Formalization and Automation of Euclidean Geometry

Vesna Pavlović, Sana Stojanović
Faculty of Mathematics, Belgrade
Spring School Geometry and Visualization,
Belgrade, Serbia, April 22, 2008.

Our Plan

- Formal theorem proving
- Formalization of Euclidean geometry
- Automation of Euclidean geometry

Our Plan

- Formal theorem proving
- Formalization of Euclidean geometry
- Automation of Euclidean geometry

What is a Proof?

To prove (Merriam-Webster)

- from Latin probare (test, approve, prove)
- to learn or find out by experience (archaic)
- to establish the existence, truth, or validity of (by evidence or logic)

What is a Mathematical Proof?

In mathematics, a proof is a demonstration that, given certain axioms, some statement of interest is necessarily true. (Wikipedia)

Example: $\sqrt{2}$ is not rational.

Proof: Assume there is $r \in Q$ such that $r^2 = 2$. Hence there are mutually prime p and q with $r = \frac{p}{q}$. Thus $2q^2 = p^2$, i.e. p^2 is divisible by 2. 2 is prime, hence it also divides p, i.e. $p = 2s$. Substituting this into $2q^2 = p^2$ and dividing by 2 gives $q^2 = 2s^2$. Hence, q is also divisible by 2. Contradiction.

Nice, but...

- Still not rigorous enough for some.
  - What are the axioms? What are the rules?
  - How big can the steps be?
  - What is obvious or trivial?
- Informal language.

Theorem: A cat has nine tails.
Proof: No cat has eight tails.
One cat has one more tail than no cat.
Hence it must have nine tails.

What is a Formal Proof?

- A derivation in a formal calculus

$\Lambda_1, \Lambda_1, \ldots \Lambda_k$ - axioms and previously proved theorems

Formal proof of a sentence P is a sequence of statements

$$S_1, S_2, \ldots S_n$$

where:

1. S_n is P and one of the following holds:

2. - S_i is one of $\Lambda_1, \Lambda_1, \ldots \Lambda_k$
   - S_i follows from the previous statements by a valid argument using the rules of reasoning

Example of a Formal Proof

Example: $A \wedge B \rightarrow B \wedge A$ is derivable in the following system:

$\frac{X \in S}{S \vdash X}$ (assumption) $\qquad$ $\frac{S \cup \{X\} \vdash Y}{S \vdash X \rightarrow Y}$ (impI)

$\frac{S \vdash X \quad S \vdash Y}{S \vdash X \wedge Y}$ (conjI) $\qquad$ $\frac{S \cup \{X, Y\} \vdash Z}{S \cup \{X \wedge Y\} \vdash Z}$ (conjE)

Proof:

1. $\{A, B\} \vdash B$ (by assumption)
2. $\{A, B\} \vdash A$ (by assumption)
3. $\{A, B\} \vdash B \wedge A$ (by conjI with 1 and 2)
4. $\{A \wedge B\} \vdash B \wedge A$ (by conjE with 3)
5. $\{\} \vdash A \wedge B \rightarrow B \wedge A$ (by impI with 4)

Importance of Having Formal Proofs

- Books and journals full with faulty proofs
  (not necessarily faulty statements)

- Correctness of the software and hardware components
  must be confirmed as formally as it can be

What is a Formal Verification?

Formal verification is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics.

Two approaches to formal verification:

1. Model checking
   - a systematically exhaustive exploration
   of the mathematical model

2. Logical inference
   - using a formal version of
   mathematical reasoning about the system

What is a Theorem Prover?

Implementation of a formal logic on a computer

- Fully automated (propositional logic)
- Automated, but not necessarily terminating (first-order logic)
- With automation, but mainly interactive (higher-order logic)
- Based on rules and axioms
- Can deliver formal proofs

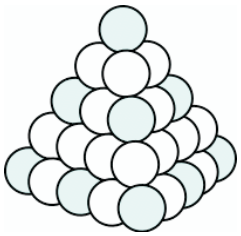
Theorem Provers

Most important theorem provers:

- HOL Light (John Harrison)
- Isabelle/Isar (Lawrence C. Paulson, Tobias Nipkow, Markus Wenzel)
- Coq (Benjamin Werner, Georgies Gonthier)
- Mizar (Andrzej Trybulec)
- ProofPower (Roger Jones, Rob Arthan)

Example: Kepler's conjecture

Importance of formalization of the statements whose proofs are not so obvious and trivial to understand



- Filling a large container with small equal-sized spheres
- Aim is to maximize the density of arrangement
  - Random packing - 65%
  - Cubic close packing $\frac{\pi}{\sqrt{18}} \simeq 0.74048.$
- Thomas Hales - proof by exhaustion
  its formalization is estimated on 20 person-years

"The Hundred Greatest Theorems"

Paul & Jack Abad, 1999.

Criteria:

- place which the theorem holds in the literature
- quality of the proof
- unexpectedness of the result

Their formulation and formalization can be found on:
http://www.cs.ru.nl/~freek/100/

About 80% of these theorems are formalized

Isabelle - Basic Concepts

- Interactive theorem proving framework
- Successor of HOL theorem prover
- Natural deduction is the main deduction system used
- Includes mechanism for term rewriting and tableaux prover
- Used for confirming correctness of security protocols, properties of programming language semantics, formalizing theorems from mathematics and CS

Isabelle - Basic Concepts

Example:

- Mathematics: if $x < 0$ and $y < 0$ then $x + y < 0$
- Formal logic: $\vdash x < 0 \wedge y < 0 \rightarrow x + y < 0$
  variation: $\{x < 0; y < 0\} \vdash x + y < 0$
- Isabelle: lemma " $x < 0 \wedge y < 0 \rightarrow x + y < 0$"
  variation: lemma: "$[|x < 0; y < 0|] \Rightarrow x + y < 0$"

- Isabelle/Isar: lemma
  assumes "$x < 0$" and "$y < 0$"
  shows "$x + y < 0$"

Isabelle - Basic Concepts

Isar formal proof language has been designed to satisfy quite contradictory requirements: being both 'declarative' and immediately 'executable'

It is Isabelle's language for readable proof documents

- it is possible to name proposition in definitions, lemmas, and proofs by "name: proposition"
- the name ?thesis always stands for the current goal
- ".." is an abbreviation for by (rule name) if name refers to one of the predefined introduction rules
- Intermediate steps of the proofs can be of the form:
  from *fact1* and *fact2* ... and *propostions*(show|have)*proposition*

Our Plan

- Formal theorem proving
- Formalization of Euclidean geometry
- Automation of Euclidean geometry

Formalization of Hilbert's Axiomatic System

- Euclid's "Elements"
- Hilbert's "der Grundlagen der Geometrie"
- Formalization:
  - Christophe Dehlinger, Jean-Francois Dufourd, Pascal Schreck: Coq proof assistant
  - Jacques Fleuriot, Laura Meikle: Isabelle/Isar proof assistant

Formalization of Hilbert's Axiomatic System (Fleuriot, Meikle)

- Procedure:
  - declaring the three primitives as types: point, line and plane
    typedecl Point
  - declaring basic relations
    consts incident :: "Point => Line => bool"
  - defining additional relations
    definition "colinear A B C == ? (l::Line). incident A l & incident B l & incident C l"
  - formulating axioms
    axioms I1: "?(A::Point)(B::Point).line l A B & A ~= B"
- Example in Isabelle/Isar (by Filip Marić)
  http://www.matf.bg.ac.yu/~vesnap/spring_school/geometry.doc

Formalization of Tarski's Axiomatic System

- Alfred Tarski's axiomatic system

- Wolfram Schwabhauser:
  Metamathematische Methoden in der Geometrie

- Formalization:

  – Julien Narboux:
    Coq proof assistant

Formalization Geometry in Proof Assistant (Narboux)

- Advantages:
  - provides very high level of confidence in the proof generated
  - permits to insert purely geometric arguments within other kind of proofs
- Problem of the degenerated cases
  - Source: axiomatic system

Why Tarski Axioms? (Narboux)

Advantages:

- They are simple (11 axioms and 2 predicates)
- Good meta-mathematical properties provide very high level of confidence in the proof generated
- Generalization to other dimensions is easy

Drawbacks:

- Lemma scheduling is more complicated
- It is not well adapted to teaching

Proving Theorems

We can split methods for proving theorems in geometry into two categories, these are:

1. Algebraic methods

2. Coordinate-free methods

Algebraic Methods

- *Hilbert* - decision method for a class of constructive geometry statements in affine geometry
- *Tarski (1951)* - decision method for the theory of real closed fields
- *Collins (1974)* - cylindrical algebraic decomposition (CAD) algorithm
- *Wu (1977)* - a breakthrough in automated theorem proving
- *Ritt (1984)* - algebraic aspect of this method is known as the Wu-Ritt-s characteristic set (CS) method
- *Buchberger* - Gröbner basis method
- *Kapur (1997)* - Dixon resultant approach

Coordinate-free Methods

- *Chou, Gao, Zhang (1993)* - area method
- *Stiffer (1993)* - Gröbner basis method
- *Chou, Gao, Zhang (1994)* - angle method

Solving Constructive Problems in Geometry

- generation of steps in geometric constructions
- solving hard problems in geometry
- producing effectively multiple and the shortest solutions of geometric theorems

Solving Constructive Problems in Geometry

- *Gelernter's* geometry machine (1959)
  - backward chaining
  - reliance on a diagram to guide a proof
- *Gao, Chou (1998)* - a global propagation method for automated generation of construction steps
  - rc-configuration is a diagram that can be constructed with ruler and compass
  - forward & backward chaining
  - using GIB (database containing all the properties of the configuration that can be deduced using a fixed set of geometric axioms)

Our Aim

- Idea of formalization of geometrical reasoning is quite new
- Instead of proving Hilbert's theorems "manually" in Isabelle/CoQ we have an idea of the automation of whole process
- Obtaining formal verification of the proofs

Our Plan

- Formal theorem proving
- Formalization of geometry
- Automation of Euclidian geometry

Automation of Euclidian Geometry

- We will study geometry as a formal system with *different systems of axioms*
- Correctness which characterize the *deduction process* itself will be satisfactory criterium
- Usage of some of the usual *deduction systems* (the ones that are close to human intuition and *available in Isabelle*)
- Distinction between syntax (and process of formal deduction) and semantics (meaning of deduction in terms of intuition)

EUCLID - the Geometry Theorem Prover

- Authors: Predrag Janicic and Stevan Kordic
- Proves theorems of geometry in an intuitive, geometrical way
- Proofs are presented in natural language form
- New form of the foundation of geometry and new classification of geometrical axioms
- The prover EUCLID determines strict structure of axioms and classification of axioms

Axiomatic of Euclidian geometry in system EUCLID

Basic symbols:

- Logic symbols: $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\forall$, $\exists$
- Variables: $x_1, x_2, x_3, \ldots$
- (Derived) constants $a_1, a_2, a_3, \ldots$
- Predicates:
  $\mathcal{S}$ - point, $\mathcal{L}$ - line, $\mathcal{P}$ - plane
  $=$ - identical
  $\mathcal{I}$ - incident
  $\mathcal{B}$ - between
  $\mathcal{C}$ or $\cong$ - congruence

Axiomatic of Euclidian geometry in system EUCLID

- Other types of geometrical objects (line segment, triangle, circle, inside of a circle, etc.) could be introduced by definitions
- This would lead to enhanced segment of "traditional" geometry that could be covered with this theory
- Justification for doing this:
  - In this manner, without using theory of sets, we can cover a large part of usual geometrical components
  - Provide properties that certain objects should have
  - Axiomatic system built in this manner would still preserve independence of axioms

List of (primitive and defined) predicates in system EUCLID

Predicate	We read
$\mathcal{S}(a)$	a is a point
$\mathcal{L}(b)$	b is a line
$\mathcal{P}(c)$	c is a plane
$a = b$	a is identical to b
$\mathcal{I}(a, b)$	a is incident to b
$\mathcal{B}(a, b, c)$	b is between a and c
$\mathcal{C}(a, b, c, d)$	(a, b) matches (c, d)
$(a, b) \cong (c, d)$	(a, b) matches (c, d)
$colin(a, b, c)$	a, b and c are colinear
$copl(a, b, c, d)$	a, b, c and d are coplanar
$intersect(a, b)$	a and b intersect

Types of axioms

- Basic axioms (initiation of types of geometrical objects and range of basic relations) are used implicitly
- Axioms of equality
- Non-productive axioms
- Branching axioms
- Productive axioms
- Strongly productive axioms

Definitions that are used in axioms

- Collinear points

  $\forall A \forall B \forall C \exists a(colin(A, B, C) \Rightarrow (\mathcal{I}(A, a) \wedge \mathcal{I}(B, a) \wedge \mathcal{I}(C, a)))$

  $\forall A \forall B \forall C \forall a((\mathcal{I}(A, a) \wedge \mathcal{I}(B, a) \wedge \mathcal{I}(C, a)) \Rightarrow colin(A, B, C))$

- Coplanar points

- Intersection of lines

- Intersection of a line and a plane

- Intersection of planes

Basic axioms

$$\forall x((\mathcal{S}(x) \wedge \neg\mathcal{L}(x) \wedge \neg\mathcal{P}(x)) \vee (\neg\mathcal{S}(x) \wedge \mathcal{L}(x) \wedge \neg\mathcal{P}(x)) \vee (\neg\mathcal{S}(x) \wedge \neg\mathcal{L}(x) \wedge \mathcal{P}(x)))$$

$$\forall x \forall y(\neg(\mathcal{S}(x) \wedge \mathcal{S}(y)) \wedge \neg(\mathcal{L}(x) \wedge \mathcal{L}(y)) \wedge \neg(\mathcal{P}(x) \wedge \mathcal{P}(y)) \Rightarrow \neg(x = y))$$

$$\forall x \forall y(\neg(\mathcal{S}(x) \wedge \mathcal{L}(y)) \wedge \neg(\mathcal{S}(x) \wedge \mathcal{P}(y)) \wedge \neg(\mathcal{L}(x) \wedge \mathcal{P}(y)) \Rightarrow \neg\mathcal{I}(x, y))$$

$$\forall x \forall y \forall z(\neg\mathcal{S}(x) \vee \neg\mathcal{S}(y) \vee \neg\mathcal{S}(z) \Rightarrow \neg\mathcal{B}(x, y, z))$$

$$\forall x \forall y \forall z \forall u(\neg\mathcal{S}(x) \vee \neg\mathcal{S}(y) \vee \neg\mathcal{S}(z) \vee \neg\mathcal{S}(u) \Rightarrow \neg(x, y) \cong (z, u))$$

Axioms of equality and consistency

$$\forall x\,(x = x)$$

$$\forall x\,\forall y\,(x = y \Rightarrow y = x)$$

$$\forall x_1 \forall x_2 \ldots \forall x_n \forall y\ (x_i = y \wedge \Phi(x_1, x_2, \ldots, x_i, \ldots, x_n)$$

$$\Rightarrow \Phi(x_1, x_2, \ldots, y, \ldots, x_n))$$

Non-productive axioms

1. If point A incidents line p, and line p incidents plane ϕ, than point A incidents plane ϕ

$$\forall A \forall p \forall \phi (\mathcal{I}(A, p) \wedge \mathcal{I}(p, \phi) \Rightarrow \mathcal{I}(A, \phi))$$

2. In a plane ϕ there can be drawn through any point A, lying outside of a straight line a, one and only one straight line which does not intersect the line a. This straight line is called the parallel to a through the given point A.

Branching axioms

1. Of any three points situated on a straight line, there is always one and only one which lies between the other two

2. If A, B, C are three non-collinear points that are contained in a plane ϕ, a line contained in ϕ and A is not contained in a, and U is a point contained in a, where $\mathcal{B}(B, U, C)$, than line a contains point V, for which holds $\mathcal{B}(C, V, A)$ or point W, for which holds $\mathcal{B}(A, W, B)$

Productive axioms

1. If two planes α and β have a point A in common, then they have at least a second point B in common

2. Two distinct points A and B always completely determine a line a

3. Three points A, B, C not situated in the same straight line always completely determine a plane α

Strongly productive axioms

There exists four different non-coplanar points

$\exists X \exists Y \exists Z \exists U (X \neq Y \wedge X \neq Z \wedge X \neq U \wedge Y \neq Z \wedge Y \neq U \wedge Z \neq U \wedge \neg copl(X, Y, Z, U))$

Simplification of proof deduction, simple theorems

1. If point A is incident to a line a than there exists point B, different from A, which also incidents a

2. If point A is incident to a plane α than there exists points B and C incident to a plane α and A, B and C are not collinear

3. If different points A and B are incident to a plane α than there exists point C incident to a plane α and A, B and C are not collinear

Classification of structure of axioms

All axioms (basic and introduced), and also theorems and definitions, covered with EUCLID have one of the following forms:

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 \exists y_2 \ldots \exists y_m (\Phi(x_1, x_2, ..., x_n) \Rightarrow \quad (1)$$

$$\Psi(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m)) \quad (n, m \geq 1)$$

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 \exists y_2 \ldots \exists y_m (\Phi(x_1, x_2, ..., x_n) \Rightarrow \quad (2)$$

$$\Psi_1(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m) \vee \Psi_2(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m) \vee ...$$

$$... \vee \Psi_k(x_1, x_2, ...x_n, y_1, y_2, ...y_m)) \quad (n, m \geq 1)$$

Classification of structure of axioms

$$\forall x_1 \forall x_2 \ldots \forall x_n (\Phi(x_1, x_2, ..., x_n) \Rightarrow \Psi(x_1, x_2, ..., x_n)) \quad (n \geq 1) \quad (3)$$

$$\exists y_1 \exists y_2 \ldots \exists y_m (\Psi(y_1, y_2, ..., y_m)) \quad (m \geq 1) \quad (4)$$

where Φ, Ψ and Ψ_i are literals with variables $x_1, x_2, ..., x_n$, apropos $y_1, y_2, ..., y_m$

Classification of structure of axioms

- Structure of the axioms and effectiveness of deduction of proofs and nature of prover EUCLID motivated new classification of geometrical axioms.

- By this classification, geometrical axioms are not divided (as usual) to axioms of incidence, axioms of order, axioms of congruence, axioms of parallels and axioms of continuity, so they are not divided by their "topic" but by their structure

Priority of the axioms

- This arrangement of axioms, that was derived from concept of prover EUCLID, naturally corresponds to form of structures of axioms that we mentioned above. Order of axioms wasn't determined exactly, but in regard to property of group of axioms and through numerous experiments.

- Arrangement of axioms inside these groups is important for effectiveness of deduction of proofs, but it doesn't affect set of theorems that prover EUCLID can prove.

- Strongly productive axioms are axioms that often produce new objects. If we reduce its application we will increase efficiency of proof deduction.

System for proof deduction

- Proof deduction was based on systems of syllogisms
- Predicate calculus (Leibniz)
- Calculus of reasoning (Boole)
- Concept notation (Frege, formal language and properties of quantifiers, modus ponens)
- Axiomatic for predicate calculus (Post, Hilbert and Ackermann; proved property of completeness)

System for proof deduction

From there on, development of theory of proofs is consistent with development of automated theorem proving

- Skolem, Herbrand (systems for automated theorem proving)
- Bet (tableau method)
- Robinson (method of resolution)
- Gentzen (natural deduction)

Systems of deduction of proof in prover EUCLID

- Rules of deduction (in bases of predicate calculus) used in EUCLID are designed to be as close as they can to traditional, intuitive proofs

- Rules of deduction must be precise with minimum redundance

- System of deduction is somewhere between Gentzen's and Hilbert's concept

- Restriction - it can be used to prove theorems from just one class written in concrete form

Forms of the theorem provable by EUCLID

- Sentences of form

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 \exists y_2 \ldots \exists y_m \Theta(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_m)$$

  will be written in short as: $\forall \vec{x} \exists \vec{y} \Theta(\vec{x}, \vec{y})$

- In prover EUCLID theorems that we prove have one of the following forms:

$$\forall \vec{x} \exists \vec{y} \Theta(\vec{x}, \vec{y})$$
$$\forall \vec{x} \Theta(\vec{x}, \vec{y})$$
$$\exists \vec{y} \Theta(\vec{x}, \vec{y})$$

- Notice that all axioms of this theory already have one of the above forms.

Algorithms for automated theorem deduction

- Algorithm "British Museum", based on successive syntactic deduction of all formulas and checking whether the goal is proven
- System "Logic Theory Machine", involve heuristics which steer "British Museum" algorithm according to structure of theorem that is proved
- System "Geometry Machine", for proving small class of geometrical axioms

Algorithms for automated theorem deduction

- One of the first systems for automated theorem proving, Paul Gilmor (1960), based on Herbrand's theorem and its implications
  - Very ineffective and only very simple axioms were provable
  - Automated theorem proving was possible
- Method of resolution of Alan Robinson. Resolved some of the problems that implied non effectiveness of Gilmor's system.
  - Rule of resolution (replaced usual axioms and rules of deduction in predicate calculus)
  - Involved new significant procedure - unification
  - Increased number of theorems that could be proved

Automated theorem proving

- Systems for automated theorem proving based on method of resolution (specific choice of axioms and clauses for resolution and numerous heuristics)
- Even with its improvements only simple theorem were provable
- Some problems that emerged:
  - Ineffectiveness
  - Result was affirmative or negative answer to the question whether given statement is theorem
- These were all uniform proof procedures. After that, research went towards building specific systems specialized for concrete theories and often, just parts of theory

Algorithm for proof deduction in prover EUCLID

- Axiomatic system is base of algorithm for proof deduction in system EUCLID.
- Algorithm is independent from concrete computer implementation
- Algorithm covers one class of theorems
- In basic version, deduction of proofs is directed by classification of axioms and their's order inside groups
- Lots of space for heuristics

Algorithm for proof deduction in prover EUCLID

- *Acceptable objects* - notion derived from *sentinel principle* used in the prover.

- At the beginning of the proof deduction, set of acceptable objects is empty and during deduction set is expanded. This extensibility is controlled with *sentinel* which disables "infinite branches" in proof

- During proof deduction *knowledge base* is expanded. It contains facts on objects which are generated during proof deduction

Algorithm for proof deduction in prover EUCLID

- Algorithm is partly based on "method of exhaustion". In order to maximally restrain induction of new geometrical objects and facts necessary for proof and in order to maximize efficiency, axioms are divided into three groups and ranked within a group.

- Generated proofs could be (automatically) optimized, so that there are no redundant steps.

- Method used in EUCLID is "forward chaining" and is related to Herbrand's theorem. It is not very efficient but it enables automatic deduction of many proofs that are currently being produced manually.

Program EUCLID

- PROLOG version
- C version
- C++ version

Program EUCLID - C version

- Knowledge base is represented with set of arrays (one array for positive and one for negative form of a predicate).

- Relations does not hold information on types of objects. Example: $I(1,2)$ and $S(1)$ and $L(2)$

- One enumerator for all elementary objects (points, lines, planes).

- For every array, index of last added fact is saved (LIFO list).

Program EUCLID - C version

- Static organization of data (advantage over dynamic - speed of simple operations: adding new fact and erasing last added).

- Axioms are represented by functions (within whom adding new facts in knowledge base and output of proof steps in natural language is carried out).

- Axioms are hard-coded. We cannot insert new lemmas or theorems into a system.

One of the axioms in C version of prover EUCLID

- If a point A lies on a line p, and line p lies on a plane α than point A lies on a plane α

- ```
  int ax_n10()
  {
  int i1,i2;
  int x,y,z;

  for(i1=1;i1<=INC[0].index;i1++)
  {
      x=INC[i1].arg1; y=INC[i1].arg2;
      for(i2=1;i2<=INC[0].index;i2++)
            if (INC[i2].arg1==y)
            {
  ```
  ```

```
                z=INC[i2].arg2;
                if (!inc(x,z))
                {
                   add_inc(x,z);
                   sprintf(OUT,"If point %i lies on line %i
                           and ",x,y);
                   sprintf(OUT,"line %i lies on a plane %i, ",z,y);
                   sprintf(OUT,"than point %i lies on %i",x,z);
                   output(DEPTH,OUT);
                   return 1;
                }
          }
}
return 0;
}
```

Format of input file

If p and q are two lines that intersect than exists a plane which contains them both:

```
premise
line(1)
line(2)
not_identical(1,2)
intersect(1,2)

theorem
plane(-1)
incident(1, -1)
incident(2, -1)
```

Format of input file

- Premise and theorem are obligatory.

- Constants that occur in theorem formulation are denoted with natural numbers.

- Variables that are bounded by existential quantifier are denoted with negative numbers (prover tries to unify these variables with inducted geometrical objects which satisfies theorem)

Program EUCLID - C++ version

- Knowledge base is represented with class *DataBase*.
  - Points, lines and planes are represented with separate counters.
  - Relations hold information on types of objects. Multiple relations of incidence, equality, etc.

    Example: $I_1(1,2)$ implies that 1 is a point and 2 is a line
- We do not have to check weather an object is in a base, we just have to check if its number is less or equal of counter of that type of object.
- We do not preserve properties as we did before. We just store hashes of properties (since only two operations are verification of a property and insertion of new property).

Program EUCLID - C++ version

- Axioms and theorems are interpreted in the same manner:

```
class Statement{
        vector<Property> _from;   // premise
        vector<Property> _have;   // conclusion
        Statement* _by;           // statement which is applied
}
```

- Now we can, once a statement is proven, import it into a base and use it as an axiom.

Program EUCLID - C++ version

- Ideas:
  - Points, lines and planes can be represented as relations as well
  - Relations can be allowed to unify objects (because they contain all needed information for unification)
  - Change order of premise in theorem

    Example:

    Instead of usual order of premises:

    `L(1) || S(1) S(2) Diff(1,2) I1(1,1) I1(2,1)`

    we can arrange premises so that relations are next to the elementary object they describe:

    `L(1) || S(1) I1(1,1) S(2) Diff(1,2) I1(2,1)`

Program EUCLID - C++ version

Output of program:

- Natural language
- Isabelle/Isar
- Coq