје три неколинеарне тачке.

Аксиома *I*8 ће, на начин описан раније, генерисати следећа два тврђења:

1. *За дату тачку A, постоје тачке B и C такве да су тачке A, B и C неколинеарне.*
2. *За дате тачке A и B, постоји тачка C таква да су тачке A, B и C неколинеарне.*

У овом случају, прво тврђење јесте теорема, док друго тврђење није теорема. Испоставља се да се друго тврђење може *модификовати* тако да буде теорема, додавањем, у премисама, услова да су тачке A и B различите.

3.4.3 Имплементација и евалуација

У овом поглављу биће размотрена корисност двеју техника претпроцесирања аксиоматског система које су описане у претходна два поглавља, као и ефекат које те технике имају на рад самог доказивача. Алат за претпроцесирање је издвојен од рада самог доказивача тако да се модификације над аксиоматским системом изводе само једном (пре почетка рада доказивача над целом

теоријом).[7] Систем је примењен над Хилбертовим аксиоматским системом (само над аксиомама прве групе).

Аксиоматски систем. Аксиоматски систем коришћен за анализу техника представљених у овом поглављу је базиран на Хилбертовом аксиоматском систему [45]. Како је објашњено у поглављу 2.4.1, за сваки предикатски симбол теорије[8] R додаје се нови предикатски симбол $\overline{R}$ и аксиома $R \wedge \overline{R} \Rightarrow \bot$. Аксиома $R \vee \overline{R}$ ће бити додата за све предикатске симболе осим оних који представљају релације колинеарности и припадности праве равни (чиме се не нарушава потпуност процедуре одлучивања [16]). За предикатске симболе који представљају релације колинарно и припадност праве равни, одговарајућа правила искључења трећег могу да се докажу као теореме теорије коју разматрамо. Аксиоматски систем који се користи се налази у додатку B.

Аутоматско откривање симетричних предикатских симбола. Својство симетричности предикатских симбола R и $\overline{R}$ се доказује засебно и одговарајуће симетричне леме ће бити коришћене приликом комплетирања финалних доказа. Разматрају се само предикатски симболи чији су аргументи сви истог типа (позитиван и негативан облик предиката: пресек двеју правих, пресек двеју равни, распоред тачака, колинеарност, копланарност и подударност). На начин описан у претходном поглављу генерише се укупно 20 тврђења облика (3.1) и (3.2) (за предикатске симболе арности два, ова два тврђења су идентична). Предикатски симболи за које се оба тврђења докажу као теореме ће користити своје својство симетрије у процесу доказивања теорема.

Међу набројаним предикатима, наредних осам су симетрични: позитиван и негативан облик предиката пресек двеју правих, пресек двеју равни, колинеарност и копланарност. ArgoCLP је успешно доказао да је 7 од ових 8 предиката симетрично (сви сем предиката $\overline{coplanarity}$) са просечним временом извршавања 3.6 секунди и просечним бројем корака 170. За предикатске симболе за које је показано да су симетрични, све леме које изражавају симетричност пермутација су генерисане (за њихове доказе је опет коришћен доказивач ArgoCLP). Таквих лема има 40 и доказују се са временом испод 2 секунде.

[7] Материјали се могу наћи на адреси http://www.matf.bg.ac.rs/~sana/system.zip. Сви експерименти су спроведени над AMD Opteron 2GHz са 96GB RAM.

[8] Инциденција тачке и праве, инциденција тачке и равни, инциденција праве и равни; пресек двеју правих, пресек двеју равни; колинеарност трију тачака, копланарност четири тачке; релација распореда између три тачке, релација подударно између парова тачака.

Аутоматско преформулисање аксиома. Аксиоме које уводе више од једног сведока у закључку су наредне аксиоме (и могу бити откривене аутоматски):

I3a На правој постоје две различите тачке.

I3b Постоје бар три неколинеарне тачке.

I8 Постоје бар четири некопланарне тачке. [9]

Скуп аутоматски генерисаних тврђења добијених из ових аксиома (на начин описан у 3.4.2) је:

I3a1 $line(p) \wedge point(A) \wedge A \in p \Rightarrow \exists B(point(B) \wedge A \neq B \wedge B \in p)$

I3b1 $point(A) \Rightarrow \exists B \exists C(point(B) \wedge point(C) \wedge \neg col(A, B, C))$

I3b2 $point(A) \wedge point(B) \Rightarrow \exists C(point(C) \wedge \neg col(A, B, C))$

I8a $point(A) \Rightarrow \exists B \exists C \exists D(point(B) \wedge point(C) \wedge point(D) \wedge \neg cop(A, B, C, D))$

I8b $point(A) \wedge point(B) \Rightarrow \exists C \exists D(point(C) \wedge point(D) \wedge \neg cop(A, B, C, D))$

I8c $point(A) \wedge point(B) \wedge point(C) \Rightarrow \exists D(point(D) \wedge \neg cop(A, B, C, D))$

Међу овим тврђењима само два су доказана и генерисани су формални докази за њих (теореме $I3a1$ и $I3b1$). Осталим тврђењима недостају кључни услови да би били теореме: тврђењима $I3b2$ и $I8b$ недостаје премиса $A \neq B$, а тврђењу $I8c$ недостаје премиса $\neg col(A, B, C)$. Укупно време ове фазе претпроцесирања је 20 минута (са временским ограничењем од 5 минута).

Утицај на рад доказивача ArgoCLP. Описане технике су тестиране на доказивачу ArgoCLP са малим скупом од 24 једноставне теореме[10] које се доказују са доказивачем ArgoCLP. .
Теорема 1: Ако су тачке A и B различите и тачке A, B и C колинеарне, и тачке A, B и D колинеарне онда су и тачке A,C и D колинеарне.

[9] Ово нису оригиналне формулације Хилбертових аксиома. Аксиоме су незнатно модификоване због додатне прецизности а и да би могле да буду изражене у кохерентној логици. Оригиналне формулације ових аксиома гласе: [I3a] Права садржи бар две тачке, [I3b] Постоје бар три тачке које не припадају једној правој, [I8] Постоје бар четири тачке које не припадају једној равни.

[10] За предикатске симболе чије негације су уведене погодно изабраним дефиницијама, правило искључења трећег се налази у скупу теорема за тестирање.

Теорема 2: Три произвољне тачке A, B и C су или колинеарне или неколинеарне.
Теорема 3: Ако су тачке A и C различите и тачка A припада правој p и тачка B не припада правој p и тачка C припада правој p, тада тачке A, B и C нису колинеарне.
Теорема 4: Ако су тачке A и C различите и тачка A припада правој p и тачка C припада правој p и тачке A, B и C нису колинеарне, тада тачка B не припада правој p.
Теорема 5: Ако су праве p и q различите и не секу се и тачка A припада правој p, тада тачка A не припада правој q.
Теорема 6: Ако су равни α и β различите и не секу се и тачка A припада равни α, тада тачка A не припада равни β.
Теорема 7: Ако права p не припада равни α и тачка A припада правој p, тада тачка A не припада равни α.
Теорема 8: Ако су тачке A, B и C неколинеарне, тада постоји раван α тако да јој те тачке припадају.
Теорема 9: Ако су тачке A, B, C и D компланарне и тачке A, B и C нису колинеарне и припадају равни α, тада и тачка D припада равни α.
Теорема 10: За праву p и раван α важи да права припада равни или да права не припада равни.
Теорема 11: За три неколинеарне тачке важи да су у паровима различите.
Теорема 12: Постоје две различите тачке.
Теорема 13: За тачку A важи да је колинеарна сама са собом.
Теорема 14: Постоји тачка која не припада датој правој.
Теорема 15: Ако праве p и q припадају равни α, тада или постоји њихова заједничка тачка или се те две праве не секу.
Теорема 16: Ако две различите праве имају две заједничке тачке, онда су те две тачке идентичне.
Теорема 17: За две равни важи или да се не секу или да имају заједничку праву.
Теорема 18: Ако су две равни различите и имају заједничку тачку и заједничку праву, тада та тачка припада тој правој.
Теорема 19: Ако права не припада равни онда или постоји заједничка тачка или се права и раван не секу.
Теорема 20: Ако раван и права која јој не припада имају две заједничке тачке, онда су те тачке идентичне.
Теорема 21: Ако тачка не припада правој онда постоји раван која их садржи.

Теорема 22: Ако тачка не припада правој, тада постоји јединствена раван која их садржи.
Теорема 23: Ако две различите праве имају заједничку тачку, тада постоји раван која их садржи.
Теорема 24: Ако две различите праве имају заједничку тачку, тада је права која их садржи јединствена.

Просечно време доказивања ових теорема је 23 минута, али је медијана 2 минута (због постојања неколико тешких лема), док је просечан број корака 1374. Резултати описаних техника су приказане у наставку текста.

1. *Симетричност предикатских симбола.*

   Приликом евалуације добити од коришћења симетричних предикатских симбола коришћене су само симетричне леме за предикатске симболе за које је сам доказивач показао да су симетрични (односно 7 од 8 симетричних предикатских симбола). Коришћењем ове технике просечно време доказивања теорема је смањено за 56% и просечни број корака за 83% (просечно време доказивања је 10 минута и просечан број корака 230).

2. *Преформулисање аксиома.*

   Приликом евалуације добити од преформулисања аксиома коришћене су само леме које су доказане аутоматски (две леме). Када се пореди са верзијом доказивача који користи подршку за симетричне предикатске симболе, просечно време доказивања је смањено за 15% и просечни број корака за 37% (просечно време доказивања је 8.5 минута и просечан број корака 143).

3. *Укупно побољшање.*

   Када се пореде оригинални доказивач и доказивач који користи обе технике, коришћење обе технике убрзава време доказивања теорема за 63% и просечан број корака за 89%.

Добијени резултати показују корисност предложених техника и када се користе појединачно, а посебно приликом њиховог комбиновања.

3.5 Сродни системи и технике убрзавања процеса доказивања

У овом поглављу ће бити представљени други доказивачи чији је циљ генерисање читљивих доказа као и технике за убрзавање процеса доказивања

налик техникама описаним у овој тези.

3.5.1 Сродни системи

Кохерентна логика је погодна основа за аутоматске доказиваче чији је један од циљева и генерисање читљивих доказа. Такви доказивачи, се могу користити за доказивање једноставнијих тврђења или као асистенти приликом доказивања компликованијих тврђења. Колико је познато аутору ове тезе, први аутоматски доказивач теорема заснован на кохерентној логици су направили Предраг Јаничић и Стеван Кордић [46]. Тај доказивач је радио са фиксираним скупом аксиома који је јако близак Борсуковом аксиоматском систему [14] и био је у стању да докаже десетине основних теорема из стандардних уџбеника за геометрију. Генерисани докази су били читљиви али нису били формални. Систем који је описан у овој тези се у одређеној мери ослања на њихов систем али га на разне начине унапређује и поправља.

У последњих десетак година, кохерентна логика се користи све више, првенствено захваљујући Марк Безему (Marc Bezem) и његовим коауторима. Безем и Тери Кокан (Thierry Coquand) [8] су у Prolog-у развили доказивач за кохерентну логику који генерише доказе провериве у интерактивном доказивачу теорема Coq (неки проблеми решени од стране овог доказивача се могу наћи на интернету[11]). Штефан Бергхофер (Stefan Berghofer) и Безем су развили у *ML*-у интерни доказивач за кохерентну логику [7] у оквиру интерактивног доказивача теорема Isabelle.

Гансалингам и Гауерс су радили на аутоматском генерисању читљивих доказа [35], и предлажу правила извођења налик правилима кохерентне логике.

3.5.2 Технике за убрзање рада аутоматских доказивача

Постоје многе технике за убрзавање рада аутоматских и интерактивних доказивача. Почевши од хеуристика и модификација саме имплементације доказивача када се разматра проблем одређеног типа, до унапређења самог алгоритма доказивача када се разматра проблем општег типа. У овом поглављу биће разматране само технике блиске оним описаним у овој тези, односно технике које користе својство симетричности.

Постојање симетричних предиката и коришћење својства симетричности се стандардно употребљава у оквиру аутоматског резоновања [22, 28, 36]. Коришћење својстава симетричности у оквиру геометрије се јавља већ 1959. у

[11] http://www.ii.uib.no/~bezem/GL/

раду Гелертнера [36]. Он се бавио проблемима који укључују симетричне предикате и уз помоћ рачунара је ефикасно елиминисао проблеме који су узајамно симетрични. Доказивач теорема Godzila, који су креирали Норико Араи (Noriko Arai) и Ружи Масукава (Ryuji Masukawa) [2], се користи за брзо проналажење симетрија у комбинаторним проблемима. Марко Кадоли (Marco Cadoli) и Тони Манцини (Toni Mancini) [21, 58] су разматрали поставке комбинаторних проблема као логичке формуле и користили су аутоматске доказиваче теорема за препознавање симетрија и функционалних зависности.

Шанг-Чинг Чу (Shang-Ching Chou), Киао-Шан Гао (Xiao-Shan Gao) и Јинг-Зонг Занг (Jing-Zhong Zhang) су развили метод заснован на базама података [28] за откривање и доказивање нетривијалних теорема геометрије. Приметили су да већина геометријских предиката (као што је на пример колинеарност) задовољава својства транзитивности и симетричности, због чега се јавља значајно увећање базе података и дуплирање информација. Коришћењем класа еквиваленције и канонских представника за представљање чињеница у бази података (на начин сличан оном који је описан у овој тези), успели су да смање величину базе података за фактор 100.

Рикардо Кафера (Ricardo Caferra), Николас Петиер (Nicolas Peltier) и Франсоа Пуит (Francois Puitg) [22] су користили сличан приступ приликом доказивања теорема геометрије. Имплементирали су технике налик оним којим се математичари служе у уџбеницима, поправили ефикасност свог доказивача и омогућили једноставнију комуникацију између доказивача и корисника. У оквиру алгоритма унификације су имплементирали теорије једнакости, комутативности и симетричности. Захваљујући томе, у бази знања се чува само минимални представник класе еквиваленције чиме се добија значајно убрзање.

Мејкл и Флорио [63] су користили полу-аутоматски приступ[12] за доказивање теорема геометрије. У оквиру система Isabelle/HOL су аутоматизовали део геометријског резоновања (као што је на пример симетричност колинеарности) које се често користи приликом формалног доказивања проширивањем алата за поједностављивање и класично резоновање у Isabelle-у. Том приликом су дошли до закључка да коришћење теорема које обезбеђују симетричност предиката може довести до појављивања бесконачних петљи, и да је потребно посветити пажњу таквим проблемима. Техника за рад са симетричним предикатима,

[12]Под полу-аутоматским доказивањем теорема сматрамо коришћење техника за аутоматизацију које су имплементиране у оквиру доказивача за интерактивно доказивање теорема. У току интерактивног доказивања теорема, корисник има могућност коришћења аутоматизације ако сматра да му је таква помоћ потребна у случајевима да те технике успешно докажу формулисане подциљеве.

описана у овој тези, не допушта коришћење лема које обезбеђују симетричност у току рада самог доказивача. Те леме ће бити коришћене само у фази комплетирања доказа на местима на којима су коришћене информације о симетричности предиката.

3.6 ArgoCLP Prover and its Extensions — Summary

This section presents a coherent logic prover ArgoCLP [83] that is based on a simple proof procedure with forward chaining and iterative deepening. ArgoCLP assumes the coherent form of all formulae and that there are no function symbols or arity greater than 0, and can read an input theory and the conjecture given in the TPTP form[13] [87] (or in a form specially designed for ArgoCLP prover), and can export proofs to a custom XML representation (as well as directly into language of theorem prover Isabelle and in English). The XML proof representation can further be translated, by a simple XSL stylesheet, to Isabelle/Isar proofs, Coq proofs, and proofs in natural language (English and Serbian) formatted in LATEX or in HTML [82].

The basic proof procedure of the prover applies the axioms in the waterfall manner: axioms are tested in the order they are given; when one axiom has been successfully applied, then search for applicable axioms starts again from the first axiom. Thus, ordering of axioms within a theory can have significant impact on the efficiency of the prover. Within the prover, axioms can be grouped in the following groups and used with the priorities as in the following ordering[14]:

non-productive non-branching axioms: axioms of the form:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B(\vec{x})$$

non-productive branching axioms: axioms of the form:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B_1(\vec{x}) \vee \ldots \vee B_m(\vec{x})$$

productive non-branching axioms: axioms of the form:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}\, B(\vec{x}, \vec{y})$$

productive branching axioms: axioms of the form:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1\, B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\, B_m(\vec{x}, \vec{y}_m)$$

[13] http://www.cs.miami.edu/~tptp/

[14] It is assumed that $n \geq 0$ ($n > 0$ for the third and the fourth group), $m > 1$, and that one group of axioms excludes previous groups that are its special cases.

strongly productive non-branching axioms: axioms of the form:

$\exists \vec{y}\, B(\vec{y})$

strongly productive branching axioms: axioms of the form:

$\exists \vec{y}_1\, B_1(\vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\, B_m(\vec{y}_m)$

There are several other methods for improving efficiency of the prover such as: early rejection of some axiom instances, dealing with equality, restriction on branching axioms and restriction on axioms (regarding predicates that are used within a conjecture). The user can state (through a configuration file) which of the techniques should be used in the proof search.

Two types of preprocessing techniques of the axiomatic system are presented, one concerning symmetric predicates, and another restricting introduction of witnesses during proof search. These techniques were inspired by the common problem in automated and interactive theorem proving, that is coping with „simple facts". In traditional pen-and-paper theorem proving, simple facts (trivial facts that don't need deep proofs and theory-specific arguments) are typically assumed or neglected. However, in automated theorem proving they often significantly increase the search space, while in interactive theorem proving they complicate the proof and often overwhelm the main line of the proof. Both techniques were used within a prover ArgoCLP thus making it more efficient, producing significantly shorter generated proofs and (for the proofs written in natural language) more similar to the mathematical textbook proofs.

4

Дијалект кохерентне логике

У овој глави биће представљен дијалект кохерентне логике, као и математички дијалект који је био инспирација приликом његовог креирања.

4.1 Математички дијалект

Герхард Генцен је 1934-те увео рачуне природне дедукције и рачуне секвената за класичну и интуиционистичку логику [38, 88]. У његовој дисертацији [38] Генцен каже:

> *Циљ ми је био креирање формализма који би био што ближи људском резоновању. Отуда је настао „рачун природне дедукције".*

Данас су правила Генценових рачуна у основи многих система за интерактивно доказивање теорема.

Термин *математички дијалект* је 1980-е увео Николас Говерт де Бруин (Nicolaas Govert de Bruijn) приликом представљања формализма за запис математичког речника [29]. Након тога је неколико различитих аутора модификовало или надградило де Бруинов рад. Фрик Вејдик (Freek Wiedijk) је пратио де Бруинову мотивацију [97], али је такође приметио да:

> *Велики број интерактивних доказивача теорема користи језике који веома личе једни на друге. Намеће се закључак да постоји један униформни начин записивања математике до ког су различите групе људи дошле независно: нешто налик* природном *математичком дијалекту. Како је развој, различитих али суштински веома сличних облика, овог дијалекта текао независно можемо га звати* математичким дијалектом.

Математички дијалект [97] који је идентификовао Вејдик представља заједнички чинилац између језика које користе доказивачи *Hyperproof*, *Mizar* и *Isabelle/Isar*. Иако су суштински креирани независно један од другог, скоро сви су у значајној мери, базирани на природној дедукцији. Вејдиков дијалект није до сада имплементиран, иако су Hyperproof, Mizar и Isabelle/Isar развијени користећи сличне идеје.

4.2 Репрезентација доказа у кохерентној логици

У овом поглављу биће представљена нова, једноставна а веома изражајна репрезентација доказа која је заснована на кохерентној логици и користи само неколико правила извођења. Креирана је тако да омогућава једноставно генерисање доказа у језицима различитих интерактивних доказивача теорема, као и за генерисање доказа на природном језику. Језик описан том репрезентацијом назваћемо *дијалект кохерентне логике (Coherent Logic Vernacular)*.

Потреба за оваквом репрезентацијом доказа долази из чињенице да постоји неколико веома добро развијених и популарних интерактивних доказивача теорема (као што су Isabelle, Coq, Mizar, HOL-light, преглед популарних доказивача може се наћи у [98]) који развијају махом одвојене формализације различитих (а некад и истих) теорија. Формализације које се креирају на језику једног од ових доказивача, у општем случају не могу бити тривијално преписане на језик неког другог доказивача. Како скуп таквих формализација расте, јавља се потреба за превођењем између језика различитих интерактивних доказивача теорема. Овакав задатак није ни најмање једноставан из разлога што ће сваки интерактивни доказивач теорема имати неке специфичности. Упркос томе, постоје нови обећавајући приступи овом проблему [48]. Размена доказа између различитих доказивача теорема могућа је коришћењем дубоког или плитког утапања (*deep or shallow embeddings*) [70, 53]. Метју Бозфлаг (Mathieu Boespflug), Квентин Карбону (Quentin Carbonneaux) и Оливије Херман (Olivier Hermant) предлажу коришћење $\lambda\Pi$-рачуна као универзалног језика за представљање доказа [13].

Уместо креирања алата за превођење између језика различитих интерактивних доказивача, у овој тези је представљен нови начин записивања доказа у оквиру кохерентне логике као и одговарајућа репрезентација доказа записана у XML-у. Основни облик репрезентације доказа се може, уз помоћ неколико XSL алата, трансформисати у доказе записане на језицима за

формално доказивање теорема Isabelle/Isar[1] и Coq[2], као и у доказе записане на природном језику (који могу бити форматирани у LaTeX-у или у HTML-у). Репрезентација доказа која је овде представљена је релативно једноставна и нема за циљ да покрије све врсте различитих доказа који се могу јавити у математици, нити у логици првог реда. И поред тога, ова репрезентација доказа је довољно изражајна да се у њој могу записати многе занимљиве математичке теорије. Осим тога, креирање доказа у овом формату је релативно једноставно за кохерентне доказиваче, али може се користити и од стране стандардних доказивача теорема. Креирање доказа у овом формату је подржано у кохерентном доказивачу ArgoCLP. С обзиром да докази записани у овом формату имају могућност превођења у доказе записане у језицима различитих интерактивних доказивача теорема, тиме је омогућено дељење добијене формализације између различитих доказивача.

За разлику од правила које је увео Безем [9] наш систем ће бити секвентни систем, са правилима извођења за секвенте.

У правилима за секвенте користимо ознаке:

- $ax \in AX$ је формула $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}(B_1(\vec{x},\vec{y}) \vee \ldots \vee B_m(\vec{x},\vec{y}))$, $0 \leq n$, $0 \leq m$, $A_i(\vec{x})$ су атомичке формуле, $B_i(\vec{x},\vec{y})$ су конјункције атомичких формула;
- $\vec{x}$ и $\vec{y}$ означавају векторе променљивих (дужине веће или једнаке 0);
- $A_i(\vec{x})$, $B_i(\vec{x},\vec{y})$ и $B_i(\vec{y})$ не садрже слободне променљиве осим оних које се налазе у $\vec{x}$ и $\vec{y}$;
- $\vec{a}$, $\vec{b}$ и $\vec{c}$ означавају векторе константи (дужине веће или једнаке 0);
- у оквиру правила *emp*, $\vec{b}$ представља нове константе;
- $A_i(\vec{a})$ су чињенице;
- $B_i(\vec{a},\vec{b})$ и $B_i(\vec{c})$ су конјункције чињеница;
- $\underline{\Phi}$ означава листу конјуката који припадају формули (конјункцији) Φ;
- Γ означава скуп кохерентних формула (скуп аксиома теорије и текући скуп изведених чињеница).

[1] Алат за трансформисање репрезентације доказа у језик интерактивног доказивача Isabelle је развијен у сарадњи са Филипом Марићем.

[2] Алат за трансформисање репрезентације доказа у језик интерактивног доказивача Coq је креирао Жилиен Нарбу.

Правила извођења:

$$\frac{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})}, \underline{B_1(\vec{a}, \vec{b})} \vdash P}{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})} \vdash P} \; emp \; (extended \; modus \; ponens, \; m = 1)$$

$$\frac{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})}, B_1(\vec{a}, \vec{b}) \vee \ldots \vee B_m(\vec{a}, \vec{b}) \vdash P}{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})} \vdash P} \; emp \; (m > 1)$$

$$\frac{\Gamma, \underline{B_1(\vec{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_m(\vec{c})} \vdash P}{\Gamma, B_1(\vec{c}) \vee \ldots \vee B_m(\vec{c}) \vdash P} \; cs \; (case \; split)$$

$$\frac{}{\Gamma, \underline{B_i(\vec{a}, \vec{b})} \vdash \exists \vec{y}(B_1(\vec{a}, \vec{y}) \vee \ldots \vee \; B_m(\vec{a}, \vec{y}))} \; as \; (assumption)$$

$$\frac{}{\Gamma, \bot \vdash P} \; efq \; (ex \; falso \; quodlibet)$$

Примена правила у процесу доказивања одговара читању уназад, тј. применом правила добија се садржај изнад црте. Ако је дат скуп кохерентних аксиома AX и теорема $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}(B_1(\vec{x}, \vec{y}) \vee \ldots \vee \; B_m(\vec{x}, \vec{y}))$ коју треба доказати, онда је у нашем систему потребно извести наредни секвент:

$$AX, A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a}) \vdash \exists \vec{y}(B_1(\vec{a}, \vec{y}) \vee \ldots \vee \; B_m(\vec{a}, \vec{y}))$$

Наведени секвент доказује се применом правила извођења која су наведена, где су вектором $\vec{a}$ уведени нови симболи константи.

Примена ових правила не мења циљ P који се доказује, чиме је омогућено једноставно генерисање читљивих доказа. Правило *emp* (*extended modus ponens*) комбинује неколико различитих правила: елиминацију универзалних квантификатора, увођење конјункције, елиминацију импликације и елиминацију (једног или више) егзистенцијалних квантификатора. Таква комбинација омогућава да кораци доказа буду не превише једноставни а да комплетан доказ буде читљив. Постоје различита уверења о томе шта су „очигледна правила" извођења [77], али горе наведена правила су не само уобичајена у математичким доказима, већ су и погодна за аутоматизацију у оквиру доказивача за кохерентну логику. У поређењу са правилима датим у [9], у овој тези се одваја правило *case split* (елиминација дисјункције) и

ex falso quodlibet правило од једног комбинованог правила [9], због повећања читљивости добијеног доказа. Правило гранања (split правило) је битан део структуре доказа, који заслужује да буде експлицитно наведен. Правило *ex falso quodlibet* се може посматрати као *case split* са нула случајева, али на тај начин би добили мање читљиве доказе.

Сваки доказ у оквиру кохерентне логике може бити представљен на следећи начин (*emp* правило се користи нула или више пута, *cs* правило садржи макар два објекта типа *proof*):

$$proof \ ::= \ emp^* \ (cs(proof^{\geq 2}) \mid as \mid efq)$$

4.3 Алат за подршку дијалекту кохерентне логике

За запис дијалекта кохерентне логике коришћен је проширив језик за означавање текстуалних докумената XML (енг. EXtensible Markup Language)[3]. XML представља једноставан и флексибилан језик инспирисан SGML-ом (енг. Standard Generalize Markup Language, ISO 8879), који се употребљава за означавање текстуалних докумената коришћењем етикета (енг. tag), као и за размену информација између различитих информационих система. XML је примарно „метајезик" — језик који се користи за описивање других језика за означавање, није фиксан језик и нема фиксан скуп етикета које може да користи (за разлику од језика за означавање HTML). У оквиру XML-а, етикете означавају семантичку структуру документа. Прописан је од стране WWW Конзорцијума (енг. World Wide Web Consortium, скраћено W3C) и подржан је од стране већине актуелних интернет прегледача.

Постоји неколико језика који служе за представљање структуре и садржаја једног типа XML докумената. Најпопуларнији су дефиниција типа документа DTD (енг. Data Type Definition), XML Schema, Relax, etc.[54]. Спецификације дате једним од ових језика омогућавају аутоматску верификацију (валидацију) да ли дати документ задовољава синтактичке рестрикције које су задате.

Језик за трансформисање XSLT (енг. Extensible Style-sheet Language Transformation)[4] је језик за обраду докумената који се користи за трансформисање XML докумената у излазне документе. XSLT документ се састоји од низа правила (енг. templates) које XSLT процесор користи приликом

[3] http://www.w3.org/XML/
[4] http://www.w3.org/Style/XSL/

интерпретирања садржаја улазног XML документа. Ова правила говоре XSLT процесору на који начин треба да прикаже податке: као XML документ, HTML документ, обичан текст или у неком другом формату.

Репрезентација доказа у оквиру кохерентне логике описана у поглављу 4.2 ће представљати основу XML репрезентације доказа. Развијена је са циљем да представља везу између аутоматских доказивача теорема и интерактивних доказивача теорема. Формални докази (записани у језицима Isabelle/Isar и Coq) добијени на основу XML репрезентације доказа који ће бити представљен у овој тези су веома читљиви. Сам XML документ је такође читљив за човека али је обезбеђена алтернатива у виду доказа записаних у природном језику (форматирани, на пример, у LaTeX-у). У наставку је дат део DTD документа `Vernacular.dtd` који описује репрезентацију доказа у кохерентној логици.

```
...
<!--******** Theory **************-->
<!ELEMENT theory (theory_name, signature, axiom*) >
<!ELEMENT theory_name (#PCDATA)>
<!ELEMENT signature (type*, relation_symbol*, constant*) >
<!ELEMENT relation_symbol (type*)>
<!ATTLIST relation_symbol name CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT axiom (cl_formula)>
<!ATTLIST axiom name CDATA #REQUIRED>
...
```

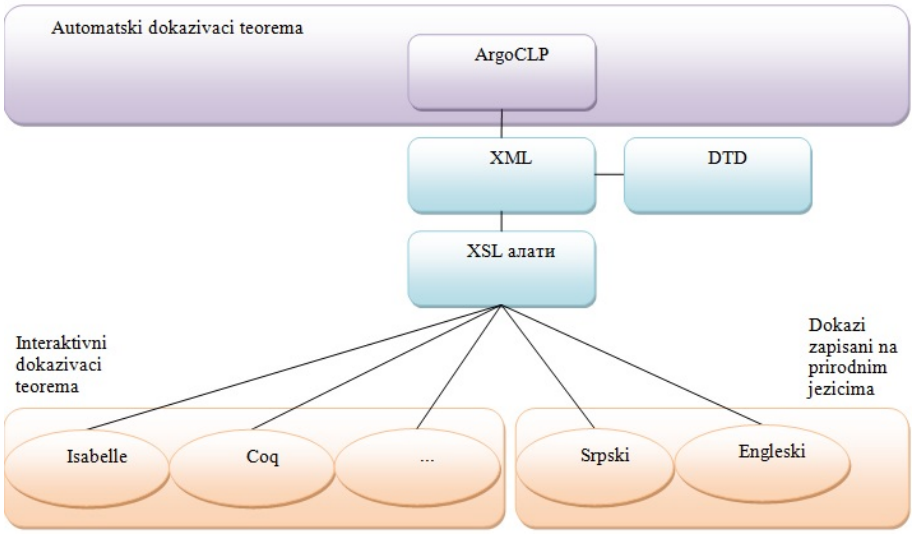
Овај део DTD документа описује појам теорије. Дефиниције, записане кроз парове кохерентних формула, користе се као аксиоме. Документ који садржи опис конкретне теорије може бити заједнички за више различитих докумената који описују конкретне теореме и доказе.

```
...
<!--******** Theorem **************-->
<!ELEMENT theorem (theorem_name, cl_formula, proof+)>
<!ELEMENT theorem_name (#PCDATA)>
<!ELEMENT conjecture (name, cl_formula)>

<!--******** Proof **************-->
<!ELEMENT proof (proof_step*, proof_closing, proof_name?)>
<!ELEMENT proof_name EMPTY>
<!ATTLIST proof_name name CDATA #REQUIRED>

<!--******** Proof steps **************-->
<!ELEMENT proof_step (indentation,modus_ponens)>
```

```
<!ELEMENT proof_closing (indentation, (case_split|efq|from),
  (goal_reached_contradiction|goal_reached_thesis))>
...
```

Овај део документа описује појам теореме и доказа. Како је описано у поглављу 4.2, доказ се састоји од низа примена правила extended modus ponens и завршава се применом једног од наредних правила: (*case split*, *as*, или *efq*). У оквиру правила којим се завршава доказ, постоји додатна информација о томе да ли је доказ комплетиран контрадикцијом ($\bot$) или проналажењем једног од дисјунката циља теореме. Ову информацију генерише аутоматски доказивач теорема и она служи повећању читљивости добијеног доказа, али се такође може користити у циљу лакше трансформације доказа. У оквиру сваког корака доказа чува се и податак о угнежденост корака. Ова информација се користи за лепши приказ самог доказа јер носи информацију о нивоу поддоказа.

Креирано је неколико XSL докумената за трансформисање добијеног XML документа у доказе записане на језицима интерактивних доказивача теорема Isabelle/Isar (`VernacularISAR.xls`) и Coq (`VernacularCoqTactics.xls`), као и у доказе записане на природном језику (енглеском и српском) форматиране у LaTeX-у и у HTML-у (`VernacularTex.xls`, `VernacularSrpskiTex.xls` и `VernacularHTML.xls`). Архитектура представљеног система је приказана на слици 4.1.



Слика 4.1: *Архитектура представљеног система*

Превођење XML документа у доказ записан у Isar језику је релативно једноставно јер се сваки корак XML документа тривијално преводи у одгова-

рајући Isar корак.[5] Приликом записивања формалних доказа у Isar-у (и Coq-у) користи се „обична" негација, уместо форме негације која се користи у оквиру кохерентне логике.

Превођење XML документа у доказ записан у језику доказивача Coq је урађено налик превођењу у Isar језик, упркос томе што је коришћење тактика у оквиру доказивача Coq много популарније од декларативног начина писања доказа. У оквиру Coq доказа приликом реферисања на направљене претпоставке користе се саме претпоставке уместо њихових имена (на пример: `by cases on (A = B \/ A <> B)`). Додатно, у случајевима када је то могуће, претпоставке уопште неће бити наведене. У циљу побољшања читљивости доказа креиране су нове тактике које имплементирају правила извођења кохерентне логике. Резоновање унапред се изводи применом `assert` тактике. Једнакост се записује као Лајбницова (Leibniz) једнакост.

Превођење у LaTeX и HTML укључује додатну XSLT датотеку у којој се може дефинисати специфичан начин записа за одређене релационе симболе (на пример, $(A, B) \cong (C, D)$ може бити коришћено за записивање релације `cong(A,B,C,D)`).

XSLT документи који су креирани су релативно једноставни и кратки — сваки има око 500 редова. На основу тога може се закључити да би и превођење у друге формате (језике других доказивача теорема као што су Mizar и HOL light, LaTeX формат за друге природне језике, MathML, OMDoc или TPTP) било релативно једноставно, чиме би се омогућило коришћење униформног формата за записивање математичких садржаја.

Кохерентни доказивач ArgoCLP, описан у овој тези, генерише излазне доказе у XML формату који је у складу са описаним DTD документом. Генерисани XML документ је релативно једноставан и састоји се од три дела: `frontpage` (у ком се налазе подаци о аутору теореме, доказивачу који се користи и датум креирања документа), `theory` (у ком се налази сигнатура теорије и аксиоме које су коришћене) и листа тврђења или теорема (потенцијално организованих по поглављима) заједно са њиховим доказима. Овако креирани XML документи могу делити заједничке `frontpage` и `theory` датотеке и могу бити коришћени за чување већих скупова теорема. У наставку текста је приложен XML документ креиран од стране доказивача ArgoCLP за једну теорему.[6]

```
<?xml version="1.0" encoding="UTF-8"?>
```

[5] Иако систем Isabelle располаже алатом за доказивање `coherent` који користи интерни доказивач теорема базиран на кохерентној логици, тај алат се не користи приликом превођења доказа из XML формата у Isar језик.

[6] Комплетан алат за рад са овим XML форматом, заједно са скупом теорема, се налази на адреси http://argo.matf.bg.ac.rs/downloads/software/clvernacular.zip.

```
<!DOCTYPE main SYSTEM "Vernacular.dtd">
<?xml-stylesheet href="VernacularISAR.xsl" type="text/xsl"?>

<main>
<xi:include href="frontpage.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="theory_thm_4_19.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>

<chapter name="th_4_19">
<xi:include href="proof_thm_4_19.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
</chapter>
</main>
```

У случају да се систем користи за генерисање доказа теорема неке теорије, може се креирати XML документ који ће садржати доказе свих појединачних теорема. У наставку текста се налази део таквог документа.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE main SYSTEM "Vernacular.dtd">
<?xml-stylesheet href="VernacularISAR.xsl" type="text/xsl"?>

<main>
<xi:include href="frontpage.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="theory.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>

<chapter name="Consequences of the Axioms A1-A5">
<xi:include href="proof_th_2_1.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="proof_th_2_2.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
...
</chapter>

<chapter name = "Simple Properties of Betweenness">
<xi:include href="proof_th_3_1.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="proof_th_3_2.xml" parse="xml" xmlns:xi="http://www.w3.org/2003/XInclude"/>
...
</chapter>
...
</main>
```

4.4 Пример генерисаних доказа

XML формат и скуп алата који је креиран за рад са њим је коришћен над низом теорема и њихових доказа генерисаним од стране доказивача ArgoCLP. У овом поглављу биће приказани докази теореме из књиге *Metamathematische Methoden in der Geometrie*, чији су аутори Волфрам Швабхојзер (Wolfram Schwabhäuser), Ванда Шмилев и Алфред Тарски [79], која представља једну од класичних математичких књига двадесетог века. Теорија која се користи је описана у поглављу 2.1. Већина теорема које се налазе у овој књизи припадају кохерентној логици или могу бити тривијално трансформисане тако да припадају кохерентној логици. Након извођења потребних трансформација,

број кохерентних теорема који је добијен (238) је нешто већи од броја теорема које су дате у књизи [86].

У наставку текста је приказан доказ једне од теорема (4.19) из књиге Тарског. Теорема је доказана помоћу доказивача ArgoCLP (коме је задата листа релевантних аксиома и теорема добијена након интеракције са резолуцијским доказивачима), генерисани доказ је записан у XML формату и након тога је преведен у доказ у природном језику коришћењем одговарајуће XSL трансформације (приликом записивања доказа у природном језику користи се инфиксна нотација $(A, B) \cong (C, D)$ предикатског симбола $cong(A, B, C, D)$ и означава да је пар тачака (A, B) подударан пару тачака (C, D), $bet(A, B, C)$ означава да се тачка B налази између тачака A и C, $col(A, B, C)$ означава да су тачке A, B и C колинеарне).

Teorema 4.1 (th_4_19) *Assuming that* $bet(A, B, C)$ *and* $AB \cong AD$ *and* $CB \cong CD$ *it holds that* $B = D$.

Proof:

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

У наставку је приказан доказ исте теореме записан у језику интерактивног доказивача Isabelle:

```
lemma th_4_19 : assumes "bet A B C" and "cong A B A D" and "cong
C B C D" shows "(B = D)"
proof -
have "bet B A A" by (rule th_3_1)
from ‘bet A B C‘ have "col C A B" by (rule ax_4_10_3)
from ‘cong A B A D‘ have "cong A D A B" by (rule th_2_2)
have "A = B ∨ A ~= B" by (subst disj_commute, rule excluded_middle)
   show ?thesis
   proof(cases "A = B")
     case True
      from ‘cong A D A B‘ and ‘A = B‘ have "cong A D A A" by simp
      from ‘cong A D A A‘ have "A = D" by (rule ax_3)
      from ‘A = B‘ and ‘A = D‘ have "B = D" by simp
      from ‘B = D‘ show ?thesis by assumption
   next
     case False
      have "A = C ∨ A ~= C"
      by (subst disj_commute, rule excluded_middle)
         show ?thesis
         proof(cases "A = C")
           case True
            from ‘bet A B C‘ and ‘A = C‘ have "bet A B A" by simp
            from ‘bet A B A‘ and ‘bet B A A‘ have "A = B"
            by (rule th_3_4)
            from ‘A ~= B‘ and ‘A = B‘ have "False" by (rule notE)
            from this show ?thesis by (rule FalseE)
         next
           case False
            from ‘A ~= C‘ have "C ~= A" by (rule not_sym)
            from ‘C ~= A‘ and ‘col C A B‘ and ‘cong C B C D‘ and
            ‘cong A B A D‘ have "B = D" by (rule th_4_18)
            from ‘B = D‘ show ?thesis by assumption
         qed
   qed
qed
```

У наставку је приказан доказ исте теореме записан у језику интерактивног доказивача Coq:

```
Theorem th_4_19 : ∀ (A:point) (B:point) (C:point) (D:point), (bet A B C ∧
cong A B A D ∧ cong C B C D) → B = D.
Proof.
 intros.
 assert (bet B A A) by applying (th_3_1 B A ) .
 assert (col C A B) by applying (ax_4_10_3 A B C ) .
 assert (cong A D A B) by applying (th_2_2 A B A D ) .
 assert (A = B ∨ A ≠ B) by applying (ax_g1 A B ) .
 by cases on (A = B ∨ A ≠ B).
 - {
    assert (cong A D A A) by (substitution).
    assert (A = D) by applying (ax_3 A D A ) .
    assert (B = D) by (substitution).
    conclude.
    }
 - {
    assert (A = C ∨ A ≠ C) by applying (ax_g1 A C ) .
    by cases on (A = C ∨ A ≠ C).
    - {
       assert (bet A B A) by (substitution).
       assert (A = B) by applying (th_3_4 A B A ) .
       assert (False) by (substitution).
       contradict.
       }
    - {
       assert (C ≠ A) by (substitution).
       assert (B = D) by applying (th_4_18 C A B D ) .
       conclude.
       }
    }
Qed.
```

Од скупа теорема записаних у кохерентној логици (238), доказивач ArgoCLP је потпуно аутоматски доказао 85 (36%) теорема и изгенерисао њихове доказе у XML формату. Након тога је креиран један заједнички XML документ у коме се налазе све доказане теореме са својим доказима и недоказане теореме означене

као претпоставке (енг. conjectures). Комплетан документ одговара оригиналној књизи и може бити експортован у LaTeX (или PDF) формат, HTML, Isabelle или Coq формат.[7]

4.5 Coherent Logic Vernacular — Summary

This chapter introduces a new vernacular designed for coherent logic and a proof representation that describes it. It is not intended to serve as "the mathematical vernacular", but it can cover a significant portion of many interesting mathematical theories. The presented proof representation is very simple, yet expressive enough and suitable for many everyday mathematical developments.

Its development is motivated by the existence of several very mature and popular interactive theorem provers (including Isabelle, Coq, Mizar, HOL-light) that have large repositories of formalized mathematics but still cannot easily share the same mathematical knowledge. There are recent efforts to build a mechanism for translation between different proof assistants, but it is non-trivial because of many deep specifics of each proof assistant.

Coherent logic vernacular presented in this paper is inspired by the mathematical vernacular proposed by Wiedijk [97] that is in a sense the common denominator of the proof languages of Hyperproof, Mizar and Isabelle/Isar.

Using a proof representation such as this one will facilitate exporting readable mathematical knowledge from automated theorem provers to interactive theorem provers. Automated theorem provers now don't have to deal with specifics of proof assistants since the task of generating object-level proofs for proof assistants (or proofs expressed in natural language) is removed from theorem provers (where it would be hard-coded). With the use of simple interchange XML format, that is delegated to a set of XSLT style-sheets, formal proofs (and natural language proofs) are generated. This feature gives flexibility to the vernacular and enables use of additional target formats (by creating additional XSLT style-sheets) without changing the prover.

The proposed proof representation uses only a few inference rules, a variant of the rules given by Bezem [9]. In the presented rules we assume:

- $ax \in AX$ is a formula of the form $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}(B_1(\vec{x},\vec{y}) \vee \ldots \vee B_m(\vec{x},\vec{y}))$, $0 \leq n$, $0 \leq m$, $A_i(\vec{x})$ are atomic formulae, $B_i(\vec{x},\vec{y})$ are conjunctions

[7] Превођење комплетног XML документа који садржи 85 доказа у формате Isabelle, Coq, HTML, LaTeX (и након тога у PDF) све заједно траје око 20s на рачунару PC са процесором AMD Opteron 6168. Генерисани Isabelle документ се верификује за 30s, а Coq документ за 6s.

of atomic formulae;

- $\vec{x}$ and $\vec{y}$ denote vectors of variables (possibly of length zero);
- $A_i(\vec{x})$, $B_i(\vec{x},\vec{y})$ and $B_i(\vec{y})$ have no free variables other than those from $\vec{x}$ и $\vec{y}$;
- $\vec{a}$, $\vec{b}$, $\vec{c}$ denote vectors of constants (possibly of length zero);
- in the rule *emp*, $\vec{b}$ are fresh constants;
- $A_i(\vec{a})$ are ground atomic formulae;
- $B_i(\vec{a},\vec{b})$ and $B_i(\vec{c})$ are conjunctions of ground atomic formulae;
- $\underline{\Phi}$ denotes the list of conjuncts in Φ;
- Γ denotes a set of coherent logic formulae.

The rules that are used are:

$$\frac{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})}, \underline{B_1(\vec{a},\vec{b})} \vdash P}{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})} \vdash P}\ emp\ (\textit{extended modus ponens},\ m = 1)$$

$$\frac{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})}, B_1(\vec{a},\vec{b}) \vee \ldots \vee B_m(\vec{a},\vec{b}) \vdash P}{\Gamma, ax, \underline{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})} \vdash P}\ emp\ (m > 1)$$

$$\frac{\Gamma, \underline{B_1(\vec{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_m(\vec{c})} \vdash P}{\Gamma, B_1(\vec{c}) \vee \ldots \vee B_m(\vec{c}) \vdash P}\ cs\ (\textit{case split})$$

$$\frac{}{\Gamma, \underline{B_i(\vec{a},\vec{b})} \vdash \exists\vec{y}(B_1(\vec{a},\vec{y}) \vee \ldots \vee\ B_m(\vec{a},\vec{y}))}\ as\ (\textit{assumption})$$

$$\frac{}{\Gamma, \bot \vdash P}\ efq\ (\textit{ex falso quodlibet})$$

Given a set of coherent axioms AX and a coherent conjecture $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists\vec{y}(B_1(\vec{x},\vec{y}) \vee \ldots \vee\ B_m(\vec{x},\vec{y}))$, the goal is to derive the following sequent:

$$AX, A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a}) \vdash \exists\vec{y}(B_1(\vec{a},\vec{y}) \vee \ldots \vee\ B_m(\vec{a},\vec{y}))$$

The sequent is proved using the given rules, while $\vec{a}$ denotes a vector of new symbols of constants. Presented rules do not change the goal P which is kept implicit thus enabling generation of readable proofs. The rule *emp* actually combines universal instantiation, conjunction introduction, extended modus ponens, and elimination of (zero or more) existential quantifiers. This seems to be a reasonable granularity for an inference step, albeit probably the maximum one for keeping proofs readable. Case distinction (split) is an important way of structuring proofs that deserves to be made explicit.

Any coherent logic proof can be represented in the following simple way (*emp* is used zero or more time, *cs* involves at least two other *proof* objects):

$$proof \ ::= \ emp^* \ (cs(proof^{\geq 2}) \mid as \mid efq)$$

The generated proofs, written in XML proof representation, are simple and can be divided into three parts: *frontpage*, *theory*, and (organized in chapters) *a list of conjectures or theorems* with their proofs. The *frontpage* file can provide the author of the theorem, the prover used for generating the proof, and the date; while the *theory* file can provide the signature and the axioms of the theory. This way, those two files can be shared by a number of XML documents enabling a simple construction of bigger collections of theorems.

XSL transformations are implemented from XML format to Isabelle/Isar (VernacularISAR.xls), Coq (VernacularCoqTactics.xls), and to a natural language (English and Serbian) in LATEX form and in HTML form (VernacularTex.xls, VernacularSrpskiTex.xls and VernacularHTML.xls). The developed XSLT style-sheets are rather simple and short — each only around 500 lines long. This shows that transformations for other target languages (other theorem provers, like Mizar and HOL light, LATEX with other natural languages, MathML, OMDoc or TPTP) can easily be constructed, thus enabling wide access to a single source of mathematical content.

5

Систем за аутоматску и интерактивну формализацију геометрије

Аутоматским доказивачима теорема и интерактивним доказивачима теорема се у последњих неколико година посвећује све више пажње што као резултат даје све већи број формализација значајних делова нетривијалног математичког знања. Успеси постигнути на тим пољима представљају подстицај за развијање система описаног у овој глави. Систем се може користити за потпуно аутоматску формализацију математичког знања и, у мало измењеном облику, за аутоматско проверавање неформалних доказа из математичких уџбеника. Систем генерише машински провериве доказе (за различите интерактивне доказиваче теорема) и читљиве доказе (налик доказима који се могу наћи у математичким уџбеницима).

5.1 Систем за аутоматску формализацију

Сви системи за доказивање теорема (описани у глави 2) имају своје предности и мане: интерактивни доказивачи теорема су поуздани, али ниво аутоматизације којим располажу има своја ограничења и добијени докази се не могу једноставно искористити у оквиру других формализација; резолуцијски доказивачи теорема су у потпуности аутоматски и веома ефикасни, али не генеришу машински провериве нити читљиве доказе (у неким случајевима докази се не генеришу уопште); доказивачи базирани на кохерентној логици су аутоматски и (неки од њих) генеришу како машински провериве тако и читљиве доказе, али нису довољно ефикасни да би самостално могли да се користе за

доказивање комплексних математичких теорема.

Идеја која стоји иза система описаног у овом поглављу јесте аутоматизација процеса формализације различитих математичких теорија (које припадају кохерентној логици, тј. са свим аксиомама и тврђењима записаним у оквиру кохерентне логике). Да би се то постигло биће коришћена све три врсте система за доказивање теорема поменута у претходном пасусу тако да се искористе најбоље карактеристике сваког од њих.

Систем комбинује предности и својства неколико резолуцијских доказивача (Vampire, E и Spass), аутоматског доказивача за кохерентну логику (ArgoCLP), и скупа XSL алата за рад са доказима у кохерентној логици (који се користе за превођење доказа у језике различитих интерактивних доказивача теорема као и у доказе записане на природном језику како је описано у глави 4).

5.1.1 Доказивање једног тврђења

Приликом доказивања конкретног тврђења систем на улазу очекује датотеку у TPTP формату која се састоји од свих аксиома и теорема које припадају теорији у оквиру које се доказује тврђење и које се користе у доказу тог тврђења (добијене, на пример, из математичке књиге која се формализује) . Скуп премиса који се користи може чинити целокупан скуп свих аксиома и теорема које претходе тврђењу које се доказује (у оквиру извора који се формализује), или се може састојати од неког мањег скупа релевантних формула, аксиома, теорема или лема, које се могу користити у доказу тврђења (ако је такав скуп познат кориснику). [1] Све аксиоме, теореме и тврђење које се доказује морају претходно бити записане у кохерентној форми. Након тога се извршавају наредни кораци:

1. Датотека креирана за конкретно тврђење се прво прослеђује резолуцијским доказивачима.

2. Резолуцијски доказивачи покушавају да докажу тврђење на основу скупа аксиома, дефиниција и теорема, које су задате улазном датотеком, задатих у оригиналном и обрнутом редоследу (из разлога што редослед премиса може имати утицај на процес доказивања самог доказивача).

   У случају да један или више резолуцијских доказивача докаже тврђење, бира се најмањи скуп аксиома/теорема коришћених у доказу (који је генерисан од стране резолуцијских доказивача) и користи се за поновно

[1] Овај део се раније налазио у поглављу 5.1.2.

доказивање тврђења на исти начин. Овај процес се наставља док скуп аксиома/теорема не остане непромењен између две узастопне итерације.

3. У случају када бар један од резолуцијских доказивача теорема докаже тврђење, добијен (минимални) скуп релевантних аксиома/теорема се прослеђује доказивачу ArgoCLP и, у случају да он успе да докаже тврђење, докази се генеришу у XML формату.

4. XML датотека која садржи траг доказа се, након тога, коришћењем XSL алата може превести из XML формата у доказе записане на језицима интерактивних доказивача Isabelle и Coq, као и у доказе записане на природном језику (енглеском и српском).

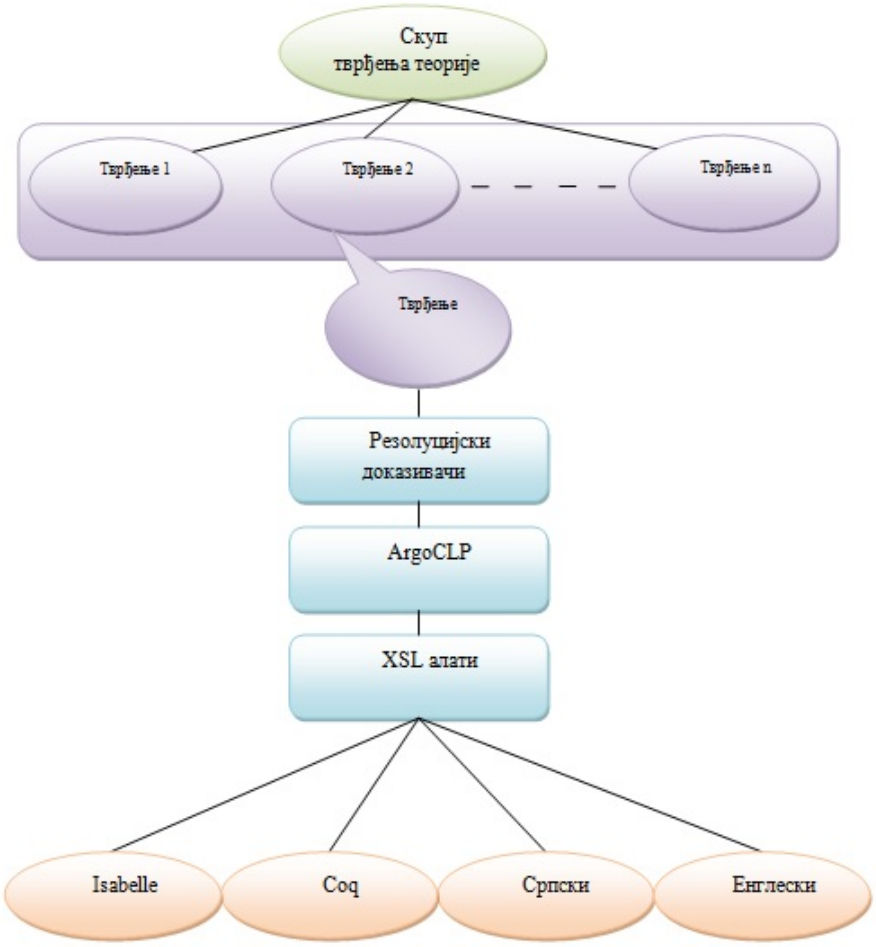
У последњем кораку процеса доказивања, скуп XSL алата се користи за генерисање машински читљивих доказа и доказа записаних на природном језику. Машински читљиви докази би се потенцијално могли конструисати и на основу трага доказа резолуцијских доказивача [10], али то није тривијалан задатак. Са друге стране коришћењем доказивача ArgoCLP, у трећем кораку, немамо гаранцију да ће свако тврђење бити успешно доказано, чак и када му се проследи скуп свих аксиома и теорема који је успешно коришћен од стране резолуцијских доказивача.

5.1.2 Доказивање више тврђења теорије

Поступак доказивања скупа тврђења неке књиге подразумева као први корак, записивање свих аксиома, дефиниција и теорема те књиге, у оквиру једне датотеке (записане у TPTP формату и у кохерентној логици). Редослед свих формула у тој датотеци одговара процесу доказивања у оквиру књиге, односно подразумева се да се формуле које се користе у доказу конкретног тврђења налазе пре њега самог. У једном пролазу кроз ту датотеку, за свако тврђење које се доказује креира се посебна TPTP датотека која се састоји од свих формула које претходе тврђењу у тој датотеци, заједно са тврђењем које се доказује. Тврђење, коме одговара тако формирана датотека, се доказује на начин описан у претходном поглављу. Архитектура описаног система је приказана на слици 5.1. Да би систем разликовао аксиоме и дефиниције од теорема које се доказују, имена свих аксиома и дефиниција почињу са "ax_", док имена свих теорема почињу са "th_".

У случају да систем докаже текуће тврђење, XML датотека која садржи траг доказа се може искористити као део већег документа који ће садржати доказе

свих теорема задате књиге као што је описано у поглављу 4.3. На тај начин овај систем омогућава генерисање једног документа који описује целокупну књигу са аутоматски генерисаним доказима.



Слика 5.1: *Архитектура система*

5.2 Систем за проверавање неформалних доказа

Из перспективе доказивања теорема уз помоћ рачунара докази наведени у математичким уџбеницима најчешће се сматрају само скицом доказа, имају јако пуно изостављених корака и не могу се једноставно верификовати уз помоћ рачунара. С друге стране, студенти и ученици средњих школа те доказе без проблема могу да разумеју. Чињеница да су ти докази преживели неколико стотина година у математичким уџбеницима говори о томе да су ти докази „довољно прецизни" за ученике и да су употребљиви у свакодневној математичкој пракси.

Осим тога, додавање детаља (које формално доказивање теорема захтева) би учинило те доказе опширнијим, тежим за праћење, и умањило би могућност ученика да активно прате и слушају предавања.

Суштински, формално доказивање теорема и доказивање теорема у образовању имају различите циљеве. Од формалних доказа се очекује максимална прецизност и несумњива исправност доказа, док се од доказа у образовању очекује да дају објашњење зашто одређено тврђење важи. У овом поглављу ће бити представљен приступ који спаја те две области и омогућава аутоматску проверу исправности неформалног доказа теореме из уџбеника, без мукотрпног задатка записивања тог доказа у оквиру програма за интерактивно доказивање теорема. Приступ ће бити имплементиран у оквиру система за проверавање неформалних доказа теорема — *ArgoGeoChecker*. Као резултат примене овог система корисник ће добити аутоматски генерисан документ који је формално проверив од стране интерактивних доказивача теорема.

Опис система Приликом доказивања теореме ученик разматра тврђење теореме и аксиоме које има на располагању, записује премисе теореме и помоћна тврђења која јасно илуструју редослед извођења у току доказа. Тако формиран скуп логичких формула звaћемо неформални доказ теореме. У неким случајевима може се десити да корисник нема доказ теореме, ако је у питању једноставна теорема или теорема коју корисник не уме да докаже. Тада се неформални доказ теореме састоји само од две формуле од којих је прва премиса теореме, а друга закључак теореме. Логичке формуле које припадају неформалном доказу теореме звaћемо кораци доказа. Систем ArgoGeoChecker ће бити коришћен за проверу неформалног доказа теореме ако је он расположив, ако постоји доказ теореме или макар упутство за доказивање. У случају да доказ није расположив, систем ће бити коришћен за проналажење доказа теореме.

Синтакса неформалног доказа Синтакса неформалног доказа је инспирисана TPTP форматом доказа и кохерентном логиком. Ако се корак доказа односи на премисе теореме, он ће бити записан са универзалним квантификатором. Ако се кораком доказа уводе нови објекти, он ће бити записан са егзистенцијалним квантификатором. А ако се кораком доказа изводи неко својство раније уведених објеката, тај корак неће бити квантификован. Формуле су записане између обичних заграда (и) (без обзира да ли су квантификоване или не). Као што је објашњено у поглављу 2.4.1 , уместо да се користе типови, уводе се додатни предикатски симболи као

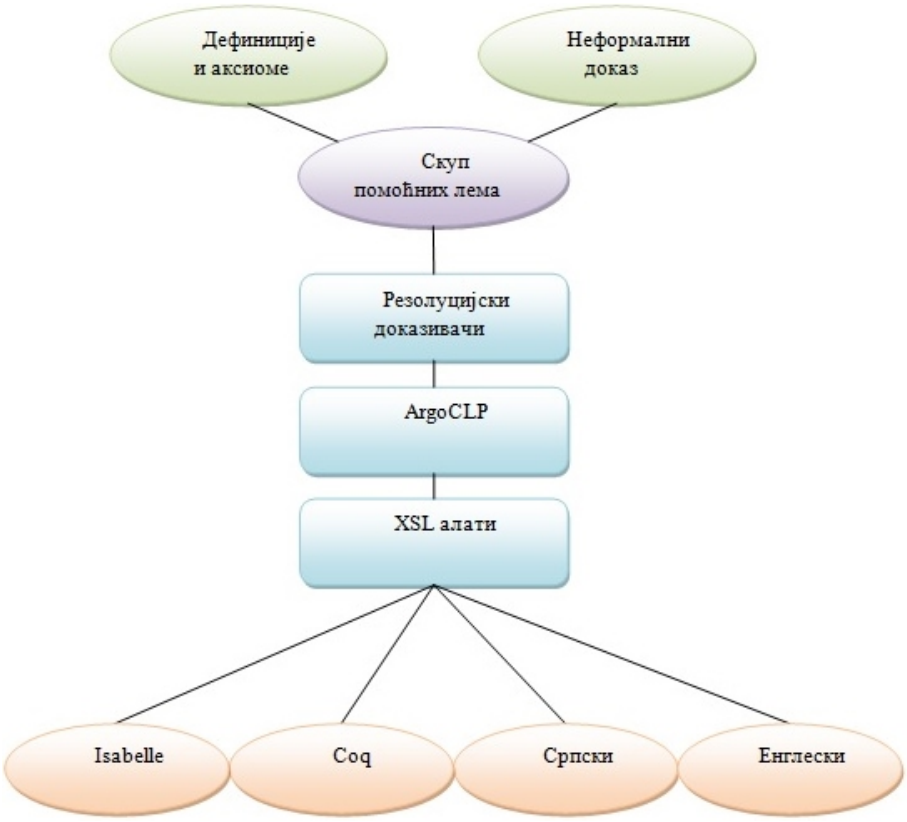
што су *point*, *line* и *plane* за запис и дефинисање типова тих променљивих. Све променљиве и сведоци се записују великим словима. Неки од примера дозвољених корака и њихов запис у оквиру система ArgoGeoChecker се налазе у табели 5.1.

Табела 5.1: *Примери дозвољених корака неформалног доказа.*

Тачка A не припада правој P.	`![P,A]:(line(P) & point(A) & ninc_po_l(A,P))`
Тачке A, B, C су неколинеарне.	`(ncol(A,B,C))`
Постоји раван R која садржи тачку A и праву P.	`?[R]:(plane(R) & inc_po_pl(A,R) & inc_l_pl(P,R))`

Архитектура система ArgoGeoChecker Приликом провере неформалног доказа теореме, систем ArgoGeoChecker проверава сваки корак доказа као засебно тврђење, односно да ли се може извести из претходних корака и скупа аксиома које је корисник навео приликом покретања програма. Скуп дефиниција и аксиома које се користе се задаје кроз засебне датотеке. За сваки корак сем првог (први корак доказа су увек премисе теореме и оне се не доказују) аутоматски се креирају помоћне леме у чијем закључку се налази текући корак а премиса је конјункција свих до тада изведених корака. Након тога се креирају помоћне датотеке (у TPTP формату) које садрже скуп дефиниција и аксиома (које се користе у доказу теореме чији се доказ проверава) и лему која се доказује. Помоћне леме се доказују на начин описан у поглављу 5.1.2. На крају процеса доказивања, формални докази помоћних лема биће обједињени у један документ који ће представљати формални аргумент исправности доказа. Тај документ може бити записан у језицима програма за интерактивно доказивање (Isabelle и Coq) и на природном језику (српском и енглеском). Архитектура система је приказана на слици 5.2.

Може се десити да, у оквиру задатог временског ограничења, систем није у стању да докаже неку од помоћних лема. У том случају формални аргумент исправности доказа ће опет бити генерисан али ће недостајати докази тих лема. Корисник тада може допунити документ који је систем генерисао или формулисати нов доказ теореме, дати детаљнију скицу доказа и покушати доказивање поново.



Слика 5.2: *Архитектура система ArgoGeoChecker*

5.3 Поређење са сродним системима

Систем за доказивање појединачних тврђења описан у овој глави близак је алату Sledgehammer, који је имплементиран у оквиру интерактивног доказивача теорема Isabelle [12, 11]. Алат Sledgehammer донео је нове могућности и нове нивое аутоматизације у интерактивно доказивање теорема. У оквиру тог алата не генеришу се читљиви докази, већ служи као помоћни алат који сугерише кориснику који скуп аксиома и теорема може бити коришћен у доказу. Осим тога, Sledgehammer се може користити и за комплетирање доказа користећи доказ побијањем добијен од стране резолуцијских доказивача [10].

За разлику од Sledgehammer алата који је подржан само у интерактивном доказивачу теорема Isabelle, систем описан у овој тези се може користити за генерисање доказа у језицима различитих интерактивних доказивача теорема, као и за генерисање доказа у читљивом облику записаним на различитим природним језицима.

Други приступ сличан систему описаном у овој глави је приступ који је направио Цезар Кализик (Cezary Kaliszyk) и Јозеф Урбан (Josef Urban) [49].

Они су истраживали колико теорема аутоматски доказивачи теорема (за логику првог реда, за логику вишег реда и за SMT проблеме) могу доказати (у сарадњи са доказивачем теорема HOL Light) од 14185 теорема које су део Flyspeck пројекта [42]. Користили су неколико аутоматских доказивача теорема са акцентом на доказивачима Vampire, E и Z3. У оригиналном приступу, зависности уочене у оквиру постојеће формализације (спискови теорема и лема које су коришћене у постојећим формалним доказима) су коришћене и аутоматским доказивачима теорема су прослеђени само скупови релевантних теорема/лема. Успешност овог приступа је 43.2% (са временским ограничењем од 900 секунди) и представља горњу границу за потпуно аутоматски приступ. У другом приступу, све теореме које претходе текућем тврђењу (посматрано хронолошки) се разматрају, и на основу скупа специјалних хеуристика (које се користе за одабир премиса) бирају се само неке од њих и прослеђују аутоматским доказивачима теорема. Успешност овог приступа је 39.5% Осим тога, Кализик и Урбан су радили и на неформалним читљивим доказима [89] и њиховој вези са формалним доказима. Радили су на доказима добијеним у оквиру Flyspeck пројекта. У оквиру овог приступа направљен је и алат за визуализацију корака доказа.

Нови аксиоматски систем *E*, описан у поглављу 2.1, намењен је доказивању постулата Еуклидових Елемената [3]. Основна идеја која стоји иза овог система је аутоматско проверавање неформалних доказа налик доказима из Елемената. Бенџамин Нортроп (Benjamin Northrop) [69] је имплементирао извођења описана тим системом[2] и креирао програм E-proof-checker[3]. Скуп аксиома система E је подељен на правила за конструисање (construction rules) и правила извођења (inference rules). Корисник на улазу задаје објекте који се користе у доказу користећи правила за конструисање објеката. Објекти се задају навођењем њихових својстава (тачка припада правој, тачка је пресечна тачка двеју правих, итд.). Након тога, примењујући правила извођења, аутоматски доказивач покушава да докаже коначно својство (описано тврђењем теореме) над до тада креираним објектима. За разлику од система описаног у овом раду, E-proof-checker нема могућност рада са другим теоријама. Креиран је искључиво за рад са Авигадовим аксиоматским системом, подржава само једну теорију и има фиксиран скуп аксиома. Генерисање формалних доказа није подржано. Колико је познато аутору, ово је једини систем (поред система описаног у овој тези) који омогућава аутоматску проверу неформалних доказа

[2] http://www.phil.cmu.edu/~avigad/formal/paris2_merged.pdf
[3] http://www.bennorthrop.com/e/e-proof-checker.php

теорема.

5.4 Framework for Formalization of Geometry - Summary

In this chapter, the framework for the formalization of mathematical knowledge is presented. The framework can produce machine verifiable proofs (for different proof assistants) but also human-readable (nearly textbook-like) proofs (for different languages). It is developed within coherent logic, i.e. all definitions, axioms and conjectures must be expressed in coherent logic.

The framework combines the power and features of several different tools: resolution theorem provers (Vampire, E, and Spass), an automated theorem prover for coherent logic (ArgoCLP), and a set of XSL tools for translation of proofs within coherent logic. The base of the framework can be used for proving individual theorems, and can be used within a larger system for proving the whole theory, or for automated verification of informal proofs (while completing a mathematical textbook in LATEX form with automatically generated proofs filled-in)

While proving a specific conjecture, framework goes through the following stages:

1. TPTP file consisting of axioms and theorems used in a proof for the conjecture is passed to resolution based theorem provers.

2. Resolution theorem provers try to prove conjecture with normal and reversed set of formulas. While proving a conjecture, resolution provers are also used to minimize set of formulas used in the proof.

3. If at least one of the resolution provers proves the conjecture, the coherent logic prover is invoked with a (minimal) set of formulas used in the proof found by resolution provers. The coherent logic prover exports a proof trace written in XML form.

4. XML proof representation is translated, using a set of XSL tools, into formal proofs written in language of interactive theorem provers Isabelle and Coq, and into natural language proofs (written in English and Serbian).

Instead of using resolution provers for reduction of set of formulas, the input premises for the coherent logic prover can be some smaller set or relevant formulae gathered from the textbook or some previous formalization.

While proving a whole theory, all axioms, definitions and conjectures must be given in order that follows the proving process, i.e. all formulas that are used in a proof of a conjecture are located before that conjecture. Conjectures within a theory are proved one by one. TPTP file is created for each conjecture with all formulas that precede the conjecture and is proven as an individual conjecture.

Modification of a system can be used for automated verification of informal proofs. The programs used for computer assisted theorem proving are steadily improving, gaining in flexibility and capability to tackle an ever-increasing set of more complex theorems. Theirs development has been guided by the notion of user-friendly interfaces and ease of use, but they still have not gained much popularity in a broad population which includes high school and university students. The system ArgoGeoChecker, presented in this thesis, aims to bridge this gap as it is specifically tailored for this audience. The systems is designed for verification of informal proofs found in mathematical geometry textbooks. Presented with a proof from a textbook, student can extract important steps and formulas used in the proof, write them in a specific format and give them to the system ArgoGeoChecker. The system will individually verify each step, generate formal and readable proof for each of them, and combine them all into one formal document that can serve as an formal justification for the given informal proof.

6

Примене система за аутоматску и интерактивну формализацију геометрије

Систем описан у претходној глави се може применити на било коју теорију записану у кохерентној логици. У овој глави биће илустрована примена тог система на три аксиоматска система еуклидске геометрије: Хилбертов аксиоматски систем, аксиоматски систем Тарског и Авигадов аксиоматски систем. Приликом рада са аксиоматским системом Тарског биће коришћен систем за доказивање целе теорије, док ће за аксиоматске системе Хилберта и Авигада биће коришћен систем за проверавање неформалних доказа.

6.1 Геометрија Тарског

Главни допринос ове тезе је аутоматска формализација једне од најзначајнијих математичких књига XX века — првог дела књиге о заснивању геометрије: *Metamathematische Methoden in der Geometrie*, чији су аутори Волфрам Швабхојзер, Ванда Шмилев и Алфред Тарски [79]. У остатку ове тезе ову књигу ћемо звати књига Тарског. Књига је настала као резултат низа аксиоматизација Тарског и приказује аксиоматски систем, као и процедуру одлучивања за ту теорију [91]. Први део књиге је настао на основу белешки са предавања Ванде Шмилев (како је приметио Бисон [5]). У њој се налазе и резултати обједињени у докторату Харагури Нарајана Гупте (Haragauri Narayan Gupta) [41]. Теорија која је приказана у овој књизи описана је у појмовима логике првог реда, користи само један примитиван објекат — тачку, има два примитивна предиката и само једанаест аксиома. У овом поглављу биће

приказано аутоматско генерисање доказа теорема књиге Тарског користећи систем описан у овој тези.

Разлози због којих је баш ова књига изабрана за аутоматску формализацију су:

- Ова књига је једна од најзначајнијих математичких књига двадесетог века.
- Све теореме из књиге су доказиве само на основу скупа почетних аксиома (тако да је довољно разматрати само тврђења која се налазе у тој књизи).
- Аксиоме које се наводе у књизи су једноставне и изражене над примитивним предикатима.
- Докази приказани у књизи су веома јасни и прецизни.
- Теорија која је описана у књизи припада логици првог реда.
- Све формуле које се налазе у књизи се могу изразити у оквиру кохерентне логике (било директно или коришћењем тривијалних трансформација) [92].
- Скуп теорема који се налази у књизи је добро заокружен скуп теорема, почевши од веома лаких до веома тешких теорема, чиме се добија нетривијалан скуп веома битних математичких теорема.
- Докази теорема ове књиге уз помоћ рачунара већ постоје. Како применом аутоматског доказивања теорема [74, 5, 6], тако и применом интерактивног доказивања теорема [67, 18]. Постојећи докази могу бити коришћени за поређење са нашим системом, али могу бити коришћени и у циљу усмеравања простора претраге у оквиру система описаног у овој тези.

У овом поглављу ће бити представљена примена система описаног у глави 5.1 на први део књиге Тарског (други део се бави питањима метаматематике). Циљ примене тог система јесте аутоматско генерисање читљивих и машински проверивих доказа теорема из књиге Тарског. Систем уједно генерише и дигиталну верзију књиге Тарског, са свим аксиомама, дефиницијама, теоремама и генерисаним доказима на природном језику. Разматраћемо, слично као у иницијалној Coq формализацији [18], само теореме које се јављају у првих 12 поглавља књиге Тарског које припадају геометрији равни. У тим поглављима, која се протежу на 120 страна књиге, налазе се 203 теореме од којих 179 припадају геометрији равни.

Извршено је неколико експеримената. У првом експерименту коришћене су само теореме из књиге, у другом експерименту коришћене су додатне леме из постојеће Coq формализације књиге Тарског [67], а у трећем експерименту коришћене су листе зависности између теорема добијене на основу доказа приказаних у постојећој Coq формализацији. Резултати показују да се 37% теорема књиге може аутоматски доказати без икаквог навођења (када се као премисе тврђења користе све претходне аксиоме и теореме из књиге). У случају коришћења додатних лема проценат доказаних лема расте на 42%. Ови резултати сугеришу да се описани систем и слични начини аутоматског доказивања теорема могу успешно користити као помоћ математичарима приликом развијања формализованог математичког знања.

6.1.1 Аксиоматски систем

Аксиоме Тарског су изражене у терминима логике првог реда са једнакошћу, без типова (једини примитивни објекти су *тачке*, и означавају се малим словима латинице), са два примитивна предикатска симбола: D (који означава релацију *подударно*) и B (који означава релацију *између*). Да би повећали читљивост добијених доказа користићемо стандардне ознаке које се користе у геометрији, тачке ће бити означене великим словима латинице, скупови тачака ће бити означени великим грчким словима, и предикатски симболи D и B ће бити означени са $cong$ (у префиксном облику) и bet. Ове ознаке ће бити коришћене у остатку текста, чак и када се цитира оригиналан материјал Тарског. Тако записане аксиоме Тарског су (претпоставља се да су све аксиоме универзално затворене):

A1 (симетрија): $cong(A, B, B, A)$

A2 (псеудо-транзитивност): $cong(A, B, P, Q) \land cong(A, B, R, S) \Rightarrow cong(P, Q, R, S)$

A3 (рефлексивност релације cong): $cong(A, B, C, C) \Rightarrow A = B$

A4 (конструкција): $\exists X\ (bet(Q, A, X) \land cong(A, X, B, C))$

A5 (пет сегмената): $A \neq B \land bet(A, B, C) \land bet(A', B', C') \land cong(A, B, A', B') \land cong(B, C, B', C') \land cong(A, D, A', D') \land cong(B, D, B', D') \Rightarrow cong(C, D, C', D')$

A6 (рефлексивност релације bet): $bet(A, B, A) \Rightarrow A = B$

A7 (Пашова аксиома):
$bet(A,P,C) \ \wedge \ bet(B,Q,C) \ \Rightarrow \exists X \ (bet(P,X,B) \ \wedge \ bet(Q,X,A))$

A8 (аксиома минималне димензије):
$\exists A \ \exists B \ \exists C \ (\neg bet(A,B,C) \ \wedge \ \neg bet(B,C,A) \ \wedge \ \neg bet(C,A,B))$

A9 (аксиома максималне димензије):
$P \neq Q \ \wedge \ cong(A,P,A,Q) \ \wedge \ cong(B,P,B,Q) \ \wedge \ cong(C,P,C,Q) \ \Rightarrow$
$(bet(A,B,C) \vee bet(B,C,A) \vee bet(C,A,B))$

A10 (Еуклидова аксиома): $bet(A,D,T) \ \wedge \ bet(B,D,C) \ \wedge \ A \neq D \ \Rightarrow$
$\exists X \ \exists Y \ (bet(A,B,X) \ \wedge \ bet(A,C,Y) \ \wedge \ bet(X,T,Y))$

A11 (непрекидност): $\forall \Phi \ \forall \Psi \ \exists A \ \forall X \ \forall Y \ ((X \in \Phi \wedge Y \in \Psi \Rightarrow bet(A,X,Y))$
$\Rightarrow \exists B \ \forall X \ \forall Y \ (X \in \Phi \wedge Y \in \Psi \Rightarrow bet(X,B,Y))$

Аксиома A8 тврди да постоје три неколинеарне тачке, што значи да је димензија простора најмање 2. Аксиома A9 тврди да, ако за три тачке важи да су на истим удаљеностима од две различите тачке, да су онда оне колинеарне. Отуда се може закључити да је димензија простора највише 2. Овако записана аксиома је у ствари специјалан случај аксиоме A9 која има општији облик:

Axiom A9':

$$\left[\bigwedge_{1 \leq i < j \leq n} P_i \neq P_j \wedge \bigwedge_{i=2}^{n} cong(A,P_1,A,P_i) \right.$$
$$\left. \wedge \bigwedge_{i=2}^{n} cong(B,P_1,B,P_i) \wedge \bigwedge_{i=2}^{n} cong(C,P_1,C,P_i) \right]$$
$$\Rightarrow [(bet(A,B,C) \vee bet(B,C,A) \vee bet(C,A,B)]$$

Практичност геометрије Тарског се огледа у томе што се димензија простора контролише уз помоћ само једног параметра (n) који се користи само у овој аксиоми. У овој тези ћемо разматрати само вредност $n = 2$ (како је приказано аксиомом A9), тј., радићемо са геометријом равни.

Аксиома A10 је еквивалентна (у односу на остале аксиоме) Еуклидовом постулату паралелности. Користи се у поглављу 12 и наредним поглављима књиге Тарског.

Аксиома A11 (аксиома непрекидности) у себи садржи скупове тачака Φ и Ψ и може се преформулисати у схема аксиому логике првог реда:

A11': $\exists A \ \forall X \ \forall Y \ ((\phi(X) \wedge \psi(Y) \Rightarrow bet(A,X,Y)) \Rightarrow \exists B \ \forall X \ \forall Y \ (\phi(X) \wedge \psi(Y) \Rightarrow$
$bet(X,B,Y))$

где су ϕ и ψ произвољне формуле логике првог реда без слободних појављивања променљивих A и B.

Занимљиво је приметити да све аксиоме, осим A11, припадају кохерентној логици и да се аксиома A11 не користи у првих 12 поглавља књиге Тарског.

Гупта је анализирао независност између аксиома овог аксиоматског система [41]. Пре неколико година, Тимоти Макариос (Timothy Makarios) је предложио ново поједностављење овог аксиоматског система [57]. Замена двеју тачака у аксиоми A5 допушта избацивање аксиоме A2. Показана је независност свих аксиома осим аксиома A3 и A7 (што представља отворен проблем).

6.1.2 Запис геометрије Тарског у кохерентној логици

Највећи део књиге Тарског се једноставно записује у кохерентној логици. У овом поглављу ће бити описано превођење свих аксиома, дефиниција и теорема, што је могуће верније оригиналној формулацији. Након превођења у кохерентну форму добија се 238 теорема, што је незнатно повећање у односу на почетних 179 теорема. Поступак превођења у кохерентну форму није увек једноставан и том приликом су направљене неке измене и модификације оригиналних формула (неке од тих измена су коришћене и у оквиру других формализација књиге Тарског).

Симболи негације.

Као што је раније речено, кохерентна логика не подржава негације, па се за сваки предикатски симбол R уводи нови предикатски симбол $\overline{R}$ који одговара изразу $\neg R$, и уводе се наредне две аксиоме: $R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot$, $R(\vec{x}) \vee \overline{R}(\vec{x})$. Конкретно, у случају геометрије Тарског може бити показано да се овај општи приступ може поједноставити. Габриел Браун (Gabriel Braun) и Жилиен Нарбу су недавно показали да су, у контексту аксиоматског система Тарског, оба тврђења $cong(\vec{x}) \vee \overline{cong}(\vec{x})$ и $bet(\vec{x}) \vee \overline{bet}(\vec{x})$ екивалентна правилу искључења трећег за једнакост. Након тог запажања Coq формализација геометрије Тарског [18] је поновно креирана тако да не користи правило искључења трећег у општем облику. Показано је да се првих 12 поглавља књиге Тарског може формализовати користећи само правило искључења трећег за једнакост и пресек правих [16].

Функцијски симболи.

Осим функцијских симбола који су везани за праве и равни (на пример, симбол $L(P, Q)$ одговара правој која је одређена двема различитим тачкама P и Q), употреба функцијских симбола је веома ретка у књизи Тарског. Функцијски симболи се први пут појављују у седмом поглављу у дефиницији 7.5 (страна 49 [79]) која уводи тачку добијену централном симетријом:

$$S_A(P) = P' := is\_midpoint(P, A, P'),$$

где се $is\_midpoint(P, A, P')$ дефинише као $bet(P, A, P') \wedge cong(A, P, A, P')$. Како се функцијски симболи не користе у кохерентној логици, уводи се додатни предикатски симбол $is\_symmetric$ који се дефинише на следећи начин:

$$is\_symmetric(P, P', A) := is\_midpoint(P, A, P').$$

Са увођењем овог предикатског симбола потребно је унети одређене промене у формулацији неких теорема (7.13, 7.15, 7.16, 7.18 и 7.19). Приликом сваког имплицитног коришћења функције $S_A(P)$ (без навођења симетричне тачке), у формулацију теореме се додају нова тачка P' и чињеница $is\_symmetric(P, P', A)$.

Наредни функцијски симбол који се користи у књизи Тарског се уводи у десетом поглављу у дефиницији 10.1:

$$M(A, B) = X := is\_midpoint(A, X, B).$$

Како већ постоји предикатски симбол који осликава ову функцију, неће се уводити додатан предикатски симбол. Употреба овог функцијског симбола у наредним формулама се изводи на исти начин као и за претходни функцијски симбол.

Осим ових функцијских симбола постоји још један симбол – is_image, дефинисан у десетом поглављу у дефиницији 10.3.

Рад са скуповима.

Тарски је изградио елементарну геометрију у оквиру логике првог реда коришћењем само једног примитивног типа објеката — *тачака*. Међутим, у неким дефиницијама се користе скупови при чему није наглашено која теорија скупова или који део теорије скупова се користи. На пример, у шестом поглављу, *полуправе* су дефинисане на следећи начин (Дефиниција 6.8, страна

44 [79]):

$$H(PA) := \{X | X \simeq_P A\},$$

где се $A \simeq_P B$ (интуитивно: тачке A и B се налазе са исте стране тачке P) дефинише као $A \neq P \wedge B \neq P \wedge (bet(P, A, B) \vee bet(P, B, A))$. Такође, под претпоставком да су P и Q две различите тачке, *права* $L(PQ)$ одређена тим тачкама се дефинише на следећи начин (Дефиниција 6.14, стана 45) ($col(A, B, C)$ се дефинише са $bet(A, B, C) \vee bet(B, C, A) \vee bet(C, A, B)$):

$$L(PQ) \ := \ \{X | col\ P\ Q\ X)\}. \tag{6.1}$$

Додатно, уводи се предикатски симбол Ln (страна 45 [79]) који представља праве, и који се користи у многим наредним теоремама:

$$Ln\ a \ := \ \exists P \exists Q(P \neq Q \wedge a = L(PQ)). \tag{6.2}$$

Скупови неће бити коришћени у оквиру формализације приказане у овој тези. Уместо њих ће се користити додатни предикатски симболи (који имитирају рад са скуповима тачака). Уместо релације $A \in p$ користи се предикатски симбол $point\_on\_line(A, X, Y)$ дефинисан на следећи начин:[1]

$$point\_on\_line(A, X, Y) \ := \ (X \neq Y \ \wedge \ col(A, X, Y))).$$

Слично, уместо релације $p = q$, користи се предикатски симбол $same\_lines(P1, P2, Q1, Q2)$ (где је p права одређена тачкама $P1$ и $P2$, а q права одређена тачкама $Q1$ и $Q2$) дефинисан на следећи начин:

$$same\_lines(P1, P2, Q1, Q2) \ :=$$

$$P1 \neq P2 \wedge Q1 \neq Q2 \wedge point\_on\_line(P1, Q1, Q2) \wedge point\_on\_line(P2, Q1, Q2).$$

Постоји још неколико предикатских симбола који су уведени на сличан начин и међу њима су (r/k означава предикатски симбол r арности k): $point\_on\_plane3p/4$, $point\_on\_plane2l/5$, $line\_on\_plane3p/5$, $same\_planes2l/8$.

Суштински, показује се да се скупови у књизи Тарског користе само за краћи запис формула и да се једноставно могу избећи. Проблем који се онда јавља је што би тако добијене формуле биле превише компликоване. На пример, презаписивање релације једнакости скупова у формули којом се записује да су

[1] Ова дефиниција се тривијално раздваја на две импликације које припадају кохерентној форми и могу се користити у оквиру овог система.

праве p и q једнаке, $Ln\ p = Ln\ q$, би дало следећу формулу $Ln\ p \wedge Ln\ q \wedge \forall X(X \in p \Leftrightarrow X \in q)$. Након тога се применом дефиниција 6.1 и 6.2 добија формула која не користи скупове: $P1 \neq P2 \wedge Q1 \neq Q2 \wedge \forall X(col\ P1\ P2\ X \Leftrightarrow col\ Q1\ Q2\ X)$. Уместо елиминисања скупова на тај начин у овој тези су коришћени нови предикатски симболи. Може се показати да су нови предикатски симболи еквивалентни оригиналним предикатским симболима. Формални доказ ових тврђења (у оквиру интерактивног доказивача Isabelle) је тема текућег рада аутора и сарадника.

Кохерентна форма.

Првих 10 аксиома и све теореме (179) из првих 12 поглавља књиге Тарског (на које су примењене модификације описане у претходном делу текста) могу бити подељене у наредне четири групе:

1. *Формуле које припадају кохерентној логици.* Као што је и очекивано, већина теорема (њих 94) из књиге Тарског се већ налази у кохерентној форми (што је и главни разлог зашто је кохерентна логика коришћена).

2. *Формуле које могу били* лако *преведене у кохерентну форму.* Теореме које припадају овој групи (њих 31) могу бити лако преведене у једну или више формула кохерентне логике, употребом наредних трансформација (све формуле су имплицитно универзално квантификоване; $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ означавају конјукцију атомичких формула; x се не појављује слободно ни у једној од наредних формула $\mathcal{A}$, $\mathcal{A} \Rightarrow \forall x\mathcal{B}$, итд.):

$$
\begin{aligned}
\mathcal{A} \Leftrightarrow \mathcal{B} &\equiv \mathcal{A} \Rightarrow \mathcal{B} \wedge \mathcal{B} \Rightarrow \mathcal{A} \\
\mathcal{A} \Leftrightarrow (\mathcal{B} \vee \mathcal{C}) &\equiv (\mathcal{A} \Rightarrow \mathcal{B} \vee \mathcal{C}) \wedge (\mathcal{B} \Rightarrow A) \wedge (\mathcal{C} \Rightarrow \mathcal{A}) \\
\mathcal{A} \Rightarrow (\mathcal{B} \Leftrightarrow \mathcal{C}) &\equiv (\mathcal{A} \wedge \mathcal{B} \Rightarrow \mathcal{C}) \wedge (\mathcal{A} \wedge \mathcal{C} \Rightarrow \mathcal{B}) \\
\mathcal{A} \Leftrightarrow \mathcal{B} \wedge (\mathcal{C} \vee \mathcal{D}) &\equiv A \Leftrightarrow (\mathcal{B} \wedge \mathcal{C}) \vee (\mathcal{B} \wedge \mathcal{D}) \\
\mathcal{A} \Rightarrow (\mathcal{B} \wedge (\mathcal{C} \Rightarrow \mathcal{D})) &\equiv (\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{A} \wedge \mathcal{C} \Rightarrow \mathcal{D}) \\
\mathcal{A} \Rightarrow \forall x \mathcal{B} &\equiv \forall x (\mathcal{A} \Rightarrow \mathcal{B}) \\
\mathcal{A} \Leftrightarrow \exists x \mathcal{B} &\equiv ((\mathcal{A} \Rightarrow \exists x \mathcal{B}) \wedge (\forall x (\mathcal{B} \Rightarrow \mathcal{A}))) \\
\mathcal{A} \Rightarrow \mathcal{B} \wedge \exists x \mathcal{C} &\equiv \mathcal{A} \Rightarrow \exists x (\mathcal{B} \wedge \mathcal{C}) \\
\mathcal{A} \Rightarrow \exists^{\leq 1} x \mathcal{B}(x) &\equiv (\forall x_1 \forall x_2 (\mathcal{A} \wedge \mathcal{B}(x_1) \wedge \mathcal{B}(x_2) \Rightarrow x_1 = x_2)) \\
\mathcal{A} \Rightarrow \exists^{=1} x \mathcal{B}(x) &\equiv (A \Rightarrow \exists x \mathcal{B}(x)) \wedge \\
&\quad (\forall x_1 \forall x_2 (\mathcal{A} \wedge \mathcal{B}(x_1) \wedge \mathcal{B}(x_2) \Rightarrow x_1 = x_2))
\end{aligned}
$$

3. *Формуле које могу бити преформулисане за запис у кохерентној логици.* Теореме ове групе (њих 48) најчешће садрже предикатске симболе који одговарају типу правих и третирају праве као скупове тачака па је отуда једноставна трансформација тих теорема потребна. У овој групи постоји само једна дефиниција (Дефиниција 8.11) која захтева мало компликованију трансформацију (a и a' су праве, предикатски симбол $per(U, X, V)$ означава да је угао UXV прав угао):

$$a \perp_X a' \ := \ X \in a \wedge X \in a' \wedge \forall U \forall V (U \in a \wedge V \in a' \Rightarrow per(U, X, V)).$$

Трансформација ове дефиниције има за циљ елиминисање унутрашњих универзалних квантификатора ($\forall U \forall V$). Први корак трансформације је увођење новог предикатског симбола ($perp\_in$) и представљање правих као парова тачака (елиминисање унутрашњих универзалних квантификатора се изводи на основу геометријских својстава који важе):

$$perp\_in(X, A, B, C, D) := (A \neq B \wedge C \neq D \wedge X \in lineAB \wedge X \in lineCD \wedge$$

$$((A \neq X \wedge C \neq X \wedge per(A, X, C)) \vee (A \neq X \wedge D \neq X \wedge per(A, X, D)) \vee$$

$$(B \neq X \wedge C \neq X \wedge per(B, X, C)) \vee (B \neq X \wedge D \neq X \wedge per(B, X, D)))).$$

Овако добијена дефиниција се затим трансформише (на раније описан начин) у осам формула кохерентне логике. Једна од тих формула (записана у TPTP формату) је:

```
fof(ax_8_11_1_1, axiom, (! [X,A,B,C,D] :
   ((perp_in(X,A,B,C,D) & A!=X & C!=X) =>
   (A!=B & C!=D & point_on_line(X,A,B) &
   point_on_line(X,C,D) & per(A,X,C))))).
```

Трансформације овог типа могу генерисати компликоване дефиниције и свакако се мора наћи компромис између читљивости добијених формула са једне стране и коришћења кохерентних формула и избегавања коришћења скупова са друге стране. С тим у виду, трансформација која је коришћена у овом случају ће произвести читљивије доказе него што би били добијени да је коришћен општи алгоритам [8] за превођење формула (логике првог реда) у кохерентну форму. У општем случају, такво превођење би генерисало одређен број помоћних предикатских симбола.

4. *Формуле које садрже n-торке (за произвољно n).* Теореме ове групе (њих 5) су обрађене само за специјалан случај када је $n \leq 2$.

TPTP формат записа.

TPTP формат записа формула[2] [87] се може користити за запис формула првог реда и самим тим, како је кохерентна логика део логике првог реда, може бити коришћен и за запис кохерентних формула. Отуда се формуле (аксиоме и теореме) из књиге Тарског, добијене горе описаним трансформацијама, могу директно записати у том формату.

Списак свих аксиома, дефиниција и теорема коришћен за запис књиге Тарског се може наћи на интернету.[3]

6.1.3 Доследност формализације

Приликом формализације математичке теорије, интерактивни доказивача теорема потврђују (веома поуздано) да су извођења направљена у току процеса

[2] http://www.cs.miami.edu/~tptp/
[3] http://argo.matf.bg.ac.rs/?content=downloads

доказивања исправна, али одговорност за формулисање аксиома и записа формула и теорема у оквиру те теорије је у потпуности препуштена кориснику који пише те доказе. Може да се деси да корисник неисправно формулише аксиоме математичке теорије, или да направи штампарску грешку и тиме да направи неконзистентну теорију. Осим тога, корисник може унети грешке у формулације дефиниција и теорема које даље могу утицати на остале дефиниције и теореме. Из ових разлога је потребно обезбедити неку врсту аутоматске провере која би могла да се користи за откривање потенцијалних неконзистентности и нејасноћа. У ту сврху могу бити коришћени аутоматски доказивачи теорема.

Приликом изградње рачунарске верзије књиге Тарског, коришћени су резолуцијски доказивачи за проверу неконзистентности (са циљем откривања наших грешака) на следећи начин: унутар одређеног временског интервала, извршени су покушаји да се изведе $\perp$ из датог скупа аксиома, дефиниција и теорема. Иако једноставна, коришћење ове хеуристике је открило неколико грешака у оквиру наше формализације (најчешће у виду штампарских грешака или изостављања предиката). Са присутним грешкама, може се десити да нека тврђења више нису теореме или, што је још горе, неконзистентне претпоставке уведене аксиомама и теоремама могу довести до тривијалних доказа наредних теорема (због неконзистентних премиса).

6.1.4 Преглед постојећих рачунарских формализација књиге Тарског

Тренутно постоје три рачунарске формализације књиге Тарског од којих су две добијене коришћењем аутоматских доказивача теорема, а трећа користи интерактивни доказивач теорема. Формализације које користе аутоматски доказивач теорема су развили Арта Куаиф [74] и касније Мајкла Бисон (Michael Beeson) и Ларија Воса (Larry Wos) [5, 6]. Формализација која користи интерактивни доказивач теорема су развили Нарбу и Браун [67, 18].

Аутоматски доказивач теорема Otter је користио Куаиф у оквиру своје формализације. Доказао је теореме из првих 5 поглавља књиге и цео процес је трајао око две недеље. Касније су Бисон и Вос користили новију верзију доказивача Otter, поновили су и продужили Куаифов рад. Давањем додатних информација доказивачу у виду наговештаја (“hints”) и резонатора (“resonators”) успели су да докажу све теореме до теореме 9.6., укључујући и њу, као и већи део теорема до поглавља 12 [5, 6]. Ова формализација је суштински полуаутоматска, јер су аутори интензивно наводили доказивач

приликом доказивања тешких теорема: давали су му информације о тачкама које треба конструисати, као и значајан скуп корака извођења. Осим тога, у оквиру ове формализације коришћене су помоћне леме које се не појављују у књизи. Неке од њих су инспирисане Coq формализацијом (на пример лема 1 у поглављу 5[4]). У оквиру ове формализације нису разматрани ни формални докази (провериви од стране интерактивних доказивача теорема) нити читљиви докази на природном језику.

Интерактивни доказивач теорема Coq (са минималним нивоом аутоматизације) су користили Браун и Нарбу и доказали су већину теорема (не укључујући теореме о геометрији простора) из првих дванаест поглавља [67, 18]. Неколико година после тог пројекта, формализација Тарског је проширена. Браун, Нарбу и Пјер Баутри (Pierre Boutry) су доказали неке геометријске теореме вишег нивоа које се заснивају на аксиоматском систему Тарског: постојање тежишта, ортоцентра, описаног круга као и нека стандардна својства четвороугла [17].

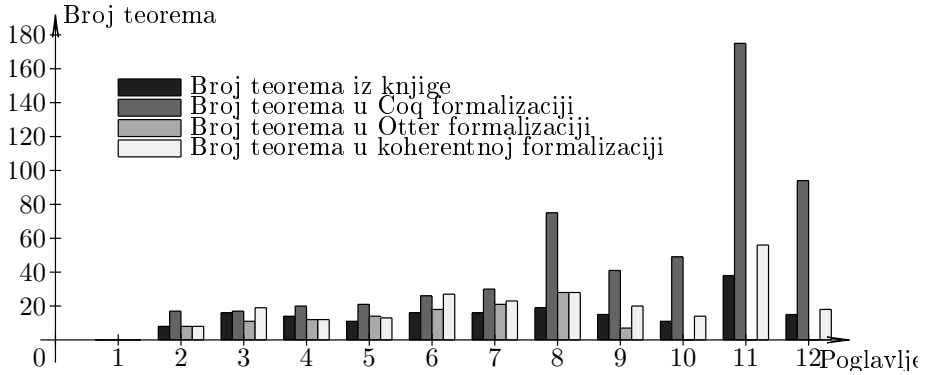
Слично као у формализацији описаној у овој тези, ни Coq ни Otter формализације не користе скупове и све формуле из књиге Тарског су измењене у складу са тим. На пример, ни једна од описаних формализација не користи концепт правих – уместо тога, праве се користе имплицитно и представљене су паровима тачака. Међутим, за разлику од приступа описаног у овој тези, описане формализације не користе додатне предикатске симболе који имитирају рад са скуповима тачака. Слично, ниједна формализација не користи функцијске симболе у запису формула: функција арности n је представљена предикатским симболом арности $n+1$. Из разлога што се скупови не користе у овим формализацијама, потребно је додати леме као што је наредна која описује псеудо-транзитивност предикатског симбола колинеарно:

$$A \neq B \wedge col(A, B, C) \wedge col(A, B, D) \Rightarrow col(A, C, D).$$

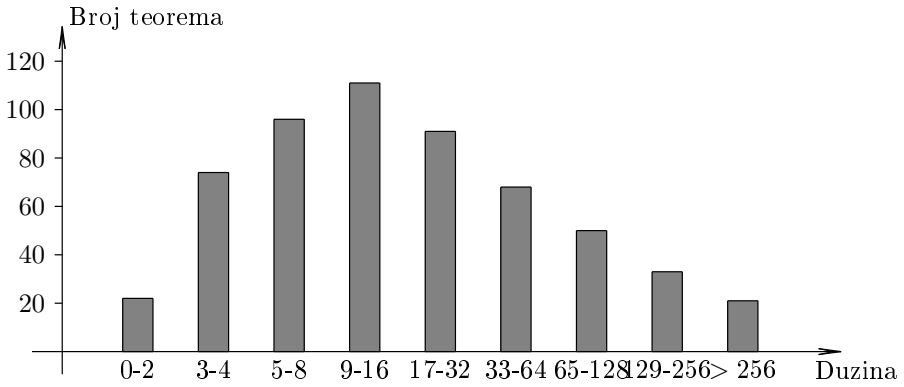
Поглавља књиге Тарског се разликују у обиму и тежини теорема које се у њима налазе што се може директно да утиче на развој рачунарске формализације. Увид у тежину и обим поглавља (наравно неформалан) се може добити посматрајући укупан број доказаних тврђења у оквиру рачунарске формализације. Укупан број доказаних тврђења је: 565 у Coq формализацији[5],

[4] http://www.michaelbeeson.com/research/FormalTarski/index.php?include=archive5

[5] У оквиру Coq формализације налазе се и неке леме које се не користе за доказивање теорема из књиге Тарског, већ за друге сврхе. Од 565 тврђења која се налазе у овој формализацији, њих 456 се користе за доказивање теорема књиге Тарског.



Слика 6.1: *Број теорема у књизи и у рачунарским формализацијама.*



Слика 6.2: *Расподела дужина доказа теорема добијених у Coq формализацији.*

119 у Otter формализацији (обухвата теореме до теореме 9.6), и 238 у нашој формализацији (у оквиру кохерентне логике). На слици 6.1 се може видети број теорема по поглављима који се налази у књизи Тарског као и у различитим рачунарским формализацијама. Као неформална мера тежине теореме може се посматрати број редова доказа потребних за формално доказивање теореме у оквиру Coq формализације. Наравно, теорема може бити доказана на више различитих начина, тако да ова мера није идеална али може послужити као илустрација. На слици 6.2 је приказана расподела теорема у односу на дужину њихових доказа у Coq формализацији.

6.1.5 Упаривање Coq формализације са кохерентном формализацијом

У оквиру спроведених експеримената, постојећа Coq формализација, као и информације добијене из ње, су коришћене за навођење аутоматске формализације (на начин који ће бити накнадно описан). Да би се то могло

урадити прво је било потребно упарити тврђења из Coq формализације са оригиналном књигом Тарског, па затим и са тврђењима добијеним у нашој кохерентној формализацији. Ово упаривање је делом изведено ручно, делом коришћењем неких помоћних алата. За одређен број тврђења ово упаривање је било тривијално, али није занемарљив ни број тврђења чије упаривање није било једноставно. Процес упаривања ће бити описан у наставку текста.

Велики број теорема Тарског је у оквиру Coq формализације био издељен на неколико теорема које махом припадају кохерентној логици (иако то аутори Coq формализације нису имали у виду, нити су планирали коришћење доказивача за кохерентну логику у будућности), што је чинило упаривање са нашом формализацијом тривијалним. Али, у тим случајевима, упаривање са теоремама књиге Тарског није било један на један.

У случају када теорема коришћена у Coq формализацији није у кохерентној форми, прво је морала бити преведена у њу. Након тога је упарена са оригиналном теоремом Тарског (у оквиру Coq формализације, теореме Тарског нису увек имале униформна имена па је ово упаривање урађено ручно, тј. поређењем формулације теорема са књигом) и убачена у нашу кохерентну формализацију.

Осим оригиналних теорема књиге Тарског, Coq формализација у себи садржи и известан број додатних лема које су коришћене да би се поједноставили докази оригиналних теорема из књиге. Све те леме је такође потребно убацити у листу формула у оквиру кохерентне формализације (на одговарајуће место). Да би се то урадило, прво су све Coq леме преведене у кохерентну форму (у оним случајевима у којима је то било потребно). На основу Coq формализације креиране су листе зависности међу тврђењима: за свако тврђење листа зависности садржи све теореме и леме коришћене у њеном доказу. Приликом спајања две формализације, листе зависности су коришћене за одређивање позиције Coq леме у кохерентној формализацији. Позиција леме (која не постоји у кохерентној формализацији) је таква да се лема налази на последњем могућем месту унутар листе тврђења (односно, непосредно пре првог тврђења у чијем се доказу та лема користи).

Спајање те две формализације (Coq формализације и кохерентне формализације) је дало велики број изазова, делом због различите синтаксе, коришћења различитих имена, различитих алата за записивање формула, али и због незнатних разлика у формулацији оригиналних теорема Тарског. Како је књига Тарског веома прецизно написана, (незнатно) различити записи формула

нису били очекивани.[6]

Након пажљивијег разматрања уочава се главни разлог ових проблема: као што је већ напоменуто, за разлику од оригиналне књиге Тарског, ниједна рачунарска формализација не користи скупове. Ипак, записивање формула књиге Тарског без коришћења скупова није увек праволинијски. Осим тога, приликом записивања формула на рачунару (без обзира на то да ли се у њима јављају скупови или не) могуће је направити одређене (синтаксне или семантичке) грешке. Као илустрација биће приказана укратко (користећи уобичајене математичке ознаке) грешка која је откривена у оквиру постојеће Coq формализације. Грешка је откривена приликом спајања две формализације.

У Coq формализацији, предикатски симбол $is\_image(P', P, A, B)$ (тачке P и P' су симетричне у односу на праву AB) је дефинисан (на основу првог дела дефиниције 10.3 из књиге Тарског) на следећи начин:

$\exists X(is\_midpoint(X, P, P') \wedge col(A, B, X) \wedge (perp(A, B, P, P') \vee P = P')$,
где $is\_midpoint(X, P, P')$ важи ако је X средишња тачка дужи PP', $col(A, B, C)$ важи ако су тачке A,B и C колинеарне, и $perp(A, B, P, P')$ важи ако је права AB нормална на PP'. У оквиру Coq формализације налази се наредна лема:

$$col(A, B, X) \Rightarrow is\_image(X, X, A, B).$$

Али, дефиниција 10.3 покрива и специјалан случај када је $A = B$ ($is\_symmetric(P, P', A)$ важи ако су тачке P и P' симетричне у односу на тачку A):

$$is\_image(P, P', A, B) := \begin{cases} \exists X \left(\begin{array}{c} is\_midpoint(X, P, P') \wedge col(A, B, X) \\ \wedge (perp(A, B, P, P') \vee P = P') \end{array} \right) & \text{if} A \neq B \\ is\_symmetric(P, P', A) & \text{if } A = B \end{cases}$$

Дакле, из чињенице $col(A, A, P)$ се може закључити да важи и $is\_image(P, P, A, A)$ па отуда и $is\_symmetric(P, P, A)$, што није тачно у општем случају (тј., није тачно када је $P \neq A$). Овај проблем се јавио из разлога што књига Тарског користи праве (које представљају скупове тачака), док Coq формализација користи само тачке и превођење формуле тако да не садржи скупове није било у потпуности верно.

Овај мали превид који се јавио у оквиру дефиниције предиката откривен је приликом упаривања ове формализације са кохерентном формализацијом и

[6]Приликом формализације математичких књига, често се срећемо са проблемом тумачења записа аутора због појаве разних нејасноћа. Овај проблем је посебно изражен за неке класичне математичке књиге као на пример Хилбертова *Grundlagen der Geometrie* [45], што је приказано у оквиру формализације те књиге [61, 62].

резултирао је неконзистентношћу која је откривена коришћењем резолуцијских доказивача. Након откривања ове грешке, унете су измене у оригиналну Coq формализацију (измене у дефиницији су захтевале измене у многим наредним теоремама и помоћним лемама) чиме је добијено много мање недегенеративних случајева (многи предуслови $A \neq B$ су уклоњени захваљујући општијој дефиницији предиката is_image).

Осим откривања ове грешке, употреба аутоматских доказивача теорема приликом процеса спајања две формализације је помогла да се открију неки дупликати и непотребне леме (у случајевима када се помоћна лема доказује уз помоћ само једне леме, онда она није потребна и може бити елиминисана).

6.1.6 Процес аутоматизације и извођење експеримената

Након преформулисања у кохерентну форму (у случају да је потребно, на начин описан у поглављу 6.1.2) свих аксиома, дефиниција и теорема из првих дванаест поглавља књиге Тарског, спроведено је неколико експеримената у циљу откривања колико теорема (од укупно 238) може бити доказано потпуно аутоматски и уз одређено навођење[7].

Постојеће формализације (полуаутоматска Otter формализација и ручна Coq формализација) могу бити коришћене као референтне тачке (за поређење са нашим потпуно аутоматским приступом), али и за симулацију навођења од стране човека. За симулацију навођења коришћене су информације добијене из интерактивно развијених доказа у оквиру интерактивног доказивача теорема Coq [67, 18].

Експерименти су изведени на серверу са 12 језгара и процесором AMD Opteron 6168 CPU. За сваку теорему прво су позвани резолуцијски доказивачи (за сваку теорему два пута, са премисама задатим у нормалном и обрнутом редоследу) са временским ограничењем од 60 секунди,[8] док је доказивач ArgoCLP коришћен са временским ограничењем од 1000 секунди. [9]

[7] Под навођењем сматрамо давање додатних информација доказивачима у виду додатних лема или смањеног скупа тврђења али примењено униформно за сва тврђења. У оквиру ових експеримената није коришћено навођење за доказивање појединачних теорема.

[8] Релативно мало временско ограничење је намерно изабрано да би добили слику о потенцијалним применама налик интерактивном доказивању (Sledgehammer приступ користи временско ограничење од 30 секунди). Повећање временског ограничења не утиче значајно на добијене резултате: у експериментима без навођења, резолуцијски доказивачи су успешно доказали 48% теорема са временским ограничењем од 1 минута, 49% теорема са временским ограничењем од 2 минута, 51% теорема са временским ограничењем од 5 минута, 54% теорема са временским ограничењем од 30 минута.

[9] Временско ограничење задато доказивачу ArgoCLP је релативно високо, али у ствари није много веће него време које је одвојено за резолуцијске доказиваче (с обзиром да они укупно добијају временско ограничење од $6 \times 60s$). Осим тога, велико временско ограничење би се

Потпуна аутоматизација, без икаквог навођења (без давања додатних информација).

У оквиру првог експеримента, систем (описан у глави 5.1) је примењен на 238 теорема добијених након записа првих дванаест поглавља књиге Тарског у кохерентној логици. Аксиоме, дефиниције и теореме су наведене управо у редоследу којим су наведене у књизи. Приликом доказивања текуће теореме, све претходне аксиоме, дефиниције и теореме (без обзира на то да ли су претходно доказане од стране система) се користе, тј. прослеђују се доказивачима. Осим тога аксиоме се користе на следећи начин:

- за доказивање теорема из првих једанаест поглавља прослеђујемо аксиоме A1-A9 доказивачима; за доказивање теорема из поглавља 12 прослеђујемо и аксиому A10. Аксиома A11 није коришћена.
- аксиоме наредног облика: $\forall\vec{x}(R(\vec{x})\wedge\overline{R}(\vec{x})\Rightarrow\bot)$, $\forall\vec{x}(R(\vec{x})\vee\overline{R}(\vec{x}))$, се додају за све примитивне и дефинисане предикатске симболе R.

Применом овог потпуно аутоматског приступа, од укупно 238 теорема, 114 (48%) је доказано од стране (бар једног) резолуцијског доказивача, 87 (37%) њих је доказано од стране доказивача ArgoCLP, а за њих 41 (17%) су доказане (од стране ArgoCLP-а) и све теореме и леме коришћене у доказу. Детаљне информације по поглављима се налазе у табели ?? и приказане су на слици 6.3.

Аутоматизација са имплицитним навођењем (са проширеним скупом лема).

У оквиру другог експеримента, скуп теорема је проширен лемама које су уведене у оквиру Coq формализације. Разлог за спровођење овог експеримента лежи у претпоставци да ће додатне леме бити лакше за доказивање од оригиналних теорема, и да ће процес доказивања самих теорема бити једноставнији када се доказивачима предају и додатне леме. Овај експеримент симулира интеракцију између математичара и рачунара: математичар гради теорију, формулише додатне леме када је потребно и покушава да докаже леме/теореме прво аутоматски, а ако то не успе, онда ручно. Овај и наредни експеримент називамо "аутоматизација са навођењем", јер Coq леме

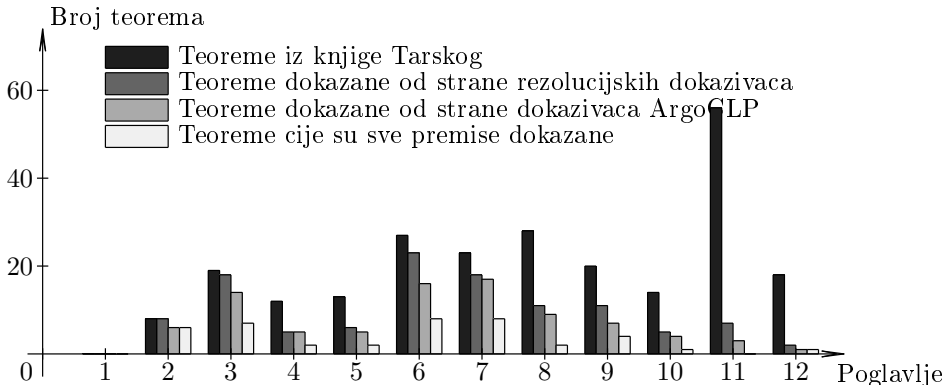
могло користити за формализацију која не би била интерактивна. За интерактивно окружење могло би се користити мање временско ограничење с обзиром да је систем применљив и за временско ограничење од 60 секунди: 84% тврђења која се доказују од стране доказивача ArgoCLP са временским ограничењем од 1000 секунди се доказују и унутар временског ограничења од 60 секунди.

Табела 6.1: *Број доказаних теорема у нашем систему (основни списак), број доказаних теорема када се додају леме из Coq формализације (проширени списак), број доказаних теорема користећи листе зависности добијене на основу доказа у Coq формализацији (листе зависности). За сваку категорију, 'рд' означава колону са теоремама које је доказао бар један резолуцијски доказивач, 'арго' означава колону са теоремама које је доказао доказивач ArgoCLP, а 'све' означава колону са теоремама које су доказане и чије су све леме и теореме које се појављују у доказу такође доказане.*

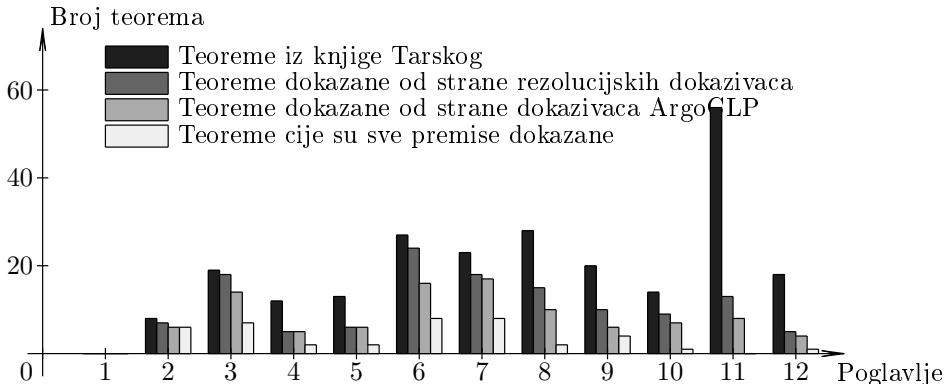
Поглавље	број	основни списак			проширени списак			листе зависности		
	теорема	рд	арго	све	рд	арго	све	рд	арго	све
1	0	0	0	0	0	0	0	0	0	0
2	8	8	6	6	7	6	6	8	7	7
3	19	18	14	7	18	14	7	18	15	13
4	12	5	5	2	5	5	2	7	5	2
5	13	6	5	2	6	6	2	9	6	2
6	27	23	16	8	24	16	8	23	14	9
7	23	18	17	8	18	17	8	19	17	11
8	28	11	9	2	15	10	2	19	8	2
9	20	11	7	4	10	6	4	9	5	4
10	14	5	4	1	9	7	1	9	7	1
11	56	7	3	0	13	8	0	24	8	0
12	18	2	1	1	5	4	1	5	3	1
укупно	238	114	87	41	130	99	41	150	95	52
проценат	100	48%	37%	17%	55%	42%	17%	63%	40%	22%
додатне леме	218				137	88	27	147	76	38
укупно	456				267	187	68	297	171	90
проценат	100				59%	41%	15%	65%	38%	20%

које су формулисали математичари сматрамо врстом додатних информација за навођење аутоматских доказивача. Овај експеримент зовемо "аутоматизација са *имплицитним* навођењем" јер су резолуцијски доказивачи добијали комплетан скуп аксиома, теорема и лема коришћених у формализацији, а не само аксиоме/леме/теореме које су коришћене у постојећим доказима (као што је урађено у наредном експерименту).

Овај експеримент је изведен у истим условима као и претходни. Овим приступом, од 238 теорема, 130 (55%) њих је доказано од стране (бар једног) резолуцијског доказивача, њих 99 (42%) је након тога доказано од стране доказивача ArgoCLP, и за њих 41 (17%) све теореме и леме коришћене у њиховом доказу су такође доказане од стране доказивача ArgoCLP. Ако посматрамо резултате само на лемама које су преузете из Coq формализације, резултати су слични за свих 218 лема: 137/88/27, што укупно гледано (када се посматрају теореме из књиге заједно са лемама из Coq формализације) даје 456 лема/теорема и резултате: 267/187/68. Детаљне информације (по поглављима)



Слика 6.3: *Број теорема доказаних потпуно аутоматским приступом.*

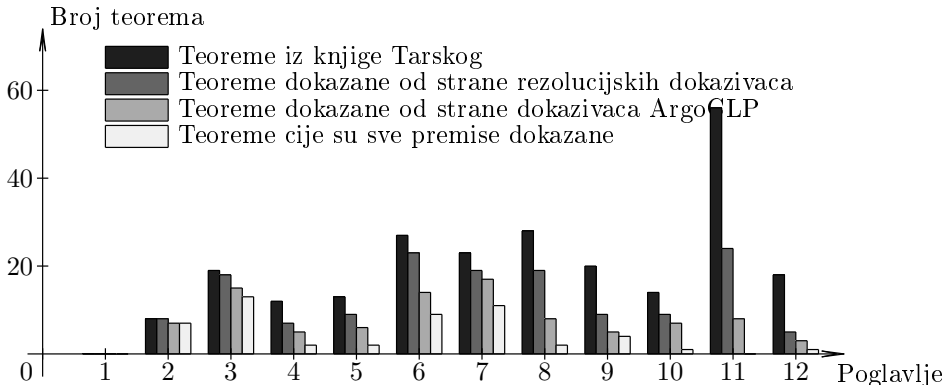


Слика 6.4: *Број теорема доказаних користећи списак додатних лема добијених из Coq формализације.*

се могу наћи у табели ?? и приказани су на слици 6.4.

Аутоматизација са експлицитним навођењем (аутоматизација са листама зависности).

У трећем експерименту, приликом доказивања конкретне теореме користи се само списак лема и теорема који се налази у доказу тог тврђења у оквиру постојеће Coq формализације (добијен на основу листа зависности). Овај експеримент симулира доказивање теореме у случају када је познато које леме су релевантне али процес генерисања формалног, машински проверивог, доказа није тривијалан. Ова ситуација се веома често среће у пракси. Често докази теорема дати у књигама више одговарају скици доказа него "правом" доказу формално гледано. Такође, може се десити да математичар има идеју о доказу теореме и може да претпостави које леме ће користити у доказу. У овом експерименту, иницијално су коришћене само дефиниције и леме/теореме



Слика 6.5: *Број теорема доказаних користећи списак додатних лема и листе зависности добијене из Coq формализације.*

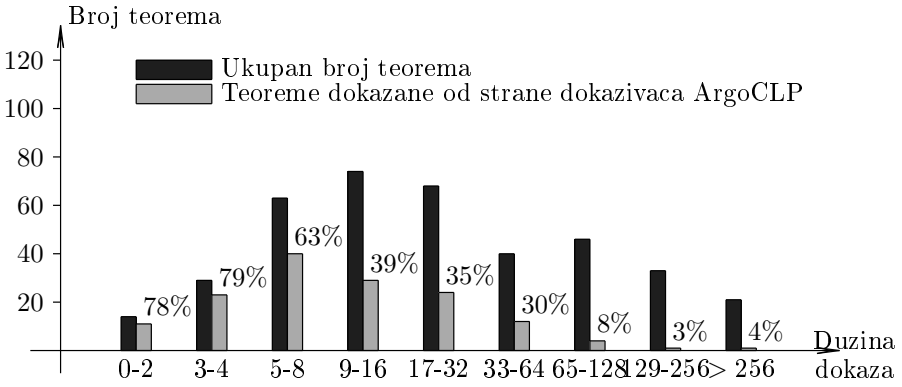
наведене у листама зависности и све аксиоме укључујући и аксиоме облика $\forall\vec{x}(R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot)$, за све примитивне и дефинисане предикатске симболе R (у случају да процес доказивања не буде успешан, овај скуп се може проширити свим дефиницијама које претходе тврђењу које се доказује).

Овим приступом се од 238 теорема доказује њих 150 (63%) од стране (бар једног) резолуцијског доказивача, њих 95 (40%) се након тога доказује од стране ArgoCLP доказивача, и за њих 52 (22%) све теореме и леме коришћене у доказу су такође доказане од стране доказивача ArgoCLP. Када се посматрају само леме преузете из Coq формализације, резултати су слични за свих 218 лема: 147/76/38, што укупно даје 456 лема/теорема са резултатима: 297/171/90. Детаљне информације (по поглављима) се могу видети у табели ?? и приказани су на слици 6.5.

Дискусија.

Спроведени експерименти јасно показују да се аутоматски доказивачи теорема могу успешно користити за доказивање значајног дела тврђења у току процеса формализације одређене теорије. Такође, коришћење додатних лема (добијених из интерактивне формализације које су направили математичари) повећава перформансе аутоматских доказивача што и није изненађујуће. Такође није изненађујуће да се коришћењем листи зависности (добијених из Coq формализације) незнатно повећава успешност резолуцијских доказивача. Оно што јесте изненађење је да кохерентни доказивач ArgoCLP бележи благи пад у броју доказаних теорема (95) када се позива са тим листама зависности (које су прво прослеђене резолуцијским доказивачима) наспрам коришћења листи зависности откривених од стране резолуцијских доказивача (99). Детаљнијим

увидом у добијене податке примећује се да у неким случајевима листе зависности добијене из Coq формализације садрже више лема него докази пронађени од стране резолуцијских доказивача. Ови резултати наговештавају да резолуцијски доказивачи могу бити искоришћени да пронађу краће, алтернативне доказе који би поједноставили неке интерактивне доказе.



Слика 6.6: *Број теорема доказаних уз помоћ доказивача ArgoCLP када се користе додатне леме, наспрам дужине одговарајућих Coq доказа.*

Када се посматра само утрошено време на спровођење експеримента, најмање временски захтеван је био трећи експеримент. Ово није изненађујуће с обзиром да су доказивачи радили са малим скуповима премиса. Овај резултат показује практичност овог приступа, у случајевима да постоје услови за његову примену.

Посматрајући тежину доказаних теорема наш систем показује боље резултате на једноставним теоремама (т.j., на теоремама са краћим доказима у оквиру Coq формализације), као што је и очекивано. На слици 6.6 је приказан проценат теорема доказаних од стране доказивача ArgoCLP када се користе додатне леме наспрам дужина одговарајућих Coq доказа.

У приказаним резултатима може се видети проценат доказаних теорема у случају да су доказане, као и када нису доказане, све леме које се користе у доказу. Иако може деловати да су корисни само они докази у којима су доказане све помоћне леме, ми сматрамо да су и случајеви када нису све помоћне леме доказане употребљиви у пракси. Ти резултати боље осликавају потребу за интервенцијом од стране математичара који решава проблем и практично описују ситуацију у којој математичар формализује одређену теорију кроз интеграцију аутоматских и интерактивних доказивача теорема. Приликом формализације теорије, теореме се доказују редом, једна по једна. За сваку теорему прво би се покушало добијање доказа потпуно аутоматски (користећи

све претходне аксиоме и теореме) а у случају да то не успе математичар би могао сам да комплетира доказ. На тај начин приликом доказивања наредне теореме све премисе би већ биле доказане.

6.2 Геометрија Хилберта

Хилбертов аксиоматски систем је најчешће коришћен аксиоматски систем који се у Србији обрађује у оквиру средњошколске геометрије, већ у првом разреду средње школе, а касније поново и на факултету. Средњошколска геометрија, односно теореме које се у оквиру ње доказују, се често сматра једноставном и докази тих теорема се најчешће наводе са врло мало информација у њима. Осим тога, у уџбеницима за математику, решења задатака често нису потпуна већ само садрже упутство за решавање. У таквим случајевима систем ArgoGeoChecker може бити веома користан.

Знатан део аксиома и теорема које се обрађују у оквиру геометрије у средњој школи припада кохерентној логици. Отуда се у великом броју случајева теореме и њихови докази могу записати у оквиру овог система без икаквих измена. У неким случајевима су докази чак у потпуности изостављени па се систему предаје сама теорема. У случају да тврђење теореме излази из оквира кохерентне логике, у највећем броју случајева оно се може лако превести у мали број формула које припадају кохерентној логици. На пример, неке теореме садрже тврђења типа "постоји само једна тачка" што је тврђење које се не може записати само једном формулом кохерентне логике. Међутим, докази таквих теорема се обично изводе из два дела. Прво се покаже да одређени објекат постоји, а затим да не могу постојати два таква објекта. Управо оваквом поделом теореме и њеног доказа добијамо тврђења која припадају кохерентној логици и која се могу записати у оквиру овог система.

6.2.1 Аксиоматски систем

У наставку текста наведене су прве две групе аксиома Хилбертовог аксиоматског система, аксиоме припадања и аксиоме распореда.

Прва група Хилбертових аксиома

I1 За сваке две различите тачке постоји права која их садржи.

I2 За сваке две различите тачке не постоји више од једне праве која их садржи.

I3a Постоје бар две различите тачке на прави.

I3b Постоје бар три неколинеарне тачке.

I4a За сваке три неколинеарне тачке постоји раван која их садржи.

I4b За сваку раван постоји тачка која јој припада.

I5 За три неколинеарне тачке постоји највише једна раван која их садржи.

I6 Ако две различите тачке праве припадају равни онда свака тачка те праве припада тој равни.

I7 Ако две равни имају једну заједничку тачку онда имају бар још једну заједничку тачку.

I8 Постоје четири некомпланарне тачке.

Друга група Хилбертових аксиома

II1 Ако је тачка B између тачака A и C онда су тачке A, B и C три различите колинеарне тачке и тачка B је између тачака C и A.

II2 Ако су A и B две различите тачке, онда постоји тачка C таква да је тачка B између тачака A и C.

II3 Ако је тачка B између тачака A и C тада тачка C није између тачака A и B нити је тачка A између тачака B и C.

II4 (Пашова аксиома) Ако су A, B и C три неколинеарне тачке и L права равни ABC која не садржи тачку A и сече праву BC у тачки P таквој да је тачка P између тачака B и C, тада права L сече или праву AC у тачки Q таквој да је тачка Q између тачака A и C или праву AB у тачки R таквој да је тачка R између тачака A и B.

6.2.2 Примена система за проверавање неформалних доказа

Рад система ArgoGeoChecker је анализиран над осамнаест теорема и задатака из уџбеника за први разред средње школе [31, 64, 84]. Коришћене су прве две групе Хилбертових аксиома. У додатку B се налазе аксиоме записане у TPTP формату након трансформације у кохерентну логику. У наставку текста приказан је један задатак и његово решење из књиге „Математика за I разред

средње школе" [31], његов неформални доказ (записан на природном језику и у синтакси система) и излаз који се добија покретањем система ArgoGeoChecker.

Задатак 1 Дата је права p и ван ње тачка A. Доказати да све праве које садрже тачку A и секу праву p припадају једној равни.

Решење

Упутство: Доказати да све ове праве припадају равни одређеној тачком A и правом p.

На основу упутства датог у решењу може се записати следећи неформални доказ:

```
Дата је права p и ван ње тачка A и права q која садржи тачку A и сече праву p.
Постоји раван R одређена тачком A и правом p.
Права q припада равни R.
```

Када се овај доказ запише у синтакси која је подржана системом добија се наредни низ формула. Овај низ формула представља улаз у систем.

```
! [P,A,Q] : (line(P) & point(A) & ninc_po_l(A,P) & line(Q) & inc_po_l(A,Q) & int_l_l(P,Q))
? [R] : (plane(R) & inc_po_pl(A,R) & inc_l_pl(P,R))
(inc_l_pl(Q,R))
```

Приликом проверавања овог решења користе се аксиоме припадања. У наставку текста приказан је аутоматски генерисан документ (аргументи исправности доказа) записан на српском језику.

Teorema 6.1 (th_11_01.) *Pod pretpostavkom da važi* $A \notin p$ *i* $A \in q$ *i prave* p *i* q *se seku pokazati da postoji ravan* α *tako da važi* $A \in \alpha$ *i* $p \in \alpha$.

Dokaz:

1. Postoje tačka B i tačka C tako da važi $B \neq C$ i $B \in p$ i $C \in p$ (aksioma $I3a$).
2. Na osnovu činjenica $B \neq C$ i $B \in p$ i $C \in p$ i $A \notin p$ važi $\neg col(B, C, A)$ (aksioma $D1a$).
3. Na osnovu činjenice $\neg col(B, C, A)$ važi $\neg col(C, A, B)$ (aksioma sym_ncol1).
4. Na osnovu činjenice $\neg col(C, A, B)$ važi $\neg col(A, B, C)$ (aksioma sym_ncol1).
5. Na osnovu činjenice $\neg col(A, B, C)$ postoji ravan α tako da važi $A \in \alpha$ i $B \in \alpha$ i $C \in \alpha$ (aksioma $I4a$).
6. Na osnovu činjenica $B \neq C$ i $B \in p$ i $C \in p$ i $B \in \alpha$ i $C \in \alpha$ važi $p \in \alpha$ (aksioma $I6$).
7. Zaključak teoreme sledi iz činjenica $A \in \alpha$ i $p \in \alpha$.

QED

Teorema 6.2 (th_11_02.) *Pod pretpostavkom da važi* $A \notin p$ *i* $A \in q$ *i prave* p *i* q *se seku i* $A \in \alpha$ *i* $p \in \alpha$ *pokazati da važi* $q \in \alpha$.

Dokaz:

1. Na osnovu činjenice prave p i q se seku postoji tačka B tako da važi $p \neq q$ i $B \in p$ i $B \in q$ (aksioma $D6$).
2. Na osnovu činjenica $p \in \alpha$ i $B \in p$ važi $B \in \alpha$ (aksioma $D11$).
3. Važi $A = B$ ili $A \neq B$.
   4. Pretpostavimo da važi: $A = B$.
      5. Na osnovu činjenica $B \in p$ i $A = B$ važi $A \in p$.
      6. Na osnovu činjenica $A \notin p$ i $A \in p$ dobijamo kontradikciju.
   7. Pretpostavimo da važi: $A \neq B$.
      8. Na osnovu činjenica $A \neq B$ i $A \in q$ i $B \in q$ i $A \in \alpha$ i $B \in \alpha$ važi $q \in \alpha$ (aksioma $I6$).
      9. Zaključak teoreme sledi iz činjenice $q \in \alpha$.
10. Teorema je dokazana u svim slučajevima.

QED

Teoreme nad kojima je sistem ArgoGeoChecker primenjen date su u narednoj listi. Neformalni dokazi teorema i automatski generisani formalni argumenti ispravnosti dokaza, kao i dokumenti zapisani na srpskom i engleskom jeziku se nalaze na internetu[10].

Teorema 1: Ako tačka A ne pripada pravoj p, tada postoji ravan koja sadrži tačku A i pravu p.

Teorema 2: Ako tačka A ne pripada pravoj p, i ako postoje dve ravni koje sadrže tačku A i pravu p, tada su te dve ravni identične.

Teorema 3: Ako dve razne ravni imaju zajedničku tačku tada je njihov presek prava.

Teorema 4: Ako dve razne ravni imaju zajedničku pravu, tada nemaju zajedničkih tačaka van te prave.

Teorema 5: Ako su A, B, C tri nekolinearne tačke, dokazati da su svake dve od te tri tačke različite.

Teorema 6: Ako su p, q, r tri razne prave od kojih se svake dve prave seku, ali ne postoji tačka koja pripada svima trima pravama, dokazati da su p, q, r kompланarne.

[10] http://argo.matf.bg.ac.rs/downloads/formalizations/ArgoGeoChecker.zip

Теорема 7: За сваке две разне тачке A и B постоји тачка C таква да важи распоред A-C-B.
Теорема 8: Ако су P, Q, R унутрашње тачке ивица BC, AC, AB неког троугла, доказати да су оне неколинеарне.
Теорема 9: Нека су a и b две различите праве. Доказати да оне имају највише једну заједничку тачку.
Теорема 10: Дате су раван α и тачка A ван равни. Доказати да права p, која садржи тачку A, не може имати са равни α више од једне заједничке тачке.
Теорема 11: Дата је права p и ван ње тачка A. Доказати да све праве које садрже тачку A и секу праву p припадају једној равни.
Теорема 12: Дате су различите праве a и b и тачка M ван њих. Ако постоје две различите праве m и n, које садрже тачку M и секу обе праве a и b, тада праве a и b припадају једној равни. Доказати.
Теорема 13: Дат је скуп тачака A, B, C, D где тачка C припада правој AB. Доказати да се равни ABD и ACD поклапају.
Теорема 14: Ван равни α дате су три неколинеарне тачке A, B, C такве да се права AB и раван α секу у тачки M, права BC и раван α секу у тачки N и права AC и раван α секу у тачки P. Доказати да су тачке M, N и P колинеарне.
Теорема 15: Ако важи распоред A-B-C и A-D-C, онда су A, B, C, D тачке једне праве и не важи распоред B-A-D.
Теорема 16: Ако су O, A, B, C тачке једне праве и ако важи распоред A-O-B и A-O-C онда не важи распоред B-O-C.
Теорема 17: Ако важи распоред A-O-B и B-O-C и C-O-D, онда су тачке O, A, B, C и D тачке једне праве и важи распоред A-O-D.
Теорема 18: Ако важи распоред A-B-C и A-C-D, онда су A, B, C и D тачке једне праве и важи распоред A-B-D и B-C-D.

У табели 6.2 су приказани резултати добијени приликом тестирања овог система. Систем је тестиран на систему 48 AMD Opteron(tm) Processor 6168. У току проверавања неформалног доказа теореме сваки корак доказа се проверава независно од осталих. Временско ограничење од 60 секунди по кораку је коришћено у свим програмима за доказивање теорема.

Систем ArgoGeoChecker је генерисао комплетне формалне аргументе исправности доказа за 9 од 18 теорема. Када се посматра укупан број корака у свим теоремама, систем је успешно доказао 51 од укупно 65 корака, односно 78% корака. С обзиром да овај систем проверава појединачне кораке доказа ова мера боље осликава успешност овог приступа. Просечно време провере доказа теореме и генерисања формалних аргумената исправности доказа у језицима

Редни број теореме	Број корака у доказу	Доказаних Vampire, E	Доказаних ArgoCLP	Време (у секундама)
1	4	4	4	9
2	4	4	4	8
3	3	3	3	8
4	4	4	4	82
5	1	1	1	16
6	2	2	2	8
7	16	13	13	136
8	1	0	0	127
9	1	1	1	7
10	1	1	1	8
11	2	2	2	73
12	6	6	5	173
13	4	3	2	128
14	8	7	6	145
15	2	1	1	128
16	1	0	0	127
17	2	1	1	134
18	3	1	1	128

Табела 6.2: *Резултати примене система ArgoGeoChecker. Број корака у доказу теореме, број доказаних корака када се користе резолуцијски доказивачи, број доказаних корака доказа када се користи кохерентни доказивач и време доказивања.*

интерактивних доказивача Isabelle и Coq, као и генерисања читљивих доказа на српском и енглеском језику је 80 секунди. Одређене теореме су наведене без доказа (у табели су означене са једним кораком - само закључак теореме). У тим случајевима ArgoGeoChecker је коришћен за налажење доказа теореме. Од пет теорема које су дате без доказа, систем је успео да нађе потпун доказ (без усмеравања од стране корисника) за три теореме. Сматрамо да је комплетан аргумент исправности доказа генерисан само за оне доказе код којих су сви кораци успешно доказани. У пракси би вештији корисник могао користити чак и непотпун аргумент исправности доказа, тако што би га допунио доказима недостајућих корака. У оквиру ових експеримената извршена је провера само оригиналних доказа записаних у уџбеницима. Нису разматране различите верзије доказа једне исте теореме нити су постојећи докази проширени додатним корацима. Допуњавање неформалних доказа новим корацима би свакако побољшало успешност овог приступа.

6.3 Геометрија Авигада

Еуклидови Елементи представљају једну од најстаријих аксиоматски заснованих књига којој је упућен значајан број критика првенствено због коришћења илустрација у њеним доказима. Авигад је креирао аксиоматски систем чијом употребом настоји да покаже да је коришћење илустрација „безбедно" и контролисано правилима која су описана његовим аксиоматским системом.

Бен Нортроп је имплементирао програм за проверавање доказа Еуклидових постулата E-proof-checker. Приступ који се користи у том програму се налази између интерактивног и аутоматског доказивања теорема. Наликује неформалном доказивању теорема, као што је већ речено у поглављу 5.3.

У Еуклидовим Елементима докази се деле на две фазе: фаза конструкције и фаза доказивања. У складу са тим, Авигадов аксиоматски систем је подељен на две групе аксиома: правила за конструисање и правила извођења. Правила за конструисање су сва правила којима се креирају нови објекти у доказу и приликом илустровања доказа. Правила извођења су правила које уводе нова својства већ уведених објеката. Ову поделу пратиће и систем E-proof-checker, у ком корисник задаје доказ у коме је јасно одвојена фаза конструкције објеката који се користе у доказу, од фазе доказивања својстава тих објеката. Након тога се посебно проверавају кораци и у једној и у другој фази (са одговарајућим скупом аксиома).[11]

Након покретања програма E-proof-checker, прво се проверава да ли су правила за конструисање исправно наведена. Након тога се покреће аутоматски процес који служи само да провери да ли се својства која су наведена у другом делу доказа могу извести применом правила извођења (у оквиру Авигадовог аксиоматског система). Поступак доказивања користи алгоритам уланчавања унапред, генерише сва могућа својства над иницијално уведеним објектима, и тек на крају свог извршавања проверава да ли су доказана тражена својства.

Главно ограничење и одступање од Авигадовог аксиоматског система односи се на то што систем E-proof-checker не подржава сабирање дужи, углова и слично. То ограничење озбиљно утиче на употребљивост система и доводи до тога да E-proof-checker не успева да докаже ни један постулат до краја.

У програму E-proof-checker, правила конструисања се не користе у процесу аутоматског доказивања теореме. Користе се само у првој фази доказивања која се обавља „ручно" уз помоћ корисника и служе увођењу свих објеката који

[11] Сви објекти који се користе у другој фази морају експлицитно бити наведени у првој фази доказа).

се користе у доказу (како објеката који су уведени тврђењем теореме, тако и објеката који се уводе на почетку доказа). Након тога се у фази доказивања користе сва преостала правила тако да процес доказивања овим системом више подсећа на интерактивно доказивање него на аутоматско доказивање теорема.

6.3.1 Аксиоматски систем

Нека од правила конструисања у оквиру Авигадовог аксиоматског система се налазе у наредној листи. Комплетан списак правила Авигадовог аксиоматског система (и правила конструисања и правила извођења), записаних у TPTP формату, налази се у додатку C. Приликом записивања правила у тексту тезе користиће се стандардне ознаке које се користе у геометрији, тачке ће бити означене великим словима а праве малим словима. Приликом задавања аксиома, Авигад је формулисао саму аксиому, предуслове потребне да би се та аксиома применила и закључке који ће важити након примене те аксиоме.

Points 1 Постоји тачка A [различита од ...].

Предуслови: Нема их.

Закључак: [Тачка A је различита од ...]

Points 3 Постоји тачка A која припада правој l која се налази између тачака B и C [различита од ...].

Предуслови: Тачка B припада правој l, тачка C припада правој l, $B \neq C$, [права l је различита од правих ...]

Закључак: Тачка A припада правој l, A се налази између B и C, [тачка A је различита од ...]

Lines and Circles 1 Постоји права l која пролази кроз тачке A и B.

Предуслови: $A \neq B$.

Закључак: Тачка A припада правој l, тачка B припада правој l.

Приликом записивања правила конструисања у оквиру система ArgoGeoChecker, предуслови и закључци правила су записани као део самих правила. На тај начин правило `Points 3` се записује на следећи начин (формула је записана у TPTP формату):

```
fof(ax_points3, axiom, (! [L,A,B] :
((line(L) & point(A) & point(B) & on(A,L) & on(B,L) & A != B)
=> (? [C] : (point(C) & on(C,L) & bet(A,C,B))))).
```

Конструкције *[различита од...]* нису записане, тј. аксиоме су записане без тих напомена. Без обзира на то што правила записана у том облику могу знатно да утичу на простор претраге, систем ArgoGeoChecker се чак и са њима показао користан. Коришћен је за проверавање појединачних корака доказа и показало се да је у стању да исправно идентификује одговарајуће правило које је коришћено у тим корацима (ако се корак односи управо на примене правила конструисања).

Правила извођења личе на стандардне аксиоме Хилбертовог аксиоматског система и неће бити посебно разматрана. Нека од правила извођења се налазе у наредној листи:

Generalities 1 Ако важи $A \neq B$, A припада l и B припада l, A припада m и B припада m, онда важи $l = m$.

Between 2 Ако је B између A и C, A припада l и B припада l, онда и C припада l.

6.3.2 Примена система за проверавање неформалних доказа

Систем за проверу неформалних доказа теорема описан у поглављу 5.2 наликује систему E-proof-checker и биће приказана његова употреба за проверавање доказа неколико примера из Авигадовог рада [3]. У записивању Еуклидових постулата и њихових доказа коришћен је запис приказан у том раду уместо оригиналних Еуклидових доказа.

Пример 1: Први Еуклидов постулат. Претпоставимо да су A и B различите тачке. Конструисати тачку C тако да важи $\overline{AB} = \overline{BC}$ и $\overline{BC} = \overline{CA}$.
Доказ:
Нека је α круг са центром у A који пролази кроз B.
Нека је β круг са центром у B који пролази кроз A.
Нека је C тачка пресека кругова α и β.
Онда важи $\overline{AB} = \overline{AC}$ (јер су полупречници круга α).
Онда важи $\overline{BA} = \overline{BC}$ (јер су полупречници круга β).
Отуда важи $\overline{AB} = \overline{BC}$ и $\overline{BC} = \overline{CA}$.
Q.E.F.

На основу овог доказа може се записати следећи неформални доказ (записан у синтакси система представљеном у овој тези, уместо ознаке $\overline{AB} = \overline{CD}$ пишемо $cong(A, B, C, D)$):

```
! [A,B] : (point(A) & point(B) & A != B)
? [K1] : (circle(K1) & center(A,K1) & onc(B,K1))
? [K2] : (circle(K2) & center(B,K2) & onc(A,K2))
? [C] : (point(C) & onc(C,K1) & onc(C,K2) & intersectscc(K1,K2))
(cong(A,B,B,C))
(cong(B,A,B,C))
(cong(A,B,B,C) & cong(B,C,C,A))
```

Позивањем система ArgoGeoChecker (коме су прослеђене *све* аксиоме Авигадовог аксиоматског система) добија се следећи излаз (и генерише се формални аргумент исправности доказа):

```
th_prop1_01.p| vampire|  0.04|  0.19|                          ax_lines_and_circles2 |      0.00||
th_prop1_02.p| vampire|  0.04|  0.25|                          ax_lines_and_circles2 |      0.00||
th_prop1_03.p| vampire|  0.17|  0.32|     ax_intersections6 ax_generalities3 ax_rule5 |      0.02||
th_prop1_04.p| vampire|  0.22|  0.27|   ax_metric3 ax_segment3_2 ax_cong_transitivity |      0.43||
th_prop1_05.p| vampire|  0.06|  0.25|                                    ax_segment3_2 |      0.03||
th_prop1_06.p|       e|  0.20|  1.23|   ax_cong_transitivity ax_metric3 ax_segment3_2 |      1.22||
Seconds: 10
```

Како се сваки корак неформалног доказа (осим првог) појединачно проверава, у излазу су приказане аксиоме koje се користе у оквиру доказа текућег корака. У сваком кораку наведен је резолуцијски доказивач који је нашао мањи скуп релевантних аксиома, време извршавања и једног и другог резолуцијског доказивача, скуп аксиома и време извршавања кохерентног доказивача. Видимо да је систем ArgoGeoChecker успешно валидирао доказ првог Еуклидовог постулата.

Авигад истиче да, приликом примене овог постулата у наредним доказима, Еуклид често користи чињеницу да је тачка C различита од тачака A и B. Као и чињеницу да тачка C не припада правој одређеној тачкама A и B. Због тога експлицитно уводи наредне две последице:

Пример 2: Последица првог Еуклидовог постулата. Претпоставимо да су A и B различите тачке и да је дата тачка C тако да важи $\overline{AB} = \overline{BC}$ и $\overline{BC} = \overline{CA}$. Показати да је тада $C \neq A$ и $C \neq B$.

Доказ:

Претпоставимо да важи $C = A$.

Онда важи $A = B$.

Контрадикција.

Онда мора да важи $C \neq A$.

Претпоставимо да важи $C = B$.

Онда важи $A = B$.

Контрадикција.

Онда мора да важи $C \neq B$.
Q.E.D.

Докази који се могу проверити уз помоћ система ArgoGeoChecker не подржавају конструкцију „претпоставимо супротно" па овај пример третирамо као да немамо на располагању доказ теореме. Тада доказ има само један корак. Први ред доказа су премисе и други ред корака су закључци тврђења које се доказује.

```
![A,B,C,L]:(point(A) & point(B) & A!=B & point(C) & cong(A,B,B,C) & cong(B,C,C,A))
(C!=A & C!=B)
```

Систем ArgoGeoChecker је успео да докаже ову последицу. Излаз из система и генерисани доказ на српском језику се налазе у наставку текста.

```
th_prop1aux1_01.p|      e|  0.04|  0.26|                  ax_cong_eq1 ax_cong_symmetry |     0.03||
Seconds: 6
```

Teorema 6.3 (th_prop1aux1_01.) *Pod pretpostavkom da važi $A \neq B$ i $AB \cong BC$ i $BC \cong CA$ pokazati da važi $C \neq A$ i $C \neq B$.*

Dokaz:

1. Na osnovu činjenice $AB \cong BC$ važi $BC \cong AB$ (aksioma *cong_symmetry*).
2. Na osnovu činjenice $BC \cong CA$ važi $CA \cong BC$ (aksioma *cong_symmetry*).
3. Na osnovu činjenice $A \neq B$ važi $B \neq A$.
4. Važi $A = C$ ili $A \neq C$.
   5. Pretpostavimo da važi: $A = C$.
      6. Na osnovu činjenica $CA \cong BC$ i $A = C$ važi $AA \cong BA$.
      7. Na osnovu činjenice $AA \cong BA$ važi $B = A$ (aksioma *cong_eq*1).
      8. Na osnovu činjenica $B \neq A$ i $B = A$ dobijamo kontradikciju.
   9. Pretpostavimo da važi: $A \neq C$.
   10. Na osnovu činjenice $A \neq C$ važi $C \neq A$.
   11. Važi $B = C$ ili $B \neq C$.
      12. Pretpostavimo da važi: $B = C$.
         13. Na osnovu činjenica $BC \cong AB$ i $B = C$ važi $BB \cong AB$.
         14. Na osnovu činjenice $BB \cong AB$ važi $A = B$ (aksioma *cong_eq*1).
         15. Na osnovu činjenica $A \neq B$ i $A = B$ dobijamo kontradikciju.
      16. Pretpostavimo da važi: $B \neq C$.

17. Na osnovu činjenice $B \neq C$ važi $C \neq B$.

18. Zaključak teoreme sledi iz činjenica $C \neq A$ i $C \neq B$.

19. Teorema je dokazana u svim slučajevima.

20. Teorema je dokazana u svim slučajevima.

QED

Пример 3: Последица првог Еуклидовог постулата. Претпоставимо да су A и B различите тачке које леже на правој l и да је дата тачка C тако да важи $\overline{AB} = \overline{BC}$ и $\overline{BC} = \overline{CA}$. Тада тачка C не припада l.

Доказ:

Претпоставимо да важи C припада l.
Претпоставимо да важи да је тачка A између тачака C и B.
Онда важи $\overline{CA} < \overline{BC}$. Контрадикција.
Претпоставимо да важи $C = A$. Онда мора да важи $A = B$. Контрадикција.
Претпоставимо да важи да је тачка C између тачака A и B.
Онда важи $\overline{CA} < \overline{AB}$. Контрадикција.
Претпоставимо да важи $C = B$. Онда мора да важи $A = B$. Контрадикција.
Претпоставимо да важи да је тачка B између тачака A и C.
Онда важи $\overline{AB} < \overline{BC}$. Контрадикција.
Контрадикција.
Q.E.D.

Слично као приликом доказивања прве последице, систем не подржава записивање конструкције „претпоставимо супротно” и овај пример тестирамо као да немамо доказ.

```
![A,B,C,L]:(point(A) & point(B) & A != B & line(L) & on(A,L) & on(B,L) & point(C)
& cong(A,B,B,C) & cong(B,C,C,A))
(non(C,L))
```

Овако записан, овај пример се не доказује. Када му се додају информације добијене првом последицом добија се наредни неформални доказ:

```
![A,B,C,L]:(point(A) & point(B) & A != B & line(L) & on(A,L) & on(B,L) & point(C)
& cong(A,B,B,C) & cong(B,C,C,A))
(C!=A)
(C!=B)
(non(C,L))
```

који се доказује системом:

```
th_prop1aux2_01.p| vampire|  0.04|  0.24|                      ax_metric1_1 ax_cong_zero1 |   0.01||
th_prop1aux2_02.p| vampire|  0.04|  0.16|                                    ax_cong_eq1 |   0.01||
th_prop1aux2_03.p|       e| 59.90| 25.93|ax_bet1 ax_branch_on ax_lines_and_circles2
    ax_generalities3 ax_circle1 ax_segment3_1 ax_metric3 ax_cong_transitivity ax_false_bet |   1.71||
Seconds: 107
```

Доказ другог Еуклидовог постулата је знатно компликованији од претходних примера. Доказ који је приказан у оквиру Авигадовог аксиоматског система је наведен у наставку текста. Након тога су приказани неформални доказ и излаз из система ArgoGeoChecker.

Пример 4: Други Еуклидов постулат. Нека су B и C две различите тачке на правој L и нека је A тачка различита од B и C. Конструисати тачку F тако да важи $\overline{AF} = \overline{BC}$.

Доказ:

Применом Постулата 1 примењеним над тачкама A и B, конструишимо тачку D тако да је D различита од тачака A и B и важи $\overline{AB} = \overline{BD}$ и $\overline{BD} = \overline{DA}$.

Нека је M права која пролази кроз тачке D и A.

Нека је N права која пролази кроз тачке D и B.

Нека је α круг са центром у B који пролази кроз C.

Нека је G тачка пресека праве N и круга α која продужава сегмент од тачке D до тачке B.

Онда важи $\overline{DG} = \overline{DB} + \overline{BG}$.

Отуда важи $\overline{DG} = \overline{DA} + \overline{BG}$ (јер је $\overline{DA} = \overline{DB}$).

Отуда важи $\overline{DA} < \overline{DG}$.

Нека је β круг са центром у D који пролази кроз G.

Отуда важи да је A унутар круга β (јер је D центар круга и важи $\overline{DA} < \overline{DG}$).

Нека је F тачка пресека круга β и праве M која продужава сегмент од тачке D до тачке A.

Онда важи $\overline{DF} = \overline{DA} + \overline{AF}$.

Онда важи $\overline{DF} = \overline{DG}$ (јер су полупречници круга β).

Отуда важи $\overline{DA} + \overline{AF} = \overline{DA} + \overline{BG}$.

Отуда важи $\overline{AF} = \overline{BG}$.

Онда важи $\overline{BG} = \overline{BC}$ (јер су полупречници круга α).

Отуда важи $\overline{AF} = \overline{BC}$.

Q.E.F.

Неформални доказ записан у систему ArgoGeoChecker одступа од Авигадовог доказа у томе што у премисама теореме не наводи праву L која се не користи у доказу.

```
! [A,B,C] : (point(A) & point(B) & point(C) & B!=C & A!=B & A!=C)
? [D] : (point(D) & D!=A & D!=B & cong(A,B,B,D) & cong(B,D,D,A))
? [M] : (line(M) & on(D,M) & on(A,M))
? [N] : (line(N) & on(D,N) & on(B,N))
? [K1] : (circle(K1) & center(B,K1) & onc(C,K1))
? [G] : (point(G) & on(G,N) & onc(G,K1) & bet(D,B,G))
(segment_add(D,B,B,G,D,G))
(segment_add(D,G,D,A,B,G))
(cong_less(D,A,D,G))
? [K2] : (circle(K2) & center(D,K2) & onc(G,K2))
(inside(A,K2))
? [F] : (point(F) & onc(F,K2) & on(F,M) & bet(D,A,F))
(segment_add(D,A,A,F,D,F))
(cong(D,F,D,G))
(segment_add(D,A,B,G,D,F))
(cong(A,F,B,G))
(cong(B,G,B,C))
(cong(A,F,B,C))
```

Још једна разлика између система E-proof-checker и ArgoGeoChecker је у томе што у систему представљеном у овој тези, фаза конструкције не мора да буде строго одвојена од фазе извођења већ могу да се преклапају као у овом примеру. Осим тога, целокупан Авигадов аксиоматски систем се користи у току целог процеса доказивања. Резолуцијски доказивачи се користе за смањење тог скупа тако да систем не мора да проверава који кораци уводе нове објекте а који само нова својства већ постојећих објеката.

```
th_prop2_01.p|        |                                                                    |          ||
th_prop2_02.p| vampire|  0.04|  0.26|                              ax_lines_and_circles1 |      0.06||
th_prop2_03.p| vampire|  0.04|  0.25|                              ax_lines_and_circles1 |      0.03||
th_prop2_04.p| vampire|  0.04|  0.25|                              ax_lines_and_circles2 |      0.03||
th_prop2_05.p|       e| 59.86| 17.05|       ax_intersections5 ax_generalities3 ax_bet1 |      0.26||
th_prop2_06.p| vampire|  0.02|  0.17|                                        ax_segment1 |      0.01||
th_prop2_07.p|        |                                                                    |          ||
th_prop2_08.p| vampire|  0.32|  0.51|           ax_metric3 ax_cong_less1 ax_cong_less3 |     15.10||
th_prop2_09.p| vampire|  0.14|  0.26|                     ax_bet1 ax_lines_and_circles2 |      3.02||
th_prop2_10.p| vampire|  0.09|  0.29|                                   ax_segment4_1 |      0.05||
th_prop2_11.p|       e| 59.65|  5.46|                         ax_intersections5 ax_bet1 |      3.25||
th_prop2_12.p| vampire|  0.05|  0.17|                                        ax_segment1 |      0.03||
th_prop2_13.p| vampire|  0.23|  0.49|                                      ax_segment3_2 |      0.25||
th_prop2_14.p| vampire|  0.64|  0.94|ax_cong_symmetry ax_metric3 ax_cong_transitivity
                                                 ax_segment_add2 ax_segment_add7 |Time limit||
th_prop2_15.p| vampire|  0.67|  0.93|  ax_metric3 ax_cong_transitivity ax_segment_add7 |Time limit||
th_prop2_16.p| vampire|  0.23|  0.49|                                      ax_segment3_2 |      0.02||
th_prop2_17.p| vampire|  0.23|  0.27|            ax_cong_symmetry ax_cong_transitivity |      5.20||
Seconds: 129
```

У излазу из система примећује се да, од 17 задатих корака, систем успешно доказује њих 13. Први корак доказа је примена првог Еуклидовог постулата.

Како тренутно систем не подржава коришћење претходно доказаних тврђења, тај корак није успешно доказан. У оквиру E-proof-checker система тај корак је расписан тако да имитира доказ првог Еуклидовог постулата.

Резултати ових примера су обећавајући али треба бити обазрив приликом њиховог тумачења. Почетни постулати и њихове последице су релативно једноставни, док су наредни постулати знатно тежи тако да се не очекује да ће овај систем, у свом основном облику, бити у стању да их докаже.

6.4 Applications of the Proving Framework — Summary

The framework described in the previous chapter can be used with any coherent logic theory. In this chapter, its application on three different axiomatic systems is presented: axiomatic system of Tarski, Hilbert's axiomatic system, and Avigad's axiomatic system. The framework will be used to generate proofs for theorems of Tarski's axiomatic system, and to check informal proofs of theorems of Hilbert's and Avigad's axiomatic system. All three axiomatic systems, and theorems within them, are already mostly in coherent logic form or can easily be translated to the coherent form.

Tarski's axiomatic system A case-study on Tarski's axiomatic system is based on the first part of a book on foundations of geometry: *Metamathematische Methoden in der Geometrie*, by Wolfram Schwabhauser, Wanda Szmielew, and Alfred Tarski [79]. The theory is described in terms of first-order logic, it uses only one primitive object — the point, has only two primitive predicates, and only eleven axioms. There are already three existing computer developments of this book: one by Quaife based on automated theorem proving [1], one by Beeson and Wos based on automated theorem proving [5, 6], and one by Narboux and Braun based on interactive theorem proving [67, 18] in proof assistant Coq. These developments were an excellent starting point for the experiments presented in this thesis.

Several experiments were conducted on the theorems from the first 12 chapters of SST that belong to plane geometry (just like in the Coq formalization). Out of the 203 theorems, 179 of them belong to plane geometry. In the first experiment, only theorems from the book are used, in the second experiment additional lemmas from the existing Coq formalization of the book are used, and in the third one specific dependency lists are used (derived from the Coq formalization for each theorem). The results show that 37% of the theorems from the book can be automatically proven

(with readable and machine verifiable proofs generated) without any guidance, and with additional lemmas this percentage rises to 42%.

Hilbert's axiomatic system Informal proofs, found in high school geometry curricula, can be automatically verified using ArgoGeoChecker system. The system allows for automated checking of individual steps of informal proofs and generates formal, computer verifiable argument of a proof along with an argument written in Serbian and English. Its use on a handful of proofs from high school textbooks is demonstrated.

Avigad's axiomatic system Proofs found in Euclid's "Elements" are often criticized for theirs use of diagrams. The axiomatic system E, developed by Avigad, Dean and Mumma is created for justification of Euclid's proofs. The system described in this thesis is used for verification of the several informal proofs and theirs auxiliaries that are found in Avigad's paper [3] using axiomatic system E. Some of them completely and one of them partially. Although the proving system is fairly successful on those examples, later theorems from Elements are considerably harder and some additional tools will be required for them.

7

Закључци и даљи рад

Аксиоматизација геометрије је тема која се разматра од самих почетака развијања математике. У овој тези су разматрани различити аксиоматски системи еуклидске геометрије и њихово коришћење у оквиру програма за аутоматско доказивање теорема. Креиран је програм за аутоматско доказивање теорема који се базира на кохерентној логици, дијалект кохерентне логике, као и систем за аутоматско доказивање теорема и проверавање неформалних доказа.

Кохерентни доказивач теорема ArgoCLP Један од доприноса ове тезе је кохерентни доказивач теорема ArgoCLP (развијен у сарадњи са још два аутора). Може се користити над произвољном теоријом чије су аксиоме и теореме записане у кохерентној форми. Подржава TPTP формат улаза података и генерише доказ записан у XML формату из ког се даље генеришу докази записани на различитим формалним и природним језицима.

Доказивач може бити коришћен за доказивање једноставнијих теорема, или као асистент у процесу доказивања тежих теорема тако што ће се користити за доказивање погодно одабраних подциљева комплексних тврђења. Осим тога, једноставност доказивача и различити излазни формати доказа омогућавају многе примене у пројектима формализације математичког знања, али и у образовању, као што је приказано у тези.

Убрзање рада доказивача се може добити једноставним (аутоматским) модификацијама аксиоматског система (које смањују број једноставних чињеница и број уведених сведока у доказима). Осим ефикасности, те модификације доприносе и читљивости добијених доказа. Откривање предикатских симбола са парцијалним својством симетричности (који су симетрични само по неким позицијама, као што су предикатски симболи *bet* і *cong*) и искоришћавање тих особина би био погодан домен за доказивач

ArgoCLP. Информације тог типа би могле и додатно да убрзају рад доказивача и учине генерисане доказе још прегледнијим.

Дијалект кохерентне логике Дијалект кохеретне логике, развијен у овој тези и репрезентација доказа њиме описана, представља основу заједничког формата доказа који може бити генерисан од стране различитих аутоматских доказивача теорема и који се може једноставно трансформисати у доказе записане на различитим језицима. Скуп XSL алата омогућава превођење овог формата у доказе записане на језицима програма за интерактивно доказивање теорема Isabelle и Coq, као и у доказе записане на природном језику (енглеском и српском). Додатна предност овог формата лежи у његовој једноставности и чињеници да доказе записане у том формату може разумети и сам корисник (без трансформисања самих доказа). Осим тога, додавање нових XSL трансформација и формирање доказа на другим (формалним и неформалним) језицима је једноставно и не захтева измене над самим доказивачем.

Текућа верзија овог формата не подржава употребу функцијских симбола арности веће од 0. У будућности планирамо да уведемо употребу функцијских симбола и у доказивач ArgoCLP, и у формат записа доказа чиме ће се још више повећати читљивост добијених доказа.

Систем за аутоматску и интерактивну формализацију Систем описан у овој тези је намењен аутоматском доказивању теорема кохерентне логике. Систем користи неколико различитих врста доказивача теорема: аутоматски доказивачи теорема се користе да од великог скупа аксиома и теорема филтрирају мали подскуп формула довољан за доказивање одређеног тврђења; кохерентни доказивач теорема се користи да, применом минималног скупа формула, изгенерише формалне и читљиве доказе; а интерактивни доказивачи теорема се користе да потврде исправност генерисаних доказа. Систем је примењен на три аксиоматска система еуклидске геометрије: аксиоматски систем Тарског, Хилбертов аксиоматски систем и аксиоматски систем Авигада.

Аутоматско доказивање теорема геометрије Систем описан у овој тези комбинује врхунске аутоматске доказиваче теорема и интерактивне доказиваче теорема и заједно их користи у процесу формализације математике. Примењен на једну од најпознатијих математичких књига, књигу Тарског, доказује 37% теорема потпуно аутоматски, притом генеришући читљиве и машински провериве доказе. Узевши у обзир да су докази добијени потпуно аутоматски, без икаквог навођења, добијени резултати су солидни али уједно и

сугеришу да постоји простор за повећање ефикасности. Информације добијене од резолуцијских доказивача могу бити детаљније анализиране и могу бити искоришћене за прецизније навођење кохерентног доказивача. У плану нам је да се фокусирамо на употребу аксиома искључења трећег $\forall\vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$ из разлога што њено коришћење знатно утиче на величину простора претраге. Описани систем може бити примењен и над сличним математичким теоријама, како за генерисање нових формалних доказа, тако и за поједностављивање постојећих доказа (у оквиру интерактивних доказивача теорема). У будућности имамо у плану и доказивање теорема из књиге Тарског које у својим доказима користе аксиому непрекидности.

Аутоматско проверавање неформалних доказа теорема Систем за проверу неформалних доказа теореме, ArgoGeoChecker, примењен је на доказима и решењима десетак теорема и задатака из средњошколских уџбеника. Приказан је његов начин употребе и практичност у конкретним ситуацијама када решења задатака садрже само упутство са минималним бројем корака доказа. У неким случајевима, кораци које систем није доказао у себи крију примене претходно доказаних теорема. У оквиру овог експеримента, приликом провере доказа коришћене су само аксиоме, није подржано коришћење претходно доказаних теорема. Проширење система тако да подржава и ову опцију је могуће, чиме би се још боље илустровала реална примена овог система у образовању. У поређењу са програмима за интерактивно доказивање теорема, систем ArgoGeoChecker је далеко једноставнији за употребу, не захтева познавање специфичних језика за интерактивно доказивање (нити тактика које се користе у њима, без којих је интерактивно доказивање практично немогуће), што га чини погодним за коришћење међу почетницима у формалном доказивању теорема. Ипак, систем захтева формулацију геометријских тврђења у прецизној синтакси што је свакако савладиво за мотивисане ученике средњих школа.

Приказана је употреба система за аутоматско проверавање неформалних доказа на неколико доказа наведених у првој књизи Еуклидових „Елемената". Коришћен је аксиоматски систем Авигада који је мотивисан Еуклидовим „Елементима" и намењен је управо аутоматској провери неформалних доказа. Систем је успео да докаже неколико почетних примера на којима је илустрован аксиоматски систем Авигада. Показана је практична примена овог приступа, али тренутна верзија овог система, која користи само резолуцијске доказиваче и кохерентни доказивач, није довољно ефикасна. За доказивање тежих

Еуклидових постулата потребно је користити моћније алате од резолуцијских доказивача теорема, што је планирано за наставак ове тезе.

7.1 Conclusions and Future Work — Summary

Axiomatization of geometry has always been an interesting field for mathematicians. In this thesis different axiomatic systems for Euclidean geometry are considered, formalized and used within programs for automated theorem proving. Automated theorem prover for coherent logic has been developed, as well as a coherent logic vernacular, and a system for automated theorem proving and verification of informal proofs.

Coherent theorem prover ArgoCLP A theorem prover ArgoCLP uses coherent logic as its underlying logic and can be used for any theory with coherent axioms and for conjectures in the coherent form. It supports input in TPTP form and generates proof trace written in XML form that can be translated into proofs of different formal and natural languages. Generated proofs are readable and similar to proofs found in mathematical textbooks. The prover, on its own, can be used for proving simple theorems or theorems of moderate hardness, and also as an assistan for proving appropriately chosen subgoals of complex conjectures or with integration with other theorem provers. The simplicity of the prover and different output formats enable many applications in formalization of mathematics and in education.

Coherent logic vernacular Coherent logic vernacular, and a proof format that describes it, can be used by different automated theorem provers and can produce proofs in different languages. It is very flexible thanks to the interchange XML format and a set of simple XSLT style-sheets that generate proofs in the languages of interactive theorem provers Isabelle and Coq, as well as readable proofs written in Serbian and English. Its advantage is that additional XSLT style-sheets (for additional target formats) can be developed without changing the prover. To further improve readability, we will consider using function symbols and work on extending coherent logic vernacular and coherent logic prover ArgoCPL to deal with function symbols.

A system for automated and interactive formalization of geometry Several different theorem provers are used within this thesis: automated theorem provers are used to filtrate large sets of axioms and theorems into a number of formulas

used while proving specific theorem; coherent theorem prover is used to generate formal and readable proofs; and interactive theorem provers are used to validate such proofs. System described in this thesis is used on three axiomatic systems for Euclidean geometry, Tarski's axiomatic system, Hilbert's axiomatic system and axiomatic system given by Jeremy Avigad.

Automated theorem proving In this thesis state-of-the-art automated theorem provers are used in formalization tasks and within a wider framework that combines different sorts of provers. For one classical twentieth century mathematical book, for 37% theorems, readable and machine verifiable proofs are generated completely automatically. This shows that automated theorem proving can provide significant help in formalization tasks within proof assistants (of course, typically in proving less difficult theorems). For future work we are planning to further explore potentials for full automation, for example to use information from resolution provers more deeply and guide our coherent prover more precisely. In particular, we would focus on the axioms $\forall\vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$ that can heavily extend the search space. We will explore ways for using our framework for simplifying existing proofs within proof assistants (by using dependencies from automated theorem provers). We will also try to use our framework to tackle theorems from SST that require using the continuity axiom (i.e., the schema of axioms).

Automated verification of informal proofs The system for automated verification of informal proofs, ArgoGeoChecker, allows for automated checking of informal proofs that are often found in high school geometry curricula. The program is very simple to use and does not require a user to have any prior experience with automated theorem proving software. In addition to being a useful educational tool, the program could also be used by mathematicians for producing new original proofs. The system is used on a handful of proofs from high school textbooks, and on an axiomatic system developed by Jeremy Avigad with Euclid's "Elements". In its original form the system is not very efficient and there is a lot of room for improvement. Currently the system does not support the use of previously proven theorems, theorem provers are provided only with axioms, and extension of the system in this way would significantly improve usefulness of this system.

Додатак А

Аксиоматски систем Тарског

Аксиоматски систем Тарског наведен у књизи *Metamathematische Methoden in der Geometrie* се налази у наставку текста. Наведене су и све дефиниције које се користе у књизи распоређене по поглављима у којима су уведене. Приликом записивања аксиома и дефиниција извршена је трансформација у кохерентну форму у случајевима када је то било потребно. Све формуле су записане у TPTP формату.

Поглавље 1 — Аксиоме Тарског

```
fof(ax_1, axiom, (! [A,B] : ((cong(A,B,B,A)))).
fof(ax_2, axiom, (! [A,B,P,Q,R,S] : ((cong(A,B,P,Q) & cong(A,B,R,S)) =>
                  cong(P,Q,R,S)))).
fof(ax_3, axiom, (! [A,B,C] : ((cong(A,B,C,C)) => (A=B)))).
fof(ax_4, axiom, (! [A,B,C,Q] : (? [X] : (bet(Q,A,X) & cong(A,X,B,C))))).
fof(ax_5, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((A != B & bet(A,B,C)
                 & bet(A1,B1,C1) & cong(A,B,A1,B1) & cong(B,C,B1,C1)
                 & cong(A,D,A1,D1) & cong(B,D,B1,D1)) => cong(C,D,C1,D1)))).
fof(ax_6, axiom, (! [A,B] : ((bet(A,B,A)) => A=B))).
fof(ax_7, axiom, (! [A,B,C,P,Q] : ((bet(A,P,C) & bet(B,Q,C)) =>
                 (? [X] : (bet(P,X,B) & bet(Q,X,A)))))).
fof(ax_8, axiom, (? [A,B,C] : (nbet(A,B,C) & nbet(B,C,A) & nbet(C,A,B)))).
fof(ax_9, axiom, (![P,Q,A,B,C] : ((P!=Q & cong(A,P,A,Q) & cong(B,P,B,Q)
                 & cong(C,P,C,Q)) => (bet(A,B,C) | bet(B,C,A) | bet(C,A,B))))).
fof(ax_10, axiom,(! [A,B,C,D,T] : ((bet(A,D,T) & bet(B,D,C) & A!=D) =>
                 (? [X,Y] : (bet(A,B,X) & bet(A,C,Y) & bet(X,T,Y)))))).
```

Поглавље 2

```
fof(ax_2_10_1, axiom, (! [A,B,C,D,A1,B1,C1,D1] : (afs(A,B,C,D,A1,B1,C1,D1)
                  => (bet(A,B,C) & bet(A1,B1,C1) & cong(A,B,A1,B1)
                  & cong(B,C,B1,C1) & cong(A,D,A1,D1) & cong(B,D,B1,D1))))).
```

```
fof(ax_2_10_2, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((bet(A,B,C) & bet(A1,B1,C1)
                & cong(A,B,A1,B1) & cong(B,C,B1,C1) & cong(A,D,A1,D1)
                & cong(B,D,B1,D1)) => afs(A,B,C,D,A1,B1,C1,D1)))).
fof(ax_branch_bet, axiom, (! [A,B,C] : (bet(A,B,C) | nbet(A,B,C)))).
fof(ax_false_bet, axiom, (! [A,B,C] : ((bet(A,B,C) & nbet(A,B,C)) => $false))).
fof(ax_branch_afs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((afs(A,B,C,D,A1,B1,C1,D1)
                | nafs(A,B,C,D,A1,B1,C1,D1))))).
fof(ax_false_afs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((afs(A,B,C,D,A1,B1,C1,D1)
                & nafs(A,B,C,D,A1,B1,C1,D1)) => $false))).
```

Поглавље 3

```
fof(ax_3_8_1, axiom, (! [A,B,C,D] : (bet4(A,B,C,D) => (bet(A,B,C) & bet(A,B,D)
                & bet(A,C,D) & bet(B,C,D))))).
fof(ax_3_8_2, axiom, (! [A,B,C,D] : ((bet(A,B,C) & bet(A,B,D) & bet(A,C,D)
                & bet(B,C,D)) => bet4(A,B,C,D)))).
fof(ax_branch_bet4, axiom, (! [A,B,C,D] : ((bet4(A,B,C,D) | nbet4(A,B,C,D))))).
fof(ax_false_bet4, axiom, (! [A,B,C,D] : ((bet4(A,B,C,D) & nbet4(A,B,C,D))
                => $false))).
```

Поглавље 4

```
fof(ax_4_1_1, axiom, (! [A,B,C,D,A1,B1,C1,D1] : (ifs(A,B,C,D,A1,B1,C1,D1)
                => (bet(A,B,C) & bet(A1,B1,C1) & cong(A,C,A1,C1)
                & cong(B,C,B1,C1) & cong(A,D,A1,D1) & cong(C,D,C1,D1))))).
fof(ax_4_1_2, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((bet(A,B,C) & bet(A1,B1,C1)
                & cong(A,C,A1,C1) & cong(B,C,B1,C1) & cong(A,D,A1,D1)
                & cong(C,D,C1,D1)) => (ifs(A,B,C,D,A1,B1,C1,D1))))).
fof(ax_branch_ifs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((ifs(A,B,C,D,A1,B1,C1,D1)
                | nifs(A,B,C,D,A1,B1,C1,D1))))).
fof(ax_false_ifs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((ifs(A,B,C,D,A1,B1,C1,D1)
                & nifs(A,B,C,D,A1,B1,C1,D1)) => $false))).
fof(ax_4_4_1, axiom, (! [A,B,C,A1,B1,C1] : ((cong3(A,B,C,A1,B1,C1))
                => (cong(A,B,A1,B1) & cong(A,C,A1,C1) & cong(B,C,B1,C1))))).
fof(ax_4_4_2, axiom, (! [A,B,C,A1,B1,C1] : ((cong(A,B,A1,B1) & cong(A,C,A1,C1)
                & cong(B,C,B1,C1)) => cong3(A,B,C,A1,B1,C1)))).
fof(ax_4_4_3, axiom, (! [A1,A2,A3,A4,B1,B2,B3,B4]:((cong4(A1,A2,A3,A4,B1,B2,B3,B4))
                => (cong(A1,A2,B1,B2) & cong(A1,A3,B1,B3) & cong(A1,A4,B1,B4)
                & cong(A2,A3,B2,B3) & cong(A2,A4,B2,B4) & cong(A3,A4,B3,B4))))).
fof(ax_4_4_4, axiom, (! [A1,A2,A3,A4,B1,B2,B3,B4] : ((cong(A1,A2,B1,B2)
                & cong(A1,A3,B1,B3) & cong(A1,A4,B1,B4) & cong(A2,A3,B2,B3)
                & cong(A2,A4,B2,B4) & cong(A3,A4,B3,B4))
                => (cong4(A1,A2,A3,A4,B1,B2,B3,B4))))).
fof(ax_branch_cong3, axiom, (! [A,B,C,A1,B1,C1] : ((cong3(A,B,C,A1,B1,C1)
                | ncong3(A,B,C,A1,B1,C1))))).
fof(ax_false_cong3, axiom, (! [A,B,C,A1,B1,C1] : ((cong3(A,B,C,A1,B1,C1)
```

```
                & ncong3(A,B,C,A1,B1,C1)) => $false))).
fof(ax_branch_cong4, axiom, (! [A,B,C,D,P,Q,R,W] : ((cong4(A,B,C,D,P,Q,R,W)
                | ncong4(A,B,C,D,P,Q,R,W))))).
fof(ax_false_cong4, axiom, (! [A,B,C,D,P,Q,R,W] : ((cong4(A,B,C,D,P,Q,R,W)
                & ncong4(A,B,C,D,P,Q,R,W)) => $false))).
fof(ax_4_10_1, axiom, (! [A,B,C] : ((col(A,B,C))
                => (bet(A,B,C) | bet(B,C,A) | bet(C,A,B))))).
fof(ax_4_10_2, axiom, (! [A,B,C] : ((bet(A,B,C)) => col(A,B,C)))).
fof(ax_4_10_3, axiom, (! [A,B,C] : ((bet(B,C,A)) => col(A,B,C)))).
fof(ax_4_10_4, axiom, (! [A,B,C] : ((bet(C,A,B)) => col(A,B,C)))).
fof(ax_4_15_1, axiom, (![A,B,C,D,A1,B1,C1,D1] :((fs(A,B,C,D,A1,B1,C1,D1))
                => (col(A,B,C) & cong3(A,B,C,A1,B1,C1) & cong(A,D,A1,D1)
                & cong(B,D,B1,D1))))).
fof(ax_4_15_2, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((col(A,B,C)
                & cong3(A,B,C,A1,B1,C1) & cong(A,D,A1,D1) & cong(B,D,B1,D1))
                => (fs(A,B,C,D,A1,B1,C1,D1))))).
fof(ax_branch_fs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((fs(A,B,C,D,A1,B1,C1,D1)
                | nfs(A,B,C,D,A1,B1,C1,D1))))).
fof(ax_false_fs, axiom, (! [A,B,C,D,A1,B1,C1,D1] : ((fs(A,B,C,D,A1,B1,C1,D1)
                & nfs(A,B,C,D,A1,B1,C1,D1)) => $false))).
```

Поглавље 5

```
fof(ax_5_4_1, axiom, (! [A,B,C,D] : (le(A,B,C,D) => (? [Y] : (bet(C,Y,D)
                & cong(A,B,C,Y)))))).
fof(ax_5_4_2, axiom, (! [A,B,C,D,Y] : ((bet(C,Y,D) & cong(A,B,C,Y))
                => (le(A,B,C,D))))).
fof(ax_5_4_3, axiom, (! [A,B,C,D] : (ge(C,D,A,B) => le(A,B,C,D)))).
fof(ax_5_4_4, axiom, (! [A,B,C,D] : (le(A,B,C,D) => ge(C,D,A,B)))).
fof(ax_branch_le, axiom, (! [A,B,C,D] : ((le(A,B,C,D) | nle(A,B,C,D))))).
fof(ax_false_le, axiom, (! [A,B,C,D] : ((le(A,B,C,D) & nle(A,B,C,D)) => $false))).
fof(ax_branch_ge, axiom, (! [A,B,C,D] : ((ge(A,B,C,D) | nge(A,B,C,D))))).
fof(ax_false_ge, axiom, (! [A,B,C,D] : ((ge(A,B,C,D) & nge(A,B,C,D)) => $false))).
fof(ax_branch_cong, axiom, (! [A,B,C,D] : (cong(A,B,C,D) | ncong(A,B,C,D)))).
fof(ax_false_cong, axiom, (! [A,B,C,D] : ((cong(A,B,C,D) & ncong(A,B,C,D)) => $false))).
fof(ax_5_14_1, axiom, ! [A,B,C,D] : ((lt(A,B,C,D)) => (le(A,B,C,D) & ncong(A,B,C,D)))).
fof(ax_5_14_2, axiom, ! [A,B,C,D] : ((le(A,B,C,D) & ncong(A,B,C,D)) => lt(A,B,C,D))).
fof(ax_5_14_3, axiom, ! [A,B,C,D] : (gt(C,D,A,B) => lt(A,B,C,D))).
fof(ax_5_14_4, axiom, (! [A,B,C,D] : (lt(A,B,C,D) => gt(C,D,A,B)))).
fof(ax_branch_lt, axiom, (! [A,B,C,D] : ((lt(A,B,C,D) | nlt(A,B,C,D))))).
fof(ax_false_lt, axiom, (! [A,B,C,D] : ((lt(A,B,C,D) & nlt(A,B,C,D)) => $false))).
fof(ax_branch_gt, axiom, (! [A,B,C,D] : ((gt(A,B,C,D) | ngt(A,B,C,D))))).
fof(ax_false_gt, axiom, (! [A,B,C,D] : ((gt(A,B,C,D) & ngt(A,B,C,D)) => $false))).
```

Поглавље 6

```
fof(ax_6_1_1, axiom, (! [A,B,P] : ((out(P,A,B)) => ((A!=P & B!=P & bet(P,A,B))
                  | (A!=P & B!=P & bet(P,B,A)))))).
fof(ax_6_1_2, axiom, (! [A,B,P] : ((A!=P & B!=P & bet(P,A,B)) => out(P,A,B)))).
fof(ax_6_1_3, axiom, (! [A,B,P] : ((A!=P & B!=P & bet(P,B,A)) => out(P,A,B)))).
fof(ax_branch_out, axiom, (![A,B,P] : ((out(P,A,B) | nout(P,A,B))))).
fof(ax_false_out, axiom, (![A,B,P] : ((out(P,A,B) & nout(P,A,B)) => $false))).
fof(ax_6_8_1, axiom, (! [P,A,X] : ((A!=P & point_on_ray(X,P,A)) => out(P,X,A)))).
fof(ax_6_8_2, axiom, (! [P,A,X] : ((A!=P & out(P,X,A)) => point_on_ray(X,P,A)))).
fof(ax_branch_point_on_ray, axiom, (! [A,B,C] : ((point_on_ray(C,A,B)
                  | npoint_on_ray(C,A,B))))).
fof(ax_false_point_on_ray, axiom, (! [A,B,C] : ((point_on_ray(C,A,B)
                  & npoint_on_ray(C,A,B)) => $false))).
fof(ax_6_14_1, axiom, (! [P,Q,A] : ((point_on_line(A,P,Q)) => (P!=Q & col(P,Q,A))))).
fof(ax_6_14_2, axiom, (! [P,Q,A] : ((P!=Q & col(P,Q,A)) => point_on_line(A,P,Q)))).
fof(ax_same_lines_1, axiom, (! [A,B,C,D] : ((A!=B & C!=D & point_on_line(C,A,B)
                   & point_on_line(D,A,B)) => same_lines(A,B,C,D)))).
fof(ax_same_lines_2, axiom, (! [A,B,C,D] : ((same_lines(A,B,C,D)) => (A!=B
                  & C!=D & point_on_line(C,A,B) & point_on_line(D,A,B))))).
fof(ax_false_same_lines, axiom, (! [A,B,C,D] : ((same_lines(A,B,C,D)
                  & nsame_lines(A,B,C,D)) => $false))).
fof(ax_branch_same_lines, axiom, (! [A,B,C,D] : (same_lines(A,B,C,D)
                  | nsame_lines(A,B,C,D)))).
fof(ax_branch_col, axiom, (! [A,B,C] : (col(A,B,C) | ncol(A,B,C)))).
fof(ax_false_col, axiom, (! [A,B,C] : ((col(A,B,C) & ncol(A,B,C)) => $false))).
fof(ax_branch_point_on_line, axiom, (! [A,B,C] : (A!=B
                  => (point_on_line(C,A,B) | npoint_on_line(C,A,B))))).
fof(ax_false_point_on_line, axiom, (! [A,B,C] : ((point_on_line(C,A,B)
                  & npoint_on_line(C,A,B)) => $false))).
fof(ax_6_22_1, axiom, (! [A,B,C,D,X] : ((inter(X,A,B,C,D)) => (A!=B & C!=D
                  & point_on_line(X,A,B) & point_on_line(X,C,D)
                  & nsame_lines(A,B,C,D))))).
fof(ax_6_22_2, axiom, (! [A,B,C,D,X] : ((A!=B & C!=D & point_on_line(X,A,B)
                  & point_on_line(X,C,D) & nsame_lines(A,B,C,D))
                  => inter(X,A,B,C,D)))).
fof(ax_branch_inter, axiom, (! [X,A,B,C,D] : (inter(X,A,B,C,D) | ninter(X,A,B,C,D)))).
fof(ax_false_inter, axiom, (! [X,A,B,C,D] : ((inter(X,A,B,C,D) & ninter(X,A,B,C,D))
                  =>$false))).
```

Поглавље 7

```
fof(ax_7_1, axiom, (! [A,M,B] : ((is_midpoint(M,A,B))
                  => (bet(A,M,B) & cong(M,A,M,B))))).
fof(ax_7_2, axiom, (! [A,M,B] : ((bet(A,M,B) & cong(M,A,M,B))
                  => is_midpoint(M,A,B)))).
fof(ax_branch_is_midpoint, axiom, (! [A,B,C] :
                  (is_midpoint(A,B,C) | nis_midpoint(A,B,C)))).
```

```
fof(ax_false_is_midpoint, axiom, (! [A,B,C] :
                  ((is_midpoint(A,B,C) & nis_midpoint(A,B,C)) => $false))).
fof(ax_7_5_1, axiom, (! [A,A1,C] :
                  (is_symmetric(A,A1,C) => is_midpoint(C,A,A1)))).
fof(ax_7_5_2, axiom, (! [A,A1,C] :
                  (is_midpoint(C,A,A1) => is_symmetric(A,A1,C)))).
fof(ax_branch_is_symmetric, axiom, (! [A,B,C] :
                  (is_symmetric(A,B,C) | nis_symmetric(A,B,C)))).
fof(ax_false_is_symmetric, axiom, (! [A,B,C] :
                  ((is_symmetric(A,B,C) & nis_symmetric(A,B,C)) => $false))).
```

Поглавље 8

```
fof(ax_8_1_1, axiom, (! [A,B,C] : ((per(A,B,C)) =>
                  (? [C1] : (cong(A,C,A,C1) & is_midpoint(B,C,C1)))))).
fof(ax_8_1_2, axiom, (! [A,B,C,C1] : ((cong(A,C,A,C1) & is_midpoint(B,C,C1))
                  => per(A,B,C)))).
fof(ax_branch_per, axiom, (![A,B,C] : (per(A,B,C) | nper(A,B,C)))).
fof(ax_false_per, axiom,(![A,B,C] : ((per(A,B,C) & nper(A,B,C)) => $false))).
fof(ax_8_11_1_1, axiom, (! [X,A,B,C,D] : ((perp_in(X,A,B,C,D) & A!=X & C!=X)
                  => (A!=B & C!=D & point_on_line(X,A,B)
                  & point_on_line(X,C,D) & per(A,X,C))))).
fof(ax_8_11_1_2, axiom, (! [X,A,B,C,D] : ((perp_in(X,A,B,C,D) & A!=X & D!=X)
                  => (A!=B & C!=D & point_on_line(X,A,B)
                  & point_on_line(X,C,D) & per(A,X,D))))).
fof(ax_8_11_1_3, axiom, (! [X,A,B,C,D] : ((perp_in(X,A,B,C,D) & B!=X & C!=X)
                  => (A!=B & C!=D & point_on_line(X,A,B)
                  & point_on_line(X,C,D) & per(B,X,C))))).
fof(ax_8_11_1_4, axiom, (! [X,A,B,C,D] : ((perp_in(X,A,B,C,D) & B!=X & D!=X)
                  => (A!=B & C!=D & point_on_line(X,A,B)
                  & point_on_line(X,C,D) & per(B,X,D))))).
fof(ax_8_11_1_5, axiom, (! [X,A,B,C,D] : ((A!=X & C!=X & A!=B & C!=D
                  & point_on_line(X,A,B) & point_on_line(X,C,D) & per(A,X,C))
                  => perp_in(X,A,B,C,D)))).
fof(ax_8_11_1_6, axiom, (! [X,A,B,C,D] : ((A!=X & D!=X & A!=B & C!=D
                  & point_on_line(X,A,B) & point_on_line(X,C,D) & per(A,X,D))
                  =>  perp_in(X,A,B,C,D)))).
fof(ax_8_11_1_7, axiom, (! [X,A,B,C,D] : ((B!=X & C!=X & A!=B & C!=D
                  & point_on_line(X,A,B) & point_on_line(X,C,D) & per(B,X,C))
                  => perp_in(X,A,B,C,D)))).
fof(ax_8_11_1_8, axiom, (! [X,A,B,C,D] : ((B!=X & D!=X & A!=B & C!=D
                  & point_on_line(X,A,B) & point_on_line(X,C,D) & per(B,X,D))
                  =>  perp_in(X,A,B,C,D)))).
fof(ax_8_11_2_1, axiom, (! [A,B,C,D] : (perp(A,B,C,D) =>
                  (? [X] : perp_in(X,A,B,C,D))))).
fof(ax_8_11_2_2, axiom, (! [X,A,B,C,D] : (perp_in(X,A,B,C,D) => perp(A,B,C,D)))).
```

```
fof(ax_branch_perp_in, axiom, (! [X,A,B,C,D] :
                  (perp_in(X,A,B,C,D) | nperp_in(X,A,B,C,D)))).
fof(ax_false_perp_in, axiom, (! [X,A,B,C,D] :
                  ((perp_in(X,A,B,C,D) & nperp_in(X,A,B,C,D)) => $false))).
fof(ax_branch_perp, axiom, (![A,B,C,D] : (perp(A,B,C,D) | nperp(A,B,C,D)))).
fof(ax_false_perp, axiom, (! [A,B,C,D] :
                  ((perp(A,B,C,D) & nperp(A,B,C,D)) => $false))).
```

Поглавље 9

```
fof(ax_9_1_1, axiom, (! [P,Q,A,B] : ((two_sides(A,B,P,Q)) =>
                  (? [T] : (P!=Q & npoint_on_line(A,P,Q) & npoint_on_line(B,P,Q)
                  & point_on_line(T,P,Q) & bet(A,T,B)))))).
fof(ax_9_1_2, axiom, (! [P,Q,A,B,T] : ((P!=Q & npoint_on_line(A,P,Q)
                  & npoint_on_line(B,P,Q) & point_on_line(T,P,Q) & bet(A,T,B))
                  => two_sides(A,B,P,Q)))).
fof(ax_9_7_1, axiom, (! [P,Q,A,B] : (one_side(A,B,P,Q) =>
                  (? [C] : (P!=Q & two_sides(A,C,P,Q) & two_sides(B,C,P,Q)))))).
fof(ax_9_7_2, axiom, (! [P,Q,A,B,C] : ((P!=Q & two_sides(A,C,P,Q)
                  & two_sides(B,C,P,Q)) => one_side(A,B,P,Q)))).
fof(ax_branch_two_sides, axiom, (! [P,Q,A,B] :
                  (P!=Q => (two_sides(A,B,P,Q) | ntwo_sides(A,B,P,Q))))).
fof(ax_false_two_sides, axiom, (! [P,Q,A,B] :
                  ((two_sides(A,B,P,Q) & ntwo_sides(A,B,P,Q)) => $false))).
fof(ax_branch_one_side, axiom, (! [P,Q,A,B] :
                  (P!=Q => (one_side(A,B,P,Q) | none_side(A,B,P,Q))))).
fof(ax_false_one_side, axiom, (! [P,Q,A,B] :
                  ((one_side(A,B,P,Q) & none_side(A,B,P,Q)) => $false))).
```

Поглавље 10

```
fof(ax_10_3_1, axiom, (![P,Q,A,B] : ((A!=B & is_image_spec(Q,P,A,B))
                  => ((?[X]:(is_midpoint(X,P,Q) & point_on_line(X,A,B) & perp(A,B,P,Q)))
                  | (?[X]:(is_midpoint(X,P,Q) & point_on_line(X,A,B) & P=Q)))))).
fof(ax_10_3_2_1, axiom, (! [P,P1,A,B,X] : ((A!=B & is_midpoint(X,P,P1)
                  & point_on_line(X,A,B) & perp(A,B,P,P1))
                  => is_image_spec(P1,P,A,B)))).
%fof(ax_10_3_2_2, axiom, (! [P,P1,A,B,X] : ((A!=B & is_midpoint(X,P,P1)
                  & point_on_line(X,A,B) & P=P1) => is_image_spec(P1,P,A,B)))).
fof(ax_10_3_2_2, axiom, (! [P,A,B,X] : ((A!=B&is_midpoint(X,P,P)
                  & point_on_line(X,A,B)) => is_image_spec(P,P,A,B)))).
fof(ax_branch_is_image_spec, axiom, (! [A,B,C,D] : ((C!=D)
                  => (is_image_spec(A,B,C,D) | nis_image_spec(A,B,C,D))))).
fof(ax_false_is_image_spec, axiom, (! [A,B,C,D] :
                  ((is_image_spec(A,B,C,D) & nis_image_spec(A,B,C,D)) => $false))).
fof(ax_10_3_3, axiom, (! [P,P1,A,B] : ((is_image(P1,P,A,B)) =>
```

```
                ((A!=B & is_image_spec(P1,P,A,B))
                | (A = B & is_midpoint(A,P,P1)))))).
fof(ax_10_3_4_1, axiom, (! [P,P1,A,B] : ((A!=B & is_image_spec(P1,P,A,B))
                => is_image(P1,P,A,B)))).
fof(ax_10_3_4_2, axiom, (! [P,P1,A] :
                (is_midpoint(A,P,P1) => is_image(P1,P,A,A)))).
fof(ax_branch_is_image, axiom, (! [P, P1, A, B] :
                (is_image(P1,P,A,B) | nis_image(P1,P,A,B)))).
fof(ax_false_is_image, axiom, (! [P, P1, A, B] :
                ((is_image(P1, P, A, B) & nis_image(P1, P, A, B)) => $false))).
```

Поглавље 11

```
fof(ax_11_2_1, axiom, (! [A,B,C,D,E,F] : (cong_angle(A,B,C,D,E,F)
                => (? [A1,C1,D1,F1] : (A!=B & C!=B & D!=E & F!=E
                & bet(B,A,A1) & cong(A,A1,E,D) & bet(B,C,C1) & cong(C,C1,E,F)
                & bet(E,D,D1) & cong(D,D1,B,A) & bet(E,F,F1) & cong(F,F1,B,C)
                & cong(A1,C1,D1,F1)))))).
fof(ax_11_2_2, axiom, (! [A,B,C,D,E,F,A1,C1,D1,F1] : ((A!=B & C!=B & D!=E
                & F!=E & bet(B,A,A1) & cong(A,A1,E,D) & bet(B,C,C1)
                & cong(C,C1,E,F) & bet(E,D,D1) & cong(D,D1,B,A) & bet(E,F,F1)
                & cong(F,F1,B,C) & cong(A1,C1,D1,F1))
                => (cong_angle(A,B,C,D,E,F))))).
fof(ax_distinct_1, axiom, (! [A,B,C] :
                ((distinct(A,B,C)) => (A!=B & A!=C & B!=C)))).
fof(ax_distinct_2, axiom, (! [A,B,C] :
                ((A!=B & A!=C & B!=C) => (distinct(A,B,C))))).
fof(ax_branch_distinct, axiom, (! [A,B,C] :
                ((distinct(A,B,C) | ndistinct(A,B,C))))).
fof(ax_false_distinct, axiom, (! [A,B,C] :
                ((distinct(A,B,C) & ndistinct(A,B,C)) => $false))).
fof(ax_11_23_1_1, axiom, (! [A,B,C,P] : ((point_in_angle(P,A,B,C))
                => ((? [X] : (A!=B & C!=B & P!=B & bet(A,X,C) & X=B))
                | (? [X] : (A!=B & C!=B & P!=B & bet(A,X,C) & out(B,X,P)))))).
%fof(ax_11_23_1_2, axiom, (! [A,B,C,P,X] : ((A!=B & C!=B & P!=B & bet(A,X,C)
                & X = B) => (point_in_angle(P,A,B,C))))).
fof(ax_11_23_1_2, axiom, (! [A,B,C,P] : ((A!=B & C!=B & P!=B & bet(A,B,C))
                => (point_in_angle(P,A,B,C))))).
fof(ax_11_23_2, axiom, (! [A,B,C,P,X] : ((A!=B & C!=B & P!=B & bet(A,X,C)
                & out(B,X,P)) => point_in_angle(P,A,B,C)))).
fof(ax_branch_point_in_angle, axiom, (! [P,A,B,C] :
                ((point_in_angle(P,A,B,C) | npoint_in_angle(P,A,B,C))))).
fof(ax_false_point_in_angle, axiom, (! [P,A,B,C] :
                ((point_in_angle(P,A,B,C) & npoint_in_angle(P,A,B,C)) => $false))).
fof(ax_11_27_1, axiom, (! [A,B,C,D,E,F] : ((le_angle(A,B,C,D,E,F)) => (? [P]
                : (point_in_angle(P,D,E,F) & cong_angle(A,B,C,D,E,P)))))).
```

```
fof(ax_11_27_2, axiom, (! [A,B,C,D,E,F,P] : ((point_in_angle(P,D,E,F)
                & cong_angle(A,B,C,D,E,P)) => (le_angle(A,B,C,D,E,F))))).
fof(ax_branch_le_angle, axiom, (! [A,B,C,D,E,F] :
                ((le_angle(A,B,C,D,E,F) | nle_angle(A,B,C,D,E,F))))).
fof(ax_false_le_angle, axiom, (! [A,B,C,D,E,F] : ((le_angle(A,B,C,D,E,F)
                & nle_angle(A,B,C,D,E,F)) => $false))).
fof(ax_11_27_3, axiom, (! [A,B,C,D,E,F] :
                (ge_angle(A,B,C,D,E,F) => le_angle(D,E,F,A,B,C)))).
fof(ax_11_27_4, axiom, (! [A,B,C,D,E,F] :
                (le_angle(D,E,F,A,B,C) => ge_angle(A,B,C,D,E,F)))).
fof(ax_branch_ge_angle, axiom, (! [A,B,C,D,E,F] :
                ((ge_angle(A,B,C,D,E,F) | nle_angle(A,B,C,D,E,F))))).
fof(ax_false_ge_angle, axiom, (! [A,B,C,D,E,F] :
                ((ge_angle(A,B,C,D,E,F) & nge_angle(A,B,C,D,E,F)) => $false))).
fof(ax_branch_cong_angle, axiom, (! [A,B,C,D,E,F] :
                (cong_angle(A,B,C,D,E,F) | ncong_angle(A,B,C,D,E,F)))).
fof(ax_cong_angle_false, axiom, (! [A,B,C,D,E,F] : ((cong_angle(A,B,C,D,E,F)
                & ncong_angle(A,B,C,D,E,F)) => $false))).
fof(ax_11_38_1, axiom, (! [A,B,C,D,E,F] : ((lt_angle(A,B,C,D,E,F))
                => (le_angle(A,B,C,D,E,F) & ncong_angle(A,B,C,D,E,F))))).
fof(ax_11_38_2, axiom, (! [A,B,C,D,E,F] : ((le_angle(A,B,C,D,E,F)
                & ncong_angle(A,B,C,D,E,F)) => (lt_angle(A,B,C,D,E,F))))).
fof(ax_11_38_3, axiom, (! [A,B,C,D,E,F] :
                (gt_angle(A,B,C,D,E,F) => lt_angle(D,E,F,A,B,C)))).
fof(ax_11_38_4, axiom, (! [A,B,C,D,E,F] :
                (lt_angle(A,B,C,D,E,F) => gt_angle(D,E,F,A,B,C)))).
fof(ax_branch_gt_angle, axiom, (! [A,B,C,D,E,F] :
                ((gt_angle(A,B,C,D,E,F) | ngt_angle(A,B,C,D,E,F))))).
fof(ax_false_gt_angle, axiom, (! [A,B,C,D,E,F] :
                ((gt_angle(A,B,C,D,E,F) & ngt_angle(A,B,C,D,E,F)) => $false))).
fof(ax_branch_lt_angle, axiom, (! [A,B,C,D,E,F] :
                ((lt_angle(A,B,C,D,E,F) | nlt_angle(A,B,C,D,E,F))))).
fof(ax_false_lt_angle, axiom, (! [A,B,C,D,E,F] :
                ((lt_angle(A,B,C,D,E,F) & nlt_angle(A,B,C,D,E,F)) => $false))).
fof(ax_11_39_1, axiom, (! [A,B,C] : ((acute(A,B,C)) => (? [A1,B1,C1]
                : (per(A1,B1,C1) & lt_angle(A,B,C,A1,B1,C1)))))).
fof(ax_11_39_2, axiom, (! [A,B,C,A1,B1,C1] : ((per(A1,B1,C1)
                & lt_angle(A,B,C,A1,B1,C1)) => (acute(A,B,C))))).
fof(ax_11_39_3, axiom, (! [A,B,C] : ((obtuse(A,B,C)) => (? [A1,B1,C1]
                : (per(A1,B1,C1) & gt_angle(A,B,C,A1,B1,C1)))))).
fof(ax_11_39_4, axiom, (! [A,B,C,A1,B1,C1] : ((per(A1,B1,C1)
                & gt_angle(A,B,C,A1,B1,C1)) => (obtuse(A,B,C))))).
fof(ax_branch_acute, axiom, (! [A,B,C] : (acute(A,B,C) | nacute(A,B,C)))).
fof(ax_false_acute, axiom, (! [A,B,C] :
                ((acute(A,B,C) & nacute(A,B,C)) => $false))).
```

```
fof(ax_branch_obtuse, axiom, (! [A,B,C] : (obtuse(A,B,C) | nobtuse(A,B,C)))).
fof(ax_false_obtuse, axiom, (! [A,B,C] :
                  ((obtuse(A,B,C) & nobtuse(A,B,C)) => $false))).
fof(ax_int1, axiom, (! [A,B,C,D] :
                  (int(A,B,C,D) => (? [X] : inter(X,A,B,C,D))))).
fof(ax_int2, axiom, (! [A,B,C,D,X] : (inter(X,A,B,C,D) => int(A,B,C,D)))).
fof(ax_branch_int, axiom, (! [A,B,C,D] : ((int(A,B,C,D) | nint(A,B,C,D))))).
fof(ax_false_int, axiom, (! [A,B,C,D] :
                  ((int(A,B,C,D) & nint(A,B,C,D)) => $false))).
```

Поглавље 12

```
fof(ax_12_2_1, axiom, (! [A,B,C,D] :
                  ((A!=B & C!=D & nint(A,B,C,D))=> parallel(A,B,C,D)))).
fof(ax_12_2_2, axiom, (! [A,B,C,D] :
                  ((parallel(A,B,C,D)) => (A!=B & C!=D & nint(A,B,C,D))))).
fof(ax_12_3_1, axiom, (! [A,B,C,D] : ((parallel_broad(A,B,C,D)) =>
                  ((A!=B & C!=D & parallel(A,B,C,D)) | (A!=B & C!=D & same_lines(A,B,C,D)))))).
fof(ax_12_3_2, axiom, (! [A,B,C,D] : ((A!=B & C!=D & parallel(A,B,C,D))
                  => (parallel_broad(A,B,C,D))))).
fof(ax_12_3_3, axiom, (! [A,B,C,D] : ((A!=B & C!=D & same_lines(A,B,C,D))
                  => (parallel_broad(A,B,C,D))))).
fof(ax_branch_parallel, axiom, (! [A,B,C,D] :
                  ((parallel(A,B,C,D) | nparallel(A,B,C,D))))).
fof(ax_false_parallel, axiom, (! [A,B,C,D] :
                  ((parallel(A,B,C,D) & nparallel(A,B,C,D)) => $false))).
fof(ax_branch_parallel_broad, axiom, (! [A,B,C,D] :
                  ((parallel_broad(A,B,C,D) | nparallel_broad(A,B,C,D))))).
fof(ax_false_parallel_broad, axiom, (! [A,B,C,D] :
                  ((parallel_broad(A,B,C,D) & nparallel_broad(A,B,C,D)) => $false))).
```

Додатак B

Хилбертов аксиоматски систем

Предикатски симболи коришћени у запису Хилбертовог аксиоматског система Позитиван облик предикатских симбола је дат у наставку текста. Негативан облик предикатских симбола се добија када се испред имена предикатског симбола допише слово n.

$inc\_po\_l(A, L)$	Тачка A припада правој L.
$inc\_po\_pl(A, P)$	Тачка A припада равни P.
$inc\_l\_pl(L, P)$	Права L припада равни P.
$int\_l\_l(L1, L2)$	Праве $L1$ и $L2$ се секу.
$int\_l\_pl(L, P)$	Права L и раван P се секу.
$int\_pl\_pl(P1, P2)$	Равни $P1$ и $P2$ се секу.
$cop(A, B, C, D)$	Тачке A, B, C и D су компланарне.
$col(A, B, C)$	Тачке A, B и C су колинеарне.
$bet(A, B, C)$	Тачка B се налази између тачака A и C.
$cong(A, B, C, D)$	Пар тачака AB је конгруентан пару тачака CD.
$cut(L, A, B)$	Права L сече сегмент AB (тј. постоји тачка X која припада правој L и лежи између тачака A и B).
$pash(A, B, C, L, P)$	Тачке A, B и C су три неколинеарне тачке равни P и L је права равни P која не садржи тачку A и сече сегмент BC.

Прва група Хилбертових аксиома — аксиоме припадања

```
fof(ax_I1, axiom, (! [A,B] : ((point(A) & point(B) & A!=B) =>
                  (?[L] : (line(L) & inc_po_l(A,L) & inc_po_l(B,L)))))).
fof(ax_I2, axiom, (! [A,B,L1,L2] : ((point(A) & point(B) & line(L1) & line(L2)
                  & A!=B & inc_po_l(A,L1) & inc_po_l(B,L1) & inc_po_l(A,L2)
                  & inc_po_l(B,L2)) => (L1=L2)))).
fof(ax_I3a, axiom, (! [L] : ((line(L)) => (? [A,B] : (point(A) & point(B)
                  & A!=B & inc_po_l(A,L) & inc_po_l(B,L)))))).
fof(ax_I3b, axiom, (? [A,B,C] : (point(A) & point(B) & point(C) & ncol(A,B,C)))).
```

```
fof(ax_I4a, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C) & ncol(A,B,C))
                  => (? [P] : (plane(P) & inc_po_pl(A,P) & inc_po_pl(B,P)
                  & inc_po_pl(C,P)))))).
fof(ax_I4b, axiom, (! [P] : ((plane(P)) => (? [A] : (point(A) & inc_po_pl(A,P)))))).
fof(ax_I5, axiom, (! [A,B,C,P1,P2] : ((point(A) & point(B) & point(C)
                  & plane(P1) & plane(P2) & ncol(A,B,C) & inc_po_pl(A,P1)
                  & inc_po_pl(B,P1) & inc_po_pl(C,P1) & inc_po_pl(A,P2)
                  & inc_po_pl(B,P2) & inc_po_pl(C,P2)) => (P1=P2)))).
fof(ax_I6, axiom, (! [A,B,L,P] : ((point(A) & point(B) & line(L) & plane(P)
                  & A!=B & inc_po_l(A,L) & inc_po_l(B,L) & inc_po_pl(A,P)
                  & inc_po_pl(B,P)) => (inc_l_pl(L,P))))).
fof(ax_I7, axiom, (! [P1,P2,A] : ((plane(P1) & plane(P2) & point(A) & P1!=P2
                  & inc_po_pl(A,P1) & inc_po_pl(A,P2)) => (? [B] : (point(B)
                  & A!=B & inc_po_pl(B,P1) & inc_po_pl(B,P2)))))).
fof(ax_I8, axiom, (? [A,B,C,D] : (point(A) & point(B) & point(C) & point(D)
                  & ncop(A,B,C,D)))).
```

Друга група Хилбертових аксиома — аксиоме распореда

```
fof(ax_II1, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C) & bet(A,B,C))
                  => (A!=B & A!=C & B!=C & col(A,B,C) & bet(C,B,A))))).
fof(ax_II2, axiom, (! [A,B] : ((point(A) & point(B) & A!=B)
                  => (? [C] : (point(C) & bet(A,B,C)))))).
fof(ax_II3, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C) & A!=B & A!=C
                   & B!=C & bet(A,B,C)) => (nbet(B,C,A) & nbet(C,A,B))))).
%Pashova aksioma
fof(ax_II4, axiom, (! [A,B,C,L,P] : ((point(A) & point(B) & point(C) & line(L)
                   & plane(P) & pash(A,B,C,L,P)) => (cut(L,C,A) | cut(L,A,B))))).

%dodatni predikati za zapis Pashove aksiome
fof(ax_cut_1, axiom, (! [L,A,B,X] : ((point(A) & point(B) & line(L) & point(X)
                  & ninc_po_l(A,L) & ninc_po_l(B,L) & inc_po_l(X,L)
                  & bet(A,X,B)) => (cut(L,A,B))))).
fof(ax_cut_2, axiom, (! [L,A,B] : ((point(A) & point(B) & line(L)
                  & cut(L,A,B)) => (? [X] : (point(X) & ninc_po_l(A,L)
                  & ninc_po_l(B,L) & inc_po_l(X,L) & bet(A,X,B)))))).
fof(ax_pash_1, axiom, (! [A,B,C,L,P] : ((point(A) & point(B) & point(C)
                  & line(L) & plane(P) & ncol(A,B,C) & inc_po_pl(A,P)
                  & inc_po_pl(B,P) & inc_po_pl(C,P) & inc_l_pl(L,P)
                  & ninc_po_l(A,L) & ninc_po_l(B,L) & ninc_po_l(C,L)
                  & cut(L,B,C)) => pash(A,B,C,L,P))))).
fof(ax_pash_2, axiom, (! [A,B,C,L,P] : ((point(A) & point(B) & point(C)
                  & line(L) & plane(P) & pash(A,B,C,L,P)) => (ncol(A,B,C)
                  & inc_po_pl(A,P) & inc_po_pl(B,P) & inc_po_pl(C,P)
                  & inc_l_pl(L,P) & ninc_po_l(A,L) & ninc_po_l(B,L)
                  & ninc_po_l(C,L) & cut(L,B,C))))).
```

Додатак C

Авигадов аксиоматски систем

Аксиоматски систем креиран за описивање извођења у Еуклидовим „Елементима" припада кохерентној логици тако да се у неизмењеном облику користи у оквиру система описаног у овој тези. У наставку текста су наведене све аксиоме, подељене по групама које је Авигад идентификовао, записане у TPTP формату.

Правила конструисања

```
% POINTS %
fof(ax_points1, axiom, (? [A] : (point(A)))).
fof(ax_points2, axiom, (! [L] : ((line(L)) => (? [A] : (point(A) & on(A,L)))))).
fof(ax_points3, axiom, (! [L,A,B] : ((line(L) & point(A) & point(B) & on(A,L)
                       & on(B,L) & A != B) => (? [C] : (point(C) & on(C,L)
                       & bet(A,C,B)))))).
fof(ax_points4, axiom, (! [L,A,B] : ((line(L) & point(A) & point(B) & on(A,L)
                       & on(B,L) & A != B) => (? [C] : (point(C) & on(C,L)
                       & bet(A,B,C)))))).
fof(ax_points5, axiom, (! [L,A] : ((line(L) & point(A) & non(A,L))
                       => (? [B] : (point(B) & sameside(A,B,L)))))).
fof(ax_points6, axiom, (! [L,A] : ((line(L) & point(A) & non(A,L))
                       => (? [B] : (point(B) & non(B,L) & nsameside(A,B,L)))))).
fof(ax_points7, axiom, (! [C] : ((circle(C)) =>
                       (? [A] : (point(A) & onc(A,C)))))).
fof(ax_points8, axiom, (! [C] : ((circle(C)) =>
                       (? [A] : (point(A) & inside(A,C)))))).
fof(ax_points9, axiom, (! [C] : ((circle(C)) =>
                       (? [A] : (point(A) & ninside(A,C) & nonc(A,C)))))).

% LINES AND CIRCLES %
fof(ax_lines_and_circles1, axiom, (! [A,B] : ((point(A) & point(B) & A != B)
                        => (? [L] : (line(L) & on(A,L) & on(B,L)))))).
```

```
fof(ax_lines_and_circles2, axiom, (! [A,B] : ((point(A) & point(B) & A != B)
                        => (? [C] : (circle(C) & center(A,C) & onc(B,C))))).

% INTERSECTIONS %
fof(ax_intersections1, axiom, (! [L,M] : ((line(L) & line(M) & intersects(L,M))
                         => (? [A] : (point(A) & on(A,L) & on(A,M))))).
fof(ax_intersections2, axiom, (! [C,M] : ((circle(C) & line(M) & intersectslc(M,C))
                        => (? [A] : (point(A) & on(A,M) & onc(A,C))))).
fof(ax_intersections3, axiom, (! [C,M] : ((circle(C) & line(M) & intersectslc(M,C))
                        => (? [A,B] : (point(A) & point(B) & on(A,M) & onc(A,C)
                        & on(B,M) & onc(B,C) & A != B))))).
fof(ax_intersections4, axiom, (! [C,L,B,D] : ((circle(C) & line(L) & point(B)
                        & point(D) & inside(B,C) & on(B,L) & ninside(D,C)
                        & nonc(D,C) & on(D,L)) => (? [A] : (point(A) & onc(A,C)
                        & on(A,L) & bet(B,A,D))))).
fof(ax_intersections5, axiom, (! [C,L,B,D] : ((circle(C) & line(L) &  point(B)
                        & point(D) & inside(B,C) & on(B,L) & D != B & on(D,L))
                        => (? [A] : (point(A) & onc(A,C) & on(A,L) & bet(A,B,D))))).
fof(ax_intersections6, axiom, (! [C1,C2] : ((circle(C1) & circle(C2)
                        & intersectscc(C1,C2)) =>
                        (? [A] : (point(A) & onc(A,C1) & onc(A,C2))))).
fof(ax_intersections7, axiom, (! [C1,C2] : ((circle(C1) & circle(C2)
                        & intersectscc(C1,C2)) => (? [A,B] : (point(A)
                        & point(B) & onc(A,C1) & onc(A,C2) & onc(B,C1)
                        & onc(B,C2) & A != B))))).
fof(ax_intersections8, axiom, (! [C1,C2,D1,D2,B,L] : ((circle(C1) & circle(C2)
                        & line(L) & point(D1) & point(D2) & point(B)
                        & intersectscc(C1,C2) & center(D1,C1) & center(D2,C2)
                        & on(D1,L) & on(D2,L) & non(B,L)) => (? [A] :
                        (point(A) & onc(A,C1) & onc(A,C2) & sameside(A,B,L))))).
fof(ax_intersections9, axiom, (! [C1,C2,D1,D2,B,L] : ((circle(C1) & circle(C2)
                        & point(D1) & point(D2) & point(B) & line(L)
                        & intersectscc(C1,C2) & center(D1,C1) & center(D2,C2)
                        & on(D1,L) & on(D2,L) & non(B,L)) => (? [A] : (point(A)
                        & onc(A,C1) & onc(A,C2) & nsameside(A,B,L) & non(A,L))))).
```

Правила извођења

```
% GENERALITIES %
fof(ax_generalities1, axiom, (! [A,B,L,M] : ((point(A) & point(B) & line(L)
                        & line(M) & A != B & on(A,L) & on(B,L) & on(A,M)
                        & on(B,M)) => L = M))).
fof(ax_generalities2, axiom, (! [A,B,C] : ((point(A) & point(B) & circle(C)
                        & center(A,C) & center(B,C)) => A = B))).
fof(ax_generalities3, axiom, (! [A,C] : ((point(A) & circle(C) & center(A,C))
```

```
                        => inside(A,C))))).
fof(ax_generalities4, axiom, (! [A,C] : ((point(A) & circle(C) & center(A,C))
                        => nonc(A,C)))).

% BETWEEN AXIOMS %
fof(ax_branch_bet, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C))
                        => (bet(A,B,C) | nbet(A,B,C))))).
fof(ax_false_bet, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
                        & bet(A,B,C) & nbet(A,B,C)) => $false))).
fof(ax_bet1, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C) & bet(A,B,C))
                        => (bet(C,B,A) & A != B & A != C & nbet(B,A,C))))).
fof(ax_bet2, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
                        & bet(A,B,C) & on(A,L) & on(B,L)) => on(C,L)))).
fof(ax_bet3, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
                        & bet(A,B,C) & on(A,L) & on(C,L)) => on(B,L)))).
fof(ax_bet4, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C) & point(D)
                        & bet(A,B,C) & bet(A,D,B)) => bet(A,D,C)))).
fof(ax_bet5, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C) & point(D)
                        & bet(A,B,C) & bet(B,C,D)) => bet(A,B,D)))).
fof(ax_bet6,axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
                        & on(A,L) & on(B,L) & on(C,L) & A != B & A != C &
                        B != C) => (bet(A,B,C) | bet(B,A,C) | bet(A,C,B))))).
fof(ax_bet7, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C) & point(D)
                        & bet(A,B,C) & bet(A,B,D)) => nbet(C,B,D)))).

% SAME-SIDE AXIOMS %
fof(ax_sameside1,axiom, (![A,L] :
                        ((point(A) & line(L) & non(A,L)) => sameside(A,A,L)))).
fof(ax_sameside2,axiom, (![A,B,L] : ((point(A) & point(B) & line(L)
                        & sameside(A,B,L)) => (sameside(B,A,L))))).
fof(ax_sameside3,axiom, (![A,B,L] : ((point(A) & point(B) & line(L)
                        & sameside(A,B,L)) => (non(A,L))))).
fof(ax_sameside4,axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C)
                        & line(L) & sameside(A,B,L) & sameside(A,C,L))
                        => sameside(B,C,L)))).
fof(ax_sameside5,axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C)
                        & line(L) & non(A,L) & non(B,L) & non(C,L) & nsameside(A,B,L))
                        => (sameside(A,C,L) | sameside(B,C,L))))).

% PASCH AXIOMS %
fof(ax_pasch1,axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
                        & bet(A,B,C) & sameside(A,C,L)) => sameside(A,B,L)))).
fof(ax_pasch2,axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
                        & bet(A,B,C) & on(A,L) & non(B,L)) => sameside(B,C,L)))).
fof(ax_pasch3,axiom, (![A,B,C,L] : ((point(A) & point(B) & point(C) & line(L)
```

```
                        & bet(A,B,C) & on(B,L)) => nsameside(A,C,L)))).
fof(ax_pasch4,axiom, (![A,B,C,L,M] : ((point(A) & point(B) & point(C)
                        & line(L) & line(M) & A != B & B != C & L != M
                        & on(A,M) & on(B,M) & on(C,M) & nsameside(A,C,L)
                        & on(B,L)) => bet(A,B,C)))).

% TRIPLE INCIDENCE AXIOMS %
fof(ax_incidence1, axiom, (! [L,M,N,A,B,C,D] : ((line(L) & line(M) & line(N)
                        & point(A) & point(B) & point(C) & point(D) & on(A,L)
                        & on(A,M) & on(A,N) & on(B,L) & on(C,M) & on(D,N)
                        & sameside(C,D,L) & sameside(B,C,N))
                        => (nsameside(B,D,M))))).
fof(ax_incidence2, axiom, (! [L,M,N,A,B,C,D] : ((line(L) & line(M) & line(N)
                        & point(A) & point(B) & point(C) & point(D) & on(A,L)
                        & on(A,M) & on(A,N) & on(B,L) & on(C,M) & on(D,N)
                        & sameside(C,D,L) & nsameside(B,D,M) & non(D,M)
                        & B != A) => (sameside(B,C,N))))).
fof(ax_incidence3, axiom, (! [L,M,N,A,B,C,D,E] : ((line(L) & line(M) & line(N)
                        & point(A) & point(B) & point(C) & point(D) & point(E)
                        & on(A,L) & on(A,M) & on(A,N) & on(B,L) & on(C,M)
                        & on(D,N) & sameside(C,D,L) & sameside(B,C,N)
                        & sameside(D,E,M) & sameside(C,E,N))
                        => (sameside(C,E,L))))).

% CIRCLE AXIOMS %
fof(ax_circle1, axiom, (! [A,B,C,L,P] : ((point(A) & point(B) & point(C)
                        & line(L) & circle(P) & on(A,L) & on(B,L) & on(C,L)
                        & inside(A,P) & onc(B,P) & onc(C,P) & B != C)
                        => (bet(B,A,C))))).
fof(ax_circle2_1, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                        & circle(P) & inside(A,P) & inside(B,P) & bet(A,C,B))
                        => (inside(C,P))))).
fof(ax_circle2_2, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                        & circle(P) & inside(A,P) & onc(B,P) & bet(A,C,B))
                        => (inside(C,P))))).
fof(ax_circle2_3, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                        & circle(P) & onc(A,P) & inside(B,P) & bet(A,C,B))
                        => (inside(C,P))))).
fof(ax_circle2_4, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                        & circle(P) & onc(A,P) & onc(B,P) & bet(A,C,B))
                        => (inside(C,P))))).
fof(ax_circle3_1, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                        & circle(P) & inside(A,P) & ninside(C,P) & bet(A,C,B))
                        => (ninside(B,P) & nonc(B,P))))).
fof(ax_circle3_2, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
```

```
                        & circle(P) & onc(A,P) & ninside(C,P) & bet(A,C,B))
                        => (ninside(B,P) & nonc(B,P))))).
fof(ax_circle4, axiom, (! [P1,P2,C,D,A,B,L] : ((circle(P1) & circle(P2)
                        & point(C) & point(D) & point(A) & point(B) & line(L)
                        & P1 != P2 & intersectscc(P1,P2) & C != D & onc(C,P1)
                        & onc(C,P2) & onc(D,P1) & onc(D,P2) & center(A,P1)
                        & center(B,P2) & on(A,L) & on(B,L))
                        => (nsameside(C,D,L))))).

% INTERSECTION RULES%
fof(ax_rule1, axiom, (! [A,B,L,M] : ((point(A) & point(B) & line(L) & line(M)
                        & non(A,L) & non(B,L) & nsameside(A,B,L) & on(A,M)
                        & on(B,M)) => (intersects(L,M))))).
fof(ax_rule2_1, axiom, (! [A,B,C,L] : ((point(A) & point(B) & circle(C)
                        & line(L) & onc(A,C) & onc(B,C) & non(A,L) & non(B,L)
                        & nsameside(A,B,L)) => (intersectslc(L,C))))).
fof(ax_rule2_2, axiom, (! [A,B,C,L] : ((point(A) & point(B) & circle(C)
                        & line(L) & onc(A,C) & inside(B,C) & non(A,L)
                        & non(B,L) & nsameside(A,B,L)) => (intersectslc(L,C))))).
fof(ax_rule2_3, axiom, (! [A,B,C,L] : ((point(A) & point(B) & circle(C)
                        & line(L) & inside(A,C) & onc(B,C) & non(A,L)
                        & non(B,L) & nsameside(A,B,L)) => (intersectslc(L,C))))).
fof(ax_rule2_4, axiom, (! [A,B,C,L] : ((point(A) & point(B) & circle(C)
                        & line(L) & inside(A,C) & inside(B,C) & non(A,L)
                        & non(B,L) & nsameside(A,B,L)) => (intersectslc(L,C))))).
fof(ax_rule3, axiom, (! [A,C,L] : ((point(A) & circle(C) & line(L)
                        & inside(A,C) & on(A,L)) => (intersectslc(L,C))))).
fof(ax_rule4, axiom, (! [A,B,C1,C2] : ((point(A) & point(B) & circle(C1)
                        & circle(C2) & onc(A,C1) & onc(B,C1) & inside(A,C2)
                        & nonc(B,C2) & ninside(B,C2)) => (intersectscc(C1,C2))))).
fof(ax_rule5, axiom, (! [A,B,C1,C2] : ((point(A) & point(B) & circle(C1)
                        & circle(C2) & onc(A,C1) & inside(B,C1) & inside(A,C2)
                        & onc(B,C2)) => (intersectscc(C1,C2))))).

% ADDITIONAL AXIOMS %
fof(ax_cong_eq1, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
                        & cong(A,A,B,C)) => (B=C)))).
fof(ax_cong_reflexivity, axiom, (! [A,B] : ((point(A) & point(B))
                        => (cong(A,B,A,B))))).
fof(ax_cong_symmetry, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C)
                        & point(D) & cong(A,B,C,D)) => (cong(C,D,A,B))))).
fof(ax_cong_transitivity, axiom, (! [A,B,P,Q,R,S] : ((point(A) & point(B)
                        & point(P) & point(Q) & point(R) & point(S)
                        & cong(A,B,P,Q) & cong(A,B,R,S)) => (cong(P,Q,R,S))))).
fof(ax_cong_zero1, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
```

```
                    & cong(A,B,C,C)) => cong_zero(A,B))).
fof(ax_cong_zero2, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
                    & cong_zero(A,B)) => cong(A,B,C,C)))).
fof(ax_segment_add1, axiom, (! [A1,A2,B1,B2,C1,C2,P,Q] :
                    ((point(A1) & point(A2) & point(B1) & point(B2)
                    & point(C1) & point(C2) & point(P) & point(Q)
                    & segment_add(A1,A2,B1,B2,C1,C2) & cong(A1,A2,P,Q))
                    => (segment_add(P,Q,B1,B2,C1,C2))))).
fof(ax_segment_add2, axiom, (! [A1,A2,B1,B2,C1,C2,P,Q] :
                    ((point(A1) & point(A2) & point(B1) & point(B2)
                    & point(C1) & point(C2) & point(P) & point(Q)
                    & segment_add(A1,A2,B1,B2,C1,C2) & cong(B1,B2,P,Q))
                    => (segment_add(A1,A2,P,Q,C1,C2))))).
fof(ax_segment_add3, axiom, (! [A1,A2,B1,B2,C1,C2,P,Q] :
                    ((point(A1) & point(A2) & point(B1) & point(B2)
                    & point(C1) & point(C2) & point(P) & point(Q)
                    & segment_add(A1,A2,B1,B2,C1,C2) & cong(C1,C2,P,Q))
                    => (segment_add(A1,A2,B1,B2,P,Q))))).
fof(ax_segment_add4, axiom, (! [A1,A2,B1,B2,C1,C2,P,Q] :
                    ((point(A1) & point(A2) & point(B1) & point(B2)
                    & point(C1) & point(C2) & point(P) & point(Q)
                    & segment_add(A1,A2,B1,B2,C1,C2))
                    => (segment_add(B1,B2,A1,A2,C1,C2))))).
fof(ax_segment_add5, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C)
                    & point(D) & cong_zero(C,D))
                    => (segment_add(A,B,C,D,A,B) & segment_add(C,D,A,B,A,B))))).
fof(ax_segment_add6, axiom, (! [A,B,C,D,P,Q,A1,B1,C1,D1,P1,Q1] :
                    ((point(A) & point(B) & point(C) & point(D) & point(P)
                    & point(Q) & segment_add(A,B,C,D,P,Q)
                    & segment_add(A1,B1,C1,D1,P1,Q1) & cong(A,B,A1,B1)
                    & cong(C,D,C1,D1)) => (cong(P,Q,P1,Q1))))).
fof(ax_segment_add7, axiom, (! [A,B,C,D,P,Q,A1,B1,C1,D1,P1,Q1] :
                    ((point(A) & point(B) & point(C) & point(D) & point(P)
                    & point(Q) & segment_add(A,B,C,D,P,Q)
                    & segment_add(A1,B1,C1,D1,P1,Q1) & cong(A,B,A1,B1)
                    & cong(P,Q,P1,Q1)) => (cong(C,D,C1,D1))))).
fof(ax_segment_add8, axiom, (! [A,B,C,D,P,Q,A1,B1,C1,D1,P1,Q1] :
                    ((point(A) & point(B) & point(C) & point(D) & point(P)
                    & point(Q) & segment_add(A,B,C,D,P,Q)
                    & segment_add(A1,B1,C1,D1,P1,Q1) & cong(C,D,C1,D1)
                    & cong(P,Q,P1,Q1)) => (cong(A,B,A1,B1))))).
fof(ax_cong_less1, axiom, (![A1,A2,B1,B2,C1,C2] : ((point(A1) & point(A2)
                    & point(B1) & point(B2) & point(C1) & point(C2)
                    & segment_add(A1,A2,B1,B2,C1,C2))
                    => (cong_less(A1,A2,C1,C2) & cong_less(B1,B2,C1,C2))))).
```

```
fof(ax_cong_less2, axiom, (! [A,B,C,D,E,F,G,H,K,L] : ((point(A) & point(B)
                        & point(C) & point(D) & point(E) & point(F) & point(G)
                        & point(H) & point(K) & point(L) & cong_less(A,B,C,D)
                        & segment_add(A,B,E,F,G,H) & segment_add(C,D,E,F,K,L))
                        => (cong_less(G,H,K,L))))).
fof(ax_cong_less3, axiom, (! [A,B,C,D,A1,B1] : ((point(A) & point(B)
                        & point(C) & point(D) & point(A1) & point(B1)
                        & cong_less(A,B,C,D) & cong(A,B,A1,B1))
                        => (cong_less(A1,B1,C,D))))).
fof(ax_cong_less4, axiom, (! [A,B,C,D,C1,D1] : ((point(A) & point(B)
                        & point(C) & point(D) & point(C1) & point(D1)
                        & cong_less(A,B,C,D) & cong(C,D,C1,D1))
                        => (cong_less(A,B,C1,D1))))).
fof(ax_cong_leq1, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C)
                        & point(D) & cong_leq(A,B,C,D))
                        => (cong(A,B,C,D) | cong_less(A,B,C,D))))).
fof(ax_cong_leq2, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C)
                        & point(D) & cong(A,B,C,D)) => (cong_leq(A,B,C,D))))).
fof(ax_cong_leq3, axiom, (! [A,B,C,D] : ((point(A) & point(B) & point(C)
                        & point(D) & cong_less(A,B,C,D))
                        => (cong_leq(A,B,C,D))))).
fof(ax_cong_angle_symmetry, axiom, (! [A,B,C,A1,B1,C1] : ((point(A) & point(B)
                        & point(C) & point(A1) & point(B1) & point(C1)
                        & cong_angle(A,B,C,A1,B1,C1))
                        => (cong_angle(A1,B1,C1,A,B,C))))).
fof(ax_cong_angle_reflexivity, axiom, (! [A,B,C] : ((point(A) & point(B)
                        & point(C)) => (cong_angle(A,B,C,A,B,C))))).
fof(ax_cong_angle_transitivity, axiom, (! [A,B,C,A1,B1,C1,A2,B2,C2]
                        : ((point(A) & point(B) & point(C) & point(A1)
                        & point(B1) & point(C1) & cong_angle(A,B,C,A1,B1,C1)
                        & cong_angle(A,B,C,A2,B2,C2))
                        => (cong_angle(A1,B1,C1,A2,B2,C2))))).
fof(ax_cong_angle_less, axiom, (! [A,B,C,D,E,F,P,Q,R] : ((point(A) & point(B)
                        & point(C) & point(D) & point(E) & point(F) & point(P)
                        & point(Q) & point(R) & angle_add(A,B,C,D,E,F,P,Q,R))
                        => (cong_angle_less(A,B,C,P,Q,R)
                        & cong_angle_less(D,E,F,P,Q,R))))).

% METRIC INFERENCES %
fof(ax_metric1_1, axiom, (! [A,B] : ((point(A) & point(B) & cong_zero(A,B))
                        => (A=B)))).
fof(ax_metric1_2, axiom, (! [A] : ((point(A)) => (cong_zero(A,A))))).
fof(ax_metric2, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C))
                        => (cong_leq(A,A,B,C))))).
fof(ax_metric3, axiom, (! [A,B] : ((point(A) & point(B)) => (cong(A,B,B,A))))).
```

```
fof(ax_metric4, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
                      & A != B & A != C) => (cong_angle(A,B,C,C,B,A))))).
fof(ax_metric5_1, axiom, (! [A,B,C,D,E,F] : ((point(A) & point(B) & point(C)
                      & point(D) & point(E) & point(F) & cong_angle_zero(A,B,C))
                      => (cong_angle_leq(A,B,C,D,E,F))))).
fof(ax_metric5_2, axiom, (! [A,B,C,R1,R2,R3,P1,P2,P3,Q1,Q2,Q3]
                      : ((point(A) & point(B) & point(C) & point(R1)
                      & point(R2) & point(R3) & point(P1) & point(P2)
                      & point(P3) & point(Q1) & point(Q2) & point(Q3)
                      & right_angle(R1,R2,R3) & right_angle(P1,P2,P3)
                      & angle_add(R1,R2,R3,P1,P2,P3,Q1,Q2,Q3))
                      => (cong_angle_leq(A,B,C,Q1,Q2,Q3))))).
fof(ax_metric6, axiom, (! [A,B] :
                      ((point(A) & point(B)) => (cong_area_zero(A,A,B))))).
fof(ax_metric7, axiom, (! [A,B,C,D,E,F] : ((point(A) & point(B) & point(C)
                      & point(D) & point(E) & point(F) & cong_area_zero(A,B,C))
                      => (cong_area_leq(A,B,C,D,E,F))))).
fof(ax_metric8, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C))
                      => (cong_area(A,B,C,C,A,B) & cong_area(A,B,C,A,C,B))))).
fof(ax_metric9, axiom, (! [A,B,C,A1,B1,C1] : ((point(A) & point(B) & point(C)
                      & point(A1) & point(B1) & point(C1) & cong(A,B,A1,B1)
                      & cong(B,C,B1,C1) & cong(C,A,C1,A1) & cong_angle(A,B,C,A1,B1,C1)
                      & cong_angle(B,C,A,B1,C1,A1) & cong_angle(C,A,B,C1,A1,B1))
                      => (cong_area(A,B,C,A1,B1,C1))))).

% TRANSFER INFERENCES %

% DIAGRAM-SEGMENT TRANSFER AXIOMS %
fof(ax_segment1, axiom, (! [A,B,C] : ((point(A) & point(B) & point(C)
                      & bet(A,B,C)) => (segment_add(A,B,B,C,A,C))))).
fof(ax_segment2, axiom, (! [A,B,C,P1,P2] : ((point(A) & point(B) & point(C)
                      & circle(P1) & circle(P2) & center(A,P1) & center(A,P2)
                      & onc(B,P1) & onc(C,P2) & cong(A,B,A,C)) => (P1 = P2)))).
fof(ax_segment3_1, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                      & circle(P) & center(A,P) & onc(B,P) & cong(A,C,A,B))
                      => (onc(C,P))))).
fof(ax_segment3_2, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                      & circle(P) & center(A,P) & onc(B,P) & onc(C,P))
                      => (cong(A,C,A,B))))).
fof(ax_segment4_1, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                      & circle(P) & center(A,P) & onc(B,P)
                      & cong_less(A,C,A,B)) => (inside(C,P))))).
fof(ax_segment4_2, axiom, (! [A,B,C,P] : ((point(A) & point(B) & point(C)
                      & circle(P) & center(A,P) & onc(B,P) & inside(C,P))
                      => (cong_less(A,C,A,B))))).
```

```
% DIAGRAM-ANGLE TRANSFER AXIOMS %
fof(ax_angle1_1, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C)
                       & line(L) & A != B & A != C & on(A,L) & on(B,L)
                       & on(C,L) & nbet(B,A,C)) => (cong_angle_zero(B,A,C))))).
fof(ax_angle1_2, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C)
                       & line(L) & A != B & A != C & on(A,L) & on(B,L)
                       & cong_angle_zero(B,A,C)) => (on(C,L) & nbet(B,A,C))))).
fof(ax_angle2_1, axiom, (! [A,B,C,D,L,M] : ((point(A) & point(B) & point(C)
                       & point(D) & line(L) & line(M) & on(A,L) & on(A,M)
                       & on(B,L) & on(C,M) & A != B & A != C & non(D,L)
                       & non(D,M) & L != M  & angle_add(B,A,D,D,A,C,B,A,C))
                       => (sameside(B,D,M) & sameside(C,D,L))))).
fof(ax_angle2_2, axiom, (! [A,B,C,D,L,M] : ((point(A) & point(B) & point(C)
                       & point(D) & line(L) & line(M) & on(A,L) & on(A,M)
                       & on(B,L) & on(C,M) & A != B & A != C & non(D,L)
                       & non(D,M) & L != M  & sameside(B,D,M) & sameside(C,D,L))
                       => (angle_add(B,A,D,D,A,C,B,A,C))))).
fof(ax_angle3_1, axiom, (! [A,B,C,D,L] : ((point(A) & point(B) & point(C)
                       & point(D) & line(L) & on(A,L) & on(B,L) & bet(A,C,B)
                       & non(D,L) & cong_angle(A,C,D,D,C,B))
                       => (right_angle(A,C,D))))).
fof(ax_angle3_2, axiom, (! [A,B,C,D,L] : ((point(A) & point(B) & point(C)
                       & point(D) & line(L) & on(A,L) & on(B,L) & bet(A,C,B)
                       & non(D,L) & right_angle(A,C,D))
                       => (cong_angle(A,C,D,D,C,B))))).
fof(ax_postulate4, axiom, (! [A,B,C,D,E,F] : ((point(A) & point(B) & point(C)
                       & point(D) & point(E) & point(F) & right_angle(A,B,C)
                       & right_angle(D,E,F)) => (cong_angle(A,B,C,D,E,F))))).
fof(ax_angle4, axiom, (! [A,B,B1,C,C1,L,M] : ((point(A) & point(B) & point(B1)
                       & point(C) & point(C1) & line(L) & line(M) & on(A,L)
                       & on(B,L) & on(B1,L) & on(A,M) & on(C,M) & on(C1,M)
                       & B != A & B1 != A & C != A & C1 != A & nbet(B,A,B1)
                       & nbet(C,A,C1)) => (cong_angle(B,A,C,B1,A,C1))))).
fof(ax_angle5, axiom, (! [A,B,C,D,L,M,N,P1,P2,P3,R1,R2,R3,R4,R5,R6,R7,R8,R9,E]
                       : ((point(A) & point(B) & point(C) & point(D) & line(L)
                       & line(M) & line(N) & point(P1) & point(P2) & point(P3)
                       & point(R1) & point(R2) & point(R3) & point(R4)
                       & point(R5) & point(R6) & point(R7) & point(R8)
                       & point(R9) & point(E) & on(A,L) & on(B,L) & on(B,M)
                       & on(C,M) & on(C,N) & on(D,N) & B != C
                       & sameside(A,D,N) & angle_add(A,B,C,B,C,D,P1,P2,P3)
                       & right_angle(R1,R2,R3) & right_angle(R4,R5,R6) \
                       & angle_add(R1,R2,R3,R4,R5,R6,R7,R8,R9)
                       & cong_angle_less(P1,P2,P3,R7,R8,R9) & on(E,L) & on(E,N))
```

```
                    => (intersects(L,N) & sameside(E,A,M))))).

% DIAGRAM-AREA TRANSFER AXIOMS %
fof(ax_area1_1, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C)
                    & line(L) & on(A,L) & on(B,L) & A != B
                    & cong_area_zero(A,B,C)) => (on(C,L))))).
fof(ax_area1_2, axiom, (! [A,B,C,L] : ((point(A) & point(B) & point(C)
                    & line(L) & on(A,L) & on(B,L) & A != B & on(C,L))
                    => cong_area_zero(A,B,C)))).
fof(ax_area2_1, axiom, (! [A,B,C,D,L] : ((point(A) & point(B) & point(C)
                    & point(D) & line(L) & on(A,L) & on(B,L) & on(C,L)
                    & A != B & A != C & B != C & non(D,L) & bet(A,C,B))
                    => (area_add(A,C,D,D,C,B,A,D,B))))).
fof(ax_area2_2, axiom, (! [A,B,C,D,L] : ((point(A) & point(B) & point(C)
                    & point(D) & line(L) & on(A,L) & on(B,L) & on(C,L)
                    & A != B & A != C & B != C & non(D,L)
                    & area_add(A,C,D,D,C,B,A,D,B)) => bet(A,C,B)))).
```

Литература

[1]

[2] Noriko H. Arai and Ryuji Masukawa. How to find symmetries hidden in combinatorial problems. In *Proceedings of the Eighth Symposium on the Integration of Symbolic Computation and Mechanized Reasoning*, pages 321–334, 2000.

[3] Jeremy Avigad, Edward Dean, and John Mumma. A Formal System for Euclid's Elements. *The Review of Symbolic Logic*, 2009.

[4] Henk Barendregt and Freek Wiedijk. The Challenge of Computer Mathematics. *Philosophical Transactions of the Royal Society*, 363(1835):2351–2375, 2005.

[5] Michael Beeson. Proof and computation in geometry. In Tetsuo Ida and Jacques D. Fleuriot, editors, *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2012.

[6] Michael Beeson and Larry Wos. Otter proofs of theorems in tarskian geometry. submitted to IJCAR, 2014.

[7] Stefan Berghofer and Marc Bezem. Implementation of a coherent logic prover for isabelle. 2008.

[8] Marc Bezem and Thierry Coquand. Automating Coherent Logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, volume 3835 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[9] Marc Bezem and Dimitri Hendriks. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic. *Journal of Automated Reasoning*, 40(1), 2008.

[10] Jasmin Christian Blanchette. Redirecting Proofs by Contradiction. In Jasmin Christian Blanchette and Josef Urban, editors, *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013, Lake Placid, NY, USA, June 9-10, 2013*, volume 14 of *EPiC Series*, pages 11–26. EasyChair, 2013.

[11] Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT Solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013.

[12] Jasmin Christian Blanchette, Lukas Bulwahn, and Tobias Nipkow. Automatic Proof and Disproof in Isabelle/HOL. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 12–27. Springer, 2011.

[13] Mathieu Boespflug, Quentin Carbonneaux, and Olivier Hermant. The λΠ-calculus Modulo as a Universal Proof Language. In *Second Workshop on Proof Exchange for Theorem Proving (PxTP)*, volume 878 of *CEUR Workshop Proceedings*, pages 28–43. CEUR-WS.org, 2012.

[14] Karol Borsuk and Wanda Szmielew. *Foundations of Geometry*. Norht-Holland Publishing Company, Amsterdam, 1960.

[15] Pierre Boutry, Gabriel Braun, and Julien Narboux. A formalization of geometry in Coq based on Tarski's axiom system. Unpublished note. http://geocoq.github.io/GeoCoq/, 2015.

[16] Pierre Boutry, Julien Narboux, Pascal Schreck, and Gabriel Braun. A short note about case distinctions in Tarski's geometry. submitted to ADG 2014, May 2014.

[17] Pierre Boutry, Julien Narboux, Pascal Schreck, and Gabriel Braun. Using small scale automation to improve both accessibility and readability of formal proofs in geometry. submitted to ADG 2014, May 2014.

[18] Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques Fleuriot, editors, *Automated Deduction in Geometry 2012*, Edinburgh, United Kingdom, September 2012.

[19] Bruno Buchberger. *An Algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal.* PhD thesis, Math. Inst. University of Innsbruck, Austria, 1965.

[20] Bruno Buchberger, Tudor Jebelean, Franz Kriftner, Mircea Marin, Elena Tomuţa, and Daniela Văsaru. A survey of the theorema project. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, ISSAC '97, pages 384–391, New York, NY, USA, 1997. ACM.

[21] Marco Cadoli and Toni Mancini. Using a theorem prover for reasoning on constraint problems. In Stefania Bandini and Sara Manzoni, editors, *AI*IA 2005: Advances in Artificial Intelligence*, volume 3673 of *Lecture Notes in Computer Science*, pages 38–49. Springer Berlin Heidelberg, 2005.

[22] Ricardo Caferra, Nicolas Peltier, and François Puitg. Emphasizing human techniques in automated geometry theorem proving: A practical realization. In Jürgen Richter-Gebert and Dongming Wang, editors, *Automated Deduction in Geometry*, volume 2061 of *Lecture Notes in Computer Science*, pages 268–305. Springer Berlin Heidelberg, 2001.

[23] S.C. Chou, X.S. Gao, and J.Z. Zhang. *Machine Proofs in Geometry.* World Scientific, Singapore, 1994.

[24] Shang-Ching Chou. *Mechanical Geometry Theorem Proving.* D.Reidel Publishing Company, Dordrecht, 1988.

[25] Shang-Ching Chou and Xiao-Shan Gao. Automated reasoning in geometry. In *Handbook of Automated Reasoning.* Elsevier and MIT Press, 2001.

[26] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated production of traditional proofs for constructive geometry theorems. In Moshe Vardi, editor, *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science LICS*, pages 48–56. IEEE Computer Society Press, June 1993.

[27] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, II. theorem proving with full-angles. *Journal of Automated Reasoning*, 17:349–370, 1996.

[28] 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.

[29] Nicolaas Govert de Bruijn. The Mathematical Vernacular, a Language for Mathematics with Typed Sets. In Dybjer et al., editor, *Proceedings of the Workshop on Programming Languages*, 1987.

[30] Christophe Dehlinger, Jean-François Dufourd, and Pascal Schreck. Higher-order intuitionistic formalization and proofs in hilbert's elementary geometry. In *Automated Deduction in Geometry*, volume 2061 of *Lecture Notes in Computer Science*. Springer, 2001.

[31] Radivoje Despotović, Ratko Tošić, and Branimir Šešelja. *Matematika za I razred srednje škole*. Zavod za udžbenike i nastavna sredstva, Beograd, 2000.

[32] Jean Duprat. Une axiomatique de la géométrie plane en coq. In *Actes des JFLA 2008*, pages 123–136. INRIA, 2008. In french.

[33] Bruno Buchberger et.al. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 2006.

[34] John Fisher and Marc Bezem. Skolem Machines and Geometric Logic. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *4th International Colloquium on Theoretical Aspects of Computing — ICTAC 2007*, volume 4711 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[35] Mohan Ganesalingam and William Timothy Gowers. A fully automatic problem solver with human-style output. *CoRR*, abs/1309.4501, 2013.

[36] H. Gelernter. A note on syntactic symmetry and the manipulation of formal systems by machine. *Information and Control*, 2(1):80 – 89, 1959.

[37] H. Gelernter, J. R. Hansen, and D. W. Loveland. Empirical explorations of the geometry theorem machine. In *Papers Presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '60 (Western), pages 143–149, New York, NY, USA, 1960. ACM.

[38] Gerhard Gentzen. Untersuchungen über das logische Schließen, I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[39] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin, and David Pichardie, editors, *ITP 2013,*

4th Conference on Interactive Theorem Proving, volume 7998 of *LNCS*, pages 163–179, Rennes, France, 2013. Springer.

[40] Frédérique Guilhot. Formalisation en coq d'un cours de géométrie pour le lycée. In *Journées Francophones des Langages Applicatifs*, Janvier 2004.

[41] Haragauri Narayan Gupta. *Contributions to the axiomatic foundations of geometry*. PhD thesis, University of California, Berkley, 1965.

[42] Thomas C. Hales. Introduction to the flyspeck project. In *Mathematics, Algorithms, Proofs*, volume 05021 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[43] John Harrison. Hol light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *Formal Methods in Computer-Aided Design*, volume 1166 of *Lecture Notes in Computer Science*. Springer, 1996.

[44] Thomas L. Heath. *The Thirteen Books of Euclid's Elements*. Dover Publications, New-York, 1956. 2nd ed.

[45] David Hilbert. *Grundlagen der Geometrie*. Leipzig, 1899.

[46] Predrag Janičić and Stevan Kordić. EUCLID — the geometry theorem prover. *FILOMAT*, 9(3):723–732, 1995.

[47] Gilles Kahn. Constructive geometry according to Jan von Plato. Coq contribution, 1995. Coq V5.10.

[48] Cezary Kaliszyk and Alexander Krauss. Scalable LCF-Style Proof Translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2013.

[49] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with flyspeck. *CoRR*, abs/1211.7012, 2012.

[50] Cezary Kaliszyk and Josef Urban. PRocH: Proof Reconstruction for HOL Light. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction*, volume 7898 of *Lecture Notes in Computer Science*, pages 267–274. Springer, 2013.

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

[52] Matt Kaufmann, J. Strother Moore, and Panagiotis Manolios. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[53] Chantal Keller and Benjamin Werner. Importing HOL Light into Coq. In *ITP*, pages 307–322, 2010.

[54] Dongwon Lee and Wesley W. Chu. Comparative analysis of six xml schema languages. *SIGMOD Record*, 29(3):76–87, 2000.

[55] Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing Projective Plane Geometry in Coq. In *Proceedings of ADG'08*, June 2008.

[56] Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing desargues' theorem in coq using ranks. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 1110–1115. ACM, 2009.

[57] Timothy Makarios. A further simplification of Tarski's axioms of geometry. 2013.

[58] Toni Mancini and Marco Cadoli. Detecting and breaking symmetries by reasoning on problem specifications. In Jean-Daniel Zucker and Lorenza Saitta, editors, *Abstraction, Reformulation and Approximation*, volume 3607 of *Lecture Notes in Computer Science*, pages 165–181. Springer Berlin Heidelberg, 2005.

[59] Vesna Marinković. Proof simplification in the framework of coherent logic. Computing and Informatics, 34(2):337–366, 2015.

[60] Laura Meikle and Jacques Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Proceedings of TPHOLs*, volume 2758 of *Lecture Notes in Computer Science*. Springer, 2003.

[61] Laura Meikle and Jacques Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Theorem Proving in Higher Order Logics*, pages 319–334, 2003.

[62] Laura Meikle and Jacques Fleuriot. Mechanical theorem proving in computation geometry. In Hoon Hong and Dongming Wang, editors, *Proceedings of Automated Deduction in Geometry (ADG04)*, volume 3763 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, November 2005.

[63] Laura Meikle and Jacques Fleuriot. Automation for geometry in isabelle/hol. In Renate A. Schmidt, Stephan Schulz, and Boris Konev, editors, *PAAR-2010*, volume 9 of *EPiC Series*, pages 84–94. EasyChair, 2012.

[64] Milan Mitrović, Srđan Ognjanovi'c, Mihailo Veljković, Ljubinka Petković, and Nenad Lazarević. *Geometrija za prvi razred Matematičke gimnazije*. KRUG, Beograd, 1998.

[65] Julien Narboux. *Formalisation et automatisation du raisonnement géométrique en Coq*. PhD thesis, Université Paris Sud, September 2006.

[66] Julien Narboux. Mechanical Theorem Proving in Tarski's Geometry. In *Proceedings of Automated Deduction in Geometry 2006*, volume 4869 of *Lecture Notes in Computer Science*. Springer, 2007.

[67] Julien Narboux. Mechanical theorem proving in Tarski's geometry. In *Proceedings of Automatic Deduction in Geometry 06*, volume 4869 of *Lecture Notes in Artificial Intelligence*, pages 139–156. Springer-Verlag, 2007.

[68] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[69] Benjamin Northrop. Automated diagrammatic reasoning: a proof checker for the language of e. master thesis. Master's thesis, Department of Philosophy, Carnegie Mellon University, 2011.

[70] Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, page 298–302. Springer Berlin Heidelberg, 2006.

[71] Jan von Plato. The axioms of constructive geometry. *Annals of Pure and Applied Logic*, 76:169–200, 1995.

[72] Jan von Plato. Formalization of Hilbert's geometry of incidence and parallelism. *Synthese*, 110:127–141, 1997.

[73] Andrew Polonsky. *Proofs, Types and Lambda Calculus*. PhD thesis, University of Bergen, 2011.

[74] Art Quaife. Automated development of tarski's geometry. *Journal of Automated Reasoning*, 5(1):97–118, 1989.

[75] Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI Communications*, 15(2-3):91–110, 2002.

[76] J. A. Robinson. A machine oriented logic based on the resolution principle. *Journal of ACM*, 12:23–41, 1965.

[77] Piotr Rudnicki. Obvious inferences. *Journal of Automated Reasoning*, 3(4):383–393, 1987.

[78] Stephan Schulz. E - a brainiac theorem prover. *AI Communications*, 15(2-3):111–126, 2002.

[79] Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.

[80] Steffen Juilf Smolka and Jasmin Christian Blanchette. Robust, Semi-Intelligible Isabelle Proofs from ATP Proofs. In Jasmin Christian Blanchette and Josef Urban, editors, *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013*, volume 14 of *EPiC Series*, pages 117–132. EasyChair, 2013.

[81] Sana Stojanović. Preprocessing of the Axiomatic System for More Efficient Automated Proving and Shorter Proofs. In Tetsuo Ida and Jacques Fleuriot, editors, *Automated Deduction in Geometry: 9th International Workshop, ADG 2012. Revised Selected Papers*, pages 181–192. Springer, 2013.

[82] Sana Stojanović, Julien Narboux, Marc Bezem, and Predrag Janičić. A vernacular for coherent logic. In Stephen Watt et.al., editor, *Intelligent Computer Mathematics - CICM 2014*, volume 8543 of *Lecture Notes in Computer Science*. Springer, 2014.

[83] 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 Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*. Springer, 2011.

[84] Vladimir Stojanović. *Matematiskop 3 - Odabrani zadaci (sa rešenjima) za učenike srednjih škola*. Naučna knjiga, Beograd, 1988.

[85] Sana Stojanović Đurđević. Automatsko proveravanje neformalnih dokaza teorema srednjoškolske geometrije. *InfoM*, to appear 2016.

[86] Sana Stojanović Đurđević, Julien Narboux, and Predrag Janičić. Automated generation of machine verifiable and readable proofs: A case study of Tarski's geometry. *Annals of Mathematics and Artificial Intelligence*, 2015.

[87] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[88] M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen.* Studies in Logic and The Foundations of Mathematics. North-Holland Publishing Company, Amesterdam, 1969.

[89] Carst Tankink, Cezary Kaliszyk, Josef Urban, and Herman Geuvers. Communicating formal proofs: The case of flyspeck. In *Interactive Theorem Proving - 4th International Conference, Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 451–456. Springer, 2013.

[90] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975.

[91] Alfred Tarski. What is elementary geometry? In P. Suppes L. Henkin and A. Tarski, editors, *The axiomatic Method, with special reference to Geometry and Physics*, pages 16–29, Amsterdam, 1959. North-Holland.

[92] Alfred Tarski and Steven Givant. Tarski's system of geometry. *The Bulletin of Symbolic Logic*, 5(2), June 1999.

[93] The Coq development team. *The Coq proof assistant reference manual, Version 8.2.* TypiCal Project, 2009.

[94] Andrzej Trybulec. Mizar. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[95] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In *Automated Deduction - CADE-22 Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 140–145. Springer, 2009.

[96] Markus Wenzel. Isar - A Generic Interpretative Approach to Readable Formal Proof Documents. In Yves Bertot, Gilles Dowek, André Hirschowitz, C. Paulin, and Laurent Théry, editors, *Theorem Proving in Higher Order Logics (TPHOLs'99)*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 1999.

[97] Freek Wiedijk. Mathematical Vernacular. Unpublished note. http://www.cs.ru.nl/~freek/notes/mv.pdf, 2000.

[98] Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[99] Freek Wiedijk. A Synthesis of the Procedural and Declarative Styles of Interactive Theorem Proving. *Logical Methods in Computer Science*, 8(1), 2012.

[100] Wen-Tsün Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.

Биографија аутора

Магистар Сана Стојановић рођена је 1. маја 1981. године у Београду, где је завршила основну школу Десанка Максимовић и Математичку гимназију. Студије на смеру Рачунарство и информатика на Математичком факултету у Београду уписала је 1999. године. Током студија била је добитник стипендије краљевине Норвешке, стипендије скупштине града Београда за 100 најбољих студената, стипендије Министарства просвете и спорта и стипендије Републичке фондације за развој научног и уметничког подмлатка.

Дипломирала је 2005. године са просечном оценом 9.72 (од 10.00) и уписала магистарске студије на истом смеру. Исте године била је изабрана у звање асистента приправника на Математичком факултету. Магистарски рад под називом „Анализа особености геометријских карактеристика глицина у протеинима" одбранила је 2009. године. Исте године је изабрана у звање асистента на Математичком факултету. Током досадашњег рада на Математичком факултету држала је вежбе из девет предмета, на редовним и мастер студијама, и била члан комисије осам мастер теза.

Бави се истраживањима у области аутоматског и интерактивног доказивања теорема, кохерентне логике и формализације геометрије. Учествовала је у раду и организацији више међународних радионица, летњих школа и конференција. Објавила је радове на водећим међународним форумима из области аутоматског доказивања теорема у рачунарству. Члан је групе за аутоматско резоновање Математичког факултета.

Изјава о ауторству

Име и презиме аутора Сана (Небојша) Стојановић

Број индекса / (по старом)

Изјављујем

да је докторска дисертација под насловом

Формализација и аутоматско доказивање теорема еуклидске геометрије

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 15.06.2016.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Сана (Небојша) Стојановић

Број индекса / (по старом)

Студијски програм Рачунарство и информатика

Наслов рада Формализација и аутоматско доказивање теорема еуклидске геометрије

Ментор др Предраг Јаничић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива 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 аутора

У Београду, 15.06.2016.

_____

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Формализација и аутоматско доказивање теорема еуклидске геометрије

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

(1.) Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 15.06.2016.

---

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 прерада**. 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 прерада**. 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.