1. Introduction

For over two millenia, geometry has been one of the major topics in education all around the world. Since Euclid's ,,Elements", geometry has been a central field for introducing students to deduction and rigorous argumentation.

In recent years, computers and technology have been intensively used to change how geometry is taught. Dynamic geometry systems such as GeoGebra, Cinderella,Geometer's Sketchpad, Cabri, Eukleides [**?**,**?**] are now regularly used in all levels of education. Students use such systems to perform geometric constructions, and obtain diagrams that can be distored by moving free points. Such dynamic diagrams are better than static images, since moving free points can shed additional light to the problem, reveal degenerate cases, help student to determine if something is true only if some special order of points is considered (for example, some property could be true only if a point if between some other two points, and be false if that is not the case, some property could be true only for accute, and not for obtuse angles etc.).

By doing extensive distortion of diagrams by moving free points, student can be pretty sure if a property is generaly true (i.e., true in all, but a small number of degenerate cases), but still, that cannot be considered to be a proof, and is error prone. Therefore, recently dynamic geometry systems have been extended by automated reasoning systems, that can automatically prove statements about constructed objects [**?**]. Such theorem provers are usually algebraic (they perform calculations on symbolic coordinates of geometric objects).

Most recent research in both dynamic geometry systems and automated theorem proving in geometry has been devoted only to two-dimensional Euclidean geometry (plane geometry). Although there have been some limited attempts to apply algebraic theorem proving methods to three-dimensional Euclidean space geometry (solid geometry), we are not aware that there are thorough descriptions of these methods, nor publicly available implementations of automated provers for solid geometry. In this paper we examine and compare several ways of applying algebraic theorem provers based on Gröbner bases and Wu's method to solid geometry problems and describe their integration with dynamic geometry systems for solid geometry. We provide an implementation of one dynamic geometry system for solid geometry, capable of proving statements about the properties of constructed objects. We also analyze a corpus of problems from solid geometry and evaluate methods to those problems. We discuss challenges and possible applications in the field of geometry education.

2. Related work

Automated reasoning in plane geometry. Automated theorem proving has a history more than fifty years long. Most successful automated theorem methods are those for proving in geometry. Automated theorem provers for geometry are based either on the synthetic approach or on the algebraic approach (based on some coordinates).

Major breakthrough in the algebraic approach was made by Wu [**?**]. The geometric construction and the statement are first encoded as a set of polynomial equalities over their coordinates, and then the Wu's method uses algebraic techniques for dealing with these polynomials. In his paper [**?**], Ritt described similar ideas as Wu, so nowdays this method is usually called Ritt-Wu's method. Many theorems were successfully proved using this method []. The success of the Wu's method motivated other researcher to develop new methods, and one of the most prominent is the Gröbner basis method based on Buchberger's algorithm []. Similar to Wu's method, the Gröbner basis also reasons representation in form of a set of polynomial equalities. Both methods produce only yes or no answers instead of human understandable proofs and cannot deal with with inequality (therefore cannot reason about the order of poins).

Several coordinate-free methods that do not use coordinate representation of points are developed in the 1980's. The reason for their development was in the idea that it is possible to develop prover that would produce human readable proofs. Most prominent are the area method [**?**] and the full angle method [**?**], and those methods are usually referred to as semi-algebraic, since they involve reasoning over some special geometric quantities (e.g., signed area or Pythagora's differences). These methods are usually not as efficient as algebraic methods and have smaller scope than algebraic methods.

One of the first synthetic provers, that uses method of resolution, was developed by Gelertner [**?**] whose idea was to develop the proof that is similar to human proof. He successfully proved many problems taken from high-school textbooks. In modern times, synthetic provers are usually based on coherent (synthetic) logic – a special fragment of first order logic convenient for geometric reasoning. For example, the prover ArgoCLP [**?**] uses coherent logic to produce human readable proofs.

Automated reasoning in solid geometry. Shang-Ching Chou at al. presented volume method [**?**]. It is a semi-algebraic method that is extension of the area method for solid geometry. Hypotheses can be described constructively and conclusions are polynomial equations of several geometry quantities, such as volumes, ratios of line segments, ratios of areas, and Pythagoras differences. The key idea of the method is to eliminate points from the conclusion of a geometry statement using several basic propositions about volumes.

To the best of our knowledge, there are no papers that describe application of Wu's or Gröbner basis method into problems in solid geometry.

3. Background

3.1. Algebrization of geometry relations

We shall assume that geometry problems are given in terms terms of relations between the geometric object involved (points, lines, planes, circles, ...). Each theorem under consideration is given in form of an implication, and both hypothesis and conclusions are given as one or more geometric relations. In some cases, the properties of constructed geometric objects should be proved. The construction is usually specified by a series of construction steps, and each construction step is a function that builds new geometric objects out of the existing ones. However, the properties of the new object can usually be specified in terms of one or more relations, so without losing generality, it can be assumed that we are dealing only with relations.

The standard algebrization procedure introduces fresh symbolic variables for point coordinates and introduces (polynomial) equations that characterize every relation between objected observed.

3.2. Algebraic methods

Algebraic methods in geometry are well described in literature []. In this section we give a brief account on algebraic methods in geometry, in order to explain some relevant steps needed in algebraization of solid objects.

Once the geometric theorem has been algebrized, it is possible to apply algebraic theorem proving methods. During algebraization are gained polynomials equations describing objects of the form $p(v_1, \ldots, v_n) = q(v_1, \ldots, v_n)$. Before applying any algebraization method, these equations are transformed to $p(v_1, \ldots, v_n) - q(v_1, \ldots, v_n) = 0$. Lets denote polynomials (left side of equations) representing relation between objects width f_i, $i = 1, \ldots, k$, and lets denote relations to be proved with g_j, $j = 1, \ldots, l$, then proving theorem reduces to checking if for each g_j it holds that

$$\forall v_1, \ldots, v_n \in \mathbb{R}. \bigwedge_{i=1}^{k} f_i(v_1, \ldots, v_n) = 0 \Longrightarrow g_j(v_1, \ldots, v_n) = 0$$

As Tarski suggested, proving geometric properties can be done using quantifier elimination procedure for the reals, but this approach is inefficient and proving nontrivial geometric properties would take too long. But, there is another approach. The main insight, given by Wu Wen-tsün in 1978, is that remarkably many geometrical theorems, when formulated as universal algebraic statements in terms of coordinates, are also true for all complex values of the coordinates. However, a care should be taken, since, in some cases, the condition holds for $\mathbb{R}$, but not for $\mathbb{C}$, and in these cases methods fail due to counterexamples in $\mathbb{C}$. The statement to be proved is following:

$$\forall v_1, \ldots, v_n \in \mathbb{C}. \bigwedge_{i=1}^{k} f_i(v_1, \ldots, v_n) = 0 \Longrightarrow g_j(v_1, \ldots, v_n) = 0$$

This is true when g_j belongs to the radical ideal $I = \langle f_1, \ldots, f_k \rangle$, i.e. when exists an integer r and polynomials $h_1, \ldots, h_k$ such that $g_j^r = \sum_{i=1}^{k} h_i f_i$.

The two most significant algebraic methods that use a kind of Euclidean division to check the validity of an observed conjecture are Buchbergers method consists in transforming the generating set into a Gröbner basis and the Wus method.

Wu's method The first step of simple Wus method [] uses the pseudo–division operation to transform the polynomial system to triangular form, i.e., to a system of equations where each successive equation introduces exactly one dependent variable. After that, the final reminder is calculated by pseudo dividing polynomial for the statements to be proved by each polynomial from triangular system. Summarizing, Wus method, in its simplest form, allows to compute some polynomials $c, h_1, \ldots, h_k$ and r such that

$$cg_j = \sum_{i=1}^{k} h_i f_i + r$$

If the final remainder r is equal to zero, then the conjecture is considered to be proved. This simple method of Wu is not complete (in algebraic sense) and sometimes the result of applying method is indecisive, i.e. the theorem can't be proved nor disproved. A more complex and complete version of the method uses ascending chains which are considered in the Ritt-Wu principle.

Gröbner basis method G is gröbner basis for ideal $I = \langle f_1, \ldots, f_k \rangle$ if and only if multivariate division any polynomial belonging to ideal I with G gives 0. This means that proving given conjecture consists of two steps. The first is determining the Gröbner basis for polynomials describing relation between objects, $f_1, \ldots, f_k$. The second step is multivariate division of polynomial describing relation to be proved, g_j with G. If the result of division is 0 then statement is proved, otherwise is disproved.

For calculating Gröbner basis is used Buchberger's algorithm witch transforms a given set of polynomials into a Gröbner basis with respect to some monomial order using B-reduction.

3.3. Implementations of algebraic methods

For testing results of applied agrebaization we used two systems – GeoProver that uses simple Wu's method and Mathematica with implemented Gröbner basis method. Shortly, both systems are going to be presented here.

GeoProver GeoProver is open source software implemented in Java providing support for algebraic theorem proving using Wu's method. GeoProver consists of two main modules: one provides support for algebraic methods and the other is a set of APIs from different geometric applications and formats to the prover.

GeoProver produces detailed reports with steps that are taken in proving process: transformation of input geometric problem to algebraic form, invoking a specified algebraic-based theorem prover, and presenting the result with time and space spent to prove the theorem and with a list of NDG conditions obtained during proving process, transformed to a readable, geometry form.

One of the main purposes of this Java implementation is integration with various dynamic geometry tools (including GeoGebra) that currently dont have support for proving geometry theorems. The architecture of GeoProver enables easy integration with other systems for interactive geometry and can be simply modified to accept various input formats for conjectures.

Gröbner basis in Mathematica ??

4. Alegebrization of geometric relations in solid geometry

To apply algebraic methods we need to be able to represent various geometric relations between solid geometry objects using polynomial equations over their coordinates. In this section we are going to give examples how this could be done, for the most common relations (a richer set of relations can be represented using similar techniques, and a detailed description is given in an online appendix).

There are different approaches for encoding relations in terms of polynomial equations and the first question is what objects of 3d geometry are basic. In the first approach, all objects are defined using points (e.g., lines are defined by two different points, and planes are defined as three different, non-colinear points). The only variables used in polynomials are coordinates of the points. In the other approach, all types of objects are represented using their own coordinates (e.g., a line is defined by the coordinates of its one point, and the coordinates of its direction vector, and a plane

is defined by the coefficients of the plane equation – the coordinates of its normal vector, and its displacement wrt. the origin). Polynomials include all those coordinate variables. We shall show how to encode relations using both of these approaches, and shall compare their efficiency.

4.1. Basic notions used for constructing polynomials

Most relations are expressed using the same set of notions which we introduce here.

Each kind of object is represented by some tuple of parameters (we shall see that these could have either symbolic, or numeric values).

Points will have three parameters, denoted by $([\_]^x, [\_]^y, [\_]^z)$ representing their coordinates. Every point is given either by its symbolic or numeric coordinates. To each new point introduced point fresh symbolic coordinates are assigned.

Lines are represented differently, depending on the used approach. In the first approach a line is given by two different given points, and a six-tuple of their coordinates. The first of the point of the line p will be denoted by p_A and the second point by p_B.

In the other approach, a line is given by a given point A and a given vector v. Vector of the line named p will be denoted by $\overrightarrow{p_v}$ and the point of the line p will be denoted by p_A. Therefore, lines in the second approach will also have six parameters denoted by $([\_]^{v_x}, [\_]^{v_y}, [\_]^{v_z}, [\_]^{A_x}, [\_]^{A_y}, [\_]^{A_z})$, which represent the line given by the equation:

$$x = k\cdot[\_]^{v_x}+[\_]^{A_x} \quad y = k\cdot[\_]^{A_y}+[\_]^{A_y} \quad z = k\cdot[\_]^{v_z}+[\_]^{A_z}.$$

In the previous equation, k denotes the line ratio, but this information is not present in the line specification (which is a six-tuple). Line ratio is going to be used in some polynomials that describe constructions involving lines, but it is going to be considered as a fresh symbolic variable.

Planes are also represented differently, depending on the used approach. In the first approach a plane is given by three non-colinear points, and a nine-tuple of their coordinates. The first point of the plane π will be denoted by π_A, the second one by π_B and the third one by π_C.

In the other approach, planes are determined by their normal vector v and an additional parameter d (displacement wrt. the origin). Vector of the plane named π will be denoted with $\overrightarrow{\pi_v}$ and free parameter for the plane will be denoted with π_d. Therefore, planes will have only four parameters, denoted by $([\_]^{v_x}, [\_]^{v_y}, [\_]^{v_z}, [\_]^d)$, which represent the plane given by the following equation:

$$[\_]^{v_x} \cdot x + [\_]^{v_y} \cdot y + [\_]^{v_z} \cdot z + [\_]^d = 0.$$

Vector determined by two points $A = (a^x, a^y, a^z)$ and $B = (b^x, b^y, b^z)$ is $\overrightarrow{AB} = (b^x - a^x, b^y - a^y, b^z - a^z)$. The standard notions of scalar product, cross product and triple product can be applied to vectors. Scalar product of vectors $v = (v^x, v^y, v^z)$ and $u = (u^x, u^y, u^z)$ is $v \cdot u = v^x \cdot u^x + v^y \cdot u^y + v^z \cdot u^z$, their vector product is determined by the matrix:

$$v \times u = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ v^x & v^y & v^z \\ u^x & u^y & u^z \end{vmatrix},$$

and their triple product with the vector $w = (w^x, w^y, w^z)$ is equal to $v \cdot (u \times w)$, and is determined by the matrix:

$$\begin{vmatrix} v^x & v^y & v^z \\ u^x & u^y & u^z \\ w^x & w^y & w^z \end{vmatrix}.$$

4.2. Representing relations between geometry objects

In this section are given polynomials that arithmetically describe relations over constructed objects (for example, two points coincide, two lines are parallel, two planes are orthogonal). Each kind of relation introduces some polynomial constraints of the coordinates of the objects that are involved, and are expressed differently depending on the chosen approach.

Input parameters for given relation are parameters of all objects involved in it. For example, for the relation `congruent` A B C D inputs are four points, A, B, C, and D, e.g their symbolic coordinates: (a^x, a^y, a^z), (b^x, b^y, b^z), (c^x, c^y, c^z) and (d^x, d^y, d^z).

▷ `congruent` A B C D

Description: Two segments, AB and CD are congruent.

Polynomials:

$\overrightarrow{AB} \cdot \overrightarrow{AB} = \overrightarrow{CD} \cdot \overrightarrow{CD}$.

Note that this gives the polynomial equation $poly = 0$, where

$$poly = (a^x - b^x)^2 + (a^y - b^y)^2 + (a^z - b^z)^2 - (c^x - d^x)^2 - (c^y - d^y)^2 - (c^z - d^z)^2$$

Explanation: Squares of the distance between A and B must be equal to the square of the distance between C and D.

▷ `segments_in_ratio` $A\ B\ C\ D\ m\ n$

Description: The lengths of segments AB and CD are in the given ratio $\frac{m}{n}$ i.e., that $\frac{|AB|}{|CD|} = \frac{m}{n}$.
Polynomials:
$n^2 \cdot \overrightarrow{AB} \cdot \overrightarrow{AB} = m^2 \cdot \overrightarrow{CD} \cdot \overrightarrow{CD}$.

Explanation: The squares of distances between A and B and between C and D must be in the ratio $\frac{m^2}{n^2}$. Note that this reduces to congruence when $m = n$.

▷ `is_midpoint` $M\ A\ B$

Description: Checks weather point M is a midpoint of the segment determined with points A and B.
Polynomials: $\overrightarrow{MA} = \overrightarrow{MB}$. Note that this yields three different polynomials $poly_1 = 0$, $poly_2 = 0$, $poly_3 = 0$:

$$poly_1 = 2m^x - a^x - b^x$$
$$poly_2 = 2m^y - a^y - b^y$$
$$poly_3 = 2m^z - a^z - b^z$$

▷ `point_segment_ratio` $M\ A\ B\ p\ q$

Description: Checks weather the point M divides segment determined with points A and B in ratio determined with p and q, e.g. $\frac{|MA|}{|MB|} = \frac{p}{q}$.
Polynomials: three polynomials derived from $q \cdot \overrightarrow{MA} = p \cdot \overrightarrow{MB}$.
Explanation: Note that `is_midpoint` $M\ A\ B$ can be also written using this rule as `point_segment_ratio` $M\ A\ B\ 1\ 1$.

▷ `equal_points` $A\ B$

Description: Checks weather two points A and B have the same coordinates.
Polynomials: three polynomials from the expression $\overrightarrow{AB} = 0$.

▷ `orthogonal_4points` $A\ B\ C\ D$

Description: Checks weather line determined with points A and B is orthogonal on the line determined with points C and D.
Polynomials: $\overrightarrow{AB} \cdot \overrightarrow{CD} = 0$

▷ `orthogonal_lines` $p\ q$

Description: Two lines, p and q are orthogonal.
Polynomials: If the first approach is used, then the lines are given by the points (p_A, p_B) and (q_A, q_B), so this reduces to the previous case and the polynomial is $\overrightarrow{p_A p_B} \cdot \overrightarrow{q_A q_B} = 0$. If the second approach is used, the direction vectors of the lines are given as the input, and the polynomial is $\overrightarrow{p_v} \cdot \overrightarrow{q_v} = 0$.

▷ `incident` $A\ p$

Description: Checks weather the point A belongs to the line p.
Polynomials: If the first approach is used, then the three polynomials are derived from $\overrightarrow{p_A p_B} \times \overrightarrow{Ap_A} = 0$. If the second approach is used, then the three polynomials are derived from $\overrightarrow{p_v} \times \overrightarrow{Ap_A} = 0$.

▷ `parallel_lines` $p\ q$

Description: Checks weather two lines, p and q are parallel.
Polynomials: The three polynomials are derived from $\overrightarrow{p_A p_B} \times \overrightarrow{q_A q_B}$ or $\overrightarrow{p_v} \times \overrightarrow{q_v}$ depending on the approach.

▷ `parallel_planes` $\alpha\ \beta$

Description: Checks weather two planes α and β are parallel.
Polynomials: If the second approach is used, the three polynomials are derived from $\overrightarrow{\alpha_v} \times \overrightarrow{\beta_v} = 0$.

$$\overrightarrow{\beta_A \beta_B} \cdot \overrightarrow{\alpha_A \alpha_C} \times \overrightarrow{\alpha_B \alpha_A} = 0$$
$$\overrightarrow{\beta_A \beta_C} \cdot \overrightarrow{\alpha_A \alpha_C} \times \overrightarrow{\alpha_A \alpha_B} = 0$$

$$(\overrightarrow{\beta_A \beta_C} \times \overrightarrow{\beta_A \beta_B}) \times (\overrightarrow{\alpha_A \alpha_C} \times \overrightarrow{\alpha_A \alpha_B}) = 0$$

▷ `orthogonal_planes` $\alpha\ \beta$

Description: Checks weather two planes, α and β are orthogonal.
Polynomials: If the first approach is used .... If the second approach is used, the polynomial $\overrightarrow{\alpha_v} \cdot \overrightarrow{\beta_v} = 0$.

▷ `point_in_plane` $A\ \pi$

Description: Checks weather the point A belongs to the plane π.
Polynomials: If the first approach is used, then the polynomial

$$\overrightarrow{\pi_A A} \cdot (\overrightarrow{\pi_A \pi_B} \times \overrightarrow{\pi_A \pi_C}) = 0.$$

If the second approach is used, then the polynomial $\overrightarrow{\pi_v} \cdot \overrightarrow{A} + \pi^d = 0$.

▷ `parallel_line_plane` p α

Description: Checks weather the line p and plane α are parallel.

Polynomials: If the first approach is used ... If the second approach is used, then polynomial $\overrightarrow{p_v} \cdot \overrightarrow{\alpha_v} = 0$.

Note: This statement also holds if line belongs to the plane.

▷ `orthogonal_line_plane` p α

Description: Checks weather the line p and plane α are orthogonal.

Polynomials: If the first approach is used ... If the second approach is used the three polynomials $\overrightarrow{p_v} \times \overrightarrow{\alpha_v} = 0$.

5. Simplification of polynomials (needed for Wu's method)

The first problem when applying GeoProver onto generated sets of polynomials were reaching time and space limit. The reason for this is large number of variables and polynomials that were created. Furthermore, polynomials in solid geometry are more complex than corresponding polynomials in plane geometry. Thus, there was a need to simplify our system.

We used approach proposed by Harrison, without lost of generality reasoning — "wlog" [**?**] by making a convenient choice of coordinate system since different choice of coordinate system could make algebraic calculations much less robust.

For three independently given points A, B and C, it is possible to choose their coordinated in such a manner that $A(0,0,0)$ is in the origin, $B(0,0,b^z)$ is on the z-axis and $C(0,c^y,c^z)$ lies in plane yOz. With this choice of coordinates, the number of variables is reduced by six and the corresponding zeroes simplify polynomials. This approach is commonly used in algebraic methods. It does not affect the generality of statement and justification for its usage comes from the fact that rotations and translations can be used to transform points into its "canonical" position. Translation and rotation are isometries which means that they preserve distance, and then also the other geometric relations such as incidence, orthogonality, size of an angle and so on.

We extended this approach and applied similar reasoning on lines and plains. In our list of constructions there exist one construction for arbitrary given line, `line` l. With this construction rule, six variables are generated, vector of the line $(l^{v_1}, l^{v_2}, l^{v_3})$ and the coordinates some point on the line (p^x, p^y, p^z). However, it is possible to choose a more convenient coordinates, for vector $(0,0,l^{v_3})$ and for point $(0,0,0)$, which means that as arbitrary line is chosen z-axis. Similarly goes for plains, and for arbitrary given plane, `plane` π, it is possible to choose coordinates $\pi(0,0,p^{v_3},0)$, which means that arbitrary chosen plain is plain xOy.

Without applying this method, even the simplest statements could not be proved. Choosing convenient coordinates greatly improved simplicity of the system of polynomials and, by doing so, made the program more efficient. However, some polynomials become redundant, as they become 0 and we just deleted them from the system. Also, some polynomials become a polynomial of one variable on some degree and they were also deleted from the system and the value of the corresponding variable was set to zero. Although, it was not necessary to delete these polynomials, the system was simplified a bit more when they were deleted.

To simplify constructions even further, we choose appropriate coordinates for figures as we explained in 5.1.

Furthermore, polynomials whose structure is $x_i - x_j$ or $x_i + x_j$ (assume $j < i$) were also deleted from the system, each appearance of x_i was replaced with x_j, e.g. with $-x_j$, and x_i was deleted from the set of variables. This simplification step was not very crucial, and most of the tested statements could be proved without adding this simplification. However, for some complex statements, it was very important to reduce the size of sets of polynomials (otherwise, space or time limit is reached) and this is a step into that direction.

5.1. Construction rules

Each construction starts from some given (free) objects and proceeds by introducing new (dependent) objects. We will allow introducing only free points (and all lines, planes, and solids are going to be constructed started from a given set of free points).

Free points are determined by their fresh, symbolic coordinates, and are not restricted by any polynomials. Constructed objects are determined again by introducing their fresh symbolic param-

eters, but also polynomials that describe connections between those parameters and the symbolic parameters of the objects that were previously constructed. We shall denote the parameters of free points by upper-case letters, and the parameters of dependent points by lower-case letters. For example, the free point A will have symbolic coordinates (A^x, A^y, A^z), and a constructed line l will have symbolic parameters $(l^{v_x}, l^{v_y}, l^{v_z}, l^{p_x}, l^{p_y}, l^{p_z})$.

Constructions of points Next we shall describe some example construction steps and their corresponding polynomials. Each construction step yields a new point. Its name will be given by the first argument of the construction step (e.g., `midpoint` M A B constructs the point M using given points A B). In each such construction step, it is assumed that the symbolic parameters of given objects are already known, and the symbolic coordinates of the resulting point are introduced (as fresh variables).

▷ `point` A

Description: This step introduces a free point A.
Input: none
Objects and parameters: A point A with the three fresh parameters (A_x, A_y, A_z)
Polynomials: no polynomials are generated.

▷ `point_ratio` M A B p q

Description: Construction of a point that divides a segment in a given ratio, i.e., for two given points A and B (these could be either free or dependent points), and two integer numbers p and q, this step constructs the point M so that $\frac{|AM|}{|BM|} = \frac{p}{q}$ holds.
Input: Two given points A and B with symbolic coordinates (a^x, a^y, a^z) and (b^x, b^y, b^z).
Objects and parameters: The point M with three fresh symbolic coordinates (m^x, m^y, m^z).
Polynomials:

$$poly_1 = p \cdot m^x + q \cdot m^x - p \cdot b^x - q \cdot a^x$$
$$poly_2 = p \cdot m^y + q \cdot m^y - p \cdot b^y - q \cdot a^y$$
$$poly_3 = p \cdot m^z + q \cdot m^z - p \cdot b^z - q \cdot a^z$$

Explanation: `point_ratio` M A B p q = `point_segment_ratio` M A B p q
Note: Note that `midpoint` is a special case of this construction, e.g. it is possible to determine midpoint in the following way `point_ratio` M A B 1 1. However, we also support the `midpoint` M A B construction.

mislim da nema potrebe svuda stavljati "Input" i "Output". Mozda u prvom primeru objasniti, ali deluje redudatno

▷ `intersection_lines` A l_1 l_2

Description: This construction introduces the point A that is the intersection of the two given lines l_1 and l_2.
Input: The two given lines l_1 and l_2 with parameters $l_1(l_1^{v_x}, l_1^{v_y}, l_1^{v_z}, l_1^{p_x}, l_1^{p_y}, l_1^{p_z})$ and $l_2(l_2^{v_x}, l_2^{v_y}, l_2^{v_z}, l_2^{p_x} l_2^{p_y}, l_2^{p_z})$
Objects and parameters: The point A with three fresh symbolic coordinates (a^x, a^y, a^z).
Polynomials: We generate polynomials using rule: `intersection_lines` A l_1 l_2 = `incident` A l_1 and `incident` A l_2

Polynomials for Wu's method:

$$poly_1 = a^x - k_1 \cdot l_1^{v_x} - l_1^{p_x}$$
$$poly_2 = a^y - k_1 \cdot l_1^{v_y} - l_1^{p_y}$$
$$poly_3 = a^z - k_1 \cdot l_1^{v_z} - l_1^{p_z}$$
$$poly_4 = a^x - k_2 \cdot l_2^{v_x} - l_2^{p_x}$$
$$poly_5 = a^y - k_2 \cdot l_2^{v_y} - l_2^{p_y}$$
$$poly_6 = a^z - k_2 \cdot l_2^{v_z} - l_2^{p_z}$$

Explanation: Since the point a should belong to both lines l_1 (given by the point $\overrightarrow{l_1^p}$ and the vector $\overrightarrow{l_1^v}$) and l_2 (given by the point $\overrightarrow{l_2^p}$ and the vector $\overrightarrow{l_2^v}$) it must satisfy their parametric equations, i.e., it must hold that $\overrightarrow{A} = \overrightarrow{l_1^p} + k_1 \cdot \overrightarrow{l_1^v}$ and $\overrightarrow{A} = \overrightarrow{l_2^p} + k_2 \cdot \overrightarrow{l_2^v}$.
This is the most interesting case of point construction. In tree–dimensional space not all lines have intersection, some lines can be parallel, and others can be skew lines. Note that the polynomials use two unknown variables k_1 and k_2 that respectively represent line ratios for lines l_1 and l_2. Therefore, the total number of unknown variables is five, but there are six polynomials that this construction generates. If all six polynomials are included in the construction set, then it will not be possible to make triangular system since there will be more polynomials than unknown variables. Five polynomials can be used to determine the solution of the system of equations and the remaining polynomial should be used to justify the solution. Any of the polynomials could be excluded and in our solution we always exclude p_6. If all five chosen polynomials have zero value for the calculated coordinates, then also the remaining polynomial must be zero (or otherwise, the lines will not always intersect). If the excluded polynomial is not considered, that could lead into wrong constructions and conclusions since now it is possible to (falsely) determine the intersection

Govori se o postupku resavanja o kom nije jos bilo reci.

point of parallel or skew lines. So, instead of just deleting this last polynomial, is should be put it in the conclusion set.

▷ `intersection_4points` $M\ A\ B\ C\ D$

Description: For four given point A, B, C and D determine the point M which is intersection of line determined with points A and B and line determined with points C and D. This is very similar to the construction we described above.
Input: Four points A, B, C and D with symbolic coordinates (a^x, a^y, a^z), (b^x, b^y, b^z), (c^x, c^y, c^z) and (d^x, d^y, d^z).
Objects and parameters: The point M with three fresh symbolic coordinates (m^x, m^y, m^z).

Polynomials:
$$poly_1 = m^x - k^x \cdot b^x + k_1 \cdot a^x - a^x$$
$$poly_2 = m^y - k^x \cdot b^y + k_1 \cdot a^y - a^y$$
$$poly_3 = m^z - k^x \cdot b^z + k_1 \cdot a^z - a^z$$
$$poly_4 = m^x - k^y \cdot d^x + k_2 \cdot c^x - c^x$$
$$poly_5 = m^y - k^y \cdot d^y + k_2 \cdot c^y - c^y$$
$$poly_6 = m^z - k^y \cdot d^z + k_2 \cdot c^z - c^z$$

Explanation: The polynomials were generated using the following line equations:
$x = k_1 \cdot (b^x - a^x) + a^x \quad y = k_1 \cdot (b^y - a^y) + a^y \quad z = k_1 \cdot (b^z - a^z) + a^z$ — line determined with points A and B
$x = k_2 \cdot (d^x - c^x) + c^x \quad y = k_2 \cdot (d^y - c^y) + c^y \quad z = k_2 \cdot (d^z - c^z) + c^z$ — line determined with points C and D.
This construction could be performed by explicitly constructing lines AB and CD and then constructing their intersection using the previous rules. However, it turns out that this would introduce many new symbolic variables (parameters of the two lines), the system would become more complex, and the algebraic solver would have to work with a much more difficult task. Therefore, it is much more efficient to have this construction as an elementary step, especially, having in mind that this step is used very often in classic geometric problems.
Note that again there are six polynomials and five unknown variables. This problem is resolved in the same manner as above by putting polynomial $poly_6$ in the conclusion set. The reason is the same as in the previous construction rule and again it is possible not to have intersection since lines can be parallel or skew.

Constructions of lines Although the construction of a free line could be easily introduced, for simplicity, we have chosen to support only free points (and a free line can be constructed as a line trough two free points).

▷ `line_points` $l\ A\ B$

Description For two given points A and B determines the line l that goes through these points.
Input: Two points A and B with symbolic coordinates (a^x, a^y, a^z) and (b^x, b^y, b^z).
Objects and parameters: The line l with parameters $(l^{v_x}, l^{v_y}, l^{v_z}, a^x, a^y, a^z)$ (new, fresh parameters $(l^{v_x}, l^{v_y}, l^{v_z})$ are parameters of line vector).

Polynomials:
$$poly_1 = l^{v_x} - a^x + b^x$$
$$poly_2 = l^{v_y} - a^y + b^y$$
$$poly_3 = l^{v_z} - a^z + b^z$$
$$\overrightarrow{l} = \overrightarrow{AB}$$
`line_points` $l\ A\ B =$ `incident` $A\ l$ and `inciden`

Explanation: Each line is determined by a set of 6 parameters, but note that in this construction we have chosen to introduce only three. This is due to the fact that the parameters for the point on the line can be exactly the coordinates of the point A (or B), that are already introduced symbolic variables, so there is no need to introduce new variables and polynomials (if the parameters l^{p_x}, l^{p_y} and l^{p_z} were introduced, they would need to satisfy the trivial polynomials $l^{p_x} - a_x$, $l^{p_y} - a_y$ and $l^{p_z} - a_z$). The parameters $(l^{v_x}, l^{v_y}, l^{v_z})$ that correspond to the vector of the line must satisfy $\overrightarrow{v} = \overrightarrow{BA}$, and the polynomials are generated from this equation (when written in symbolic coordinates).

▷ `line_orth_plane` $l\ \pi\ A$

Description: For a given plane π and a point A determines line l witch is orhogonal on the given plane π and incides with the point A.
Input: Plane π with symbolic parameters $(\pi^{v_x}, \pi^{v_y}, \pi^{v_z}, \pi^d)$ and point A with symbolic coordinates (a^x, a^y, a^z).
Objects and parameters: Line l with parameters $(\pi^{v_x}, \pi^{v_y}, \pi^{v_z}, \pi^d, a^x, a^y, a^z)$ (no new parameters are generated).
Polynomials: No polynomials are generated.
`line_orth_plane` $l\ \pi\ A =$ `incident` $A\ l$ and `parallel_line_plane` $l\ \pi$

Explanation: Both the parameters for the point on the line and the vector of the line can be represented by the already introduced symbolic coordinates — the point can be the point A and the line vector could be equal to the normal vector of the plane. Therefore, it was not necessary to add neither new symbolic parameters, nor the new polynomials.

The construction of the line trough a given point that is parallel to the given line is performed similarly (the parameters of the point and the parameters of the vector of the given line can be reused, and no new parameters and polynomials need to be introduced).

Constructions of the plane Again, for simplicity, we have chosen to support only free points (and a free plane can be constructed as a plane trough three free points).

▷ `plane_points` π A B C

Description: For three given points A, B and C determines the plane π containing all of them.

Input: Tree point A, B and C with symbolic coordinates (a^x, a^y, a^z), (b^x, b^y, b^z) and (c^x, c^y, c^z).

Objects and parameters: Plane π with fresh parameters $(\pi^{v_x}, \pi^{v_y}, \pi^{v_z}, \pi^d)$.

ovo nije sasvim tacno, jer ovde imamo tri jednacine, a 4 nepoznate

Polynomials:
$$poly_1 = \pi^{v_x} - b^y \cdot c^z + b^y \cdot a^z + a^y \cdot c^z + b^z \cdot c^y - b^z \cdot a^y - a^z \cdot c^y$$
$$poly_2 = \pi^{v_y} - b^x \cdot c^z - b^x \cdot a^z - a^x \cdot c^z - b^z \cdot c^x + b^z \cdot a^x + a^z \cdot c^x$$
$$poly_3 = \pi^{v_z} - b^x \cdot c^y + b^x \cdot a^y + a^x \cdot c^y + b^y \cdot c^x - b^y \cdot a^x - a^y \cdot c^x$$
$$poly_4 = \pi^d + \pi^{v_x} \cdot a^x + \pi^{v_y} \cdot a^y + \pi^{v_z} \cdot a^z$$
$$\texttt{plane\_points}\ \pi\ A\ B\ C = \texttt{point\_in\_plane}\ A\ \pi \text{ and } \texttt{point\_in\_plane}\ B\ \pi \text{ and } \texttt{point\_in\_plane}\ C\ \pi$$

Explanation: The normal vector of the line can be obtained as a vector product of the vectors $\overrightarrow{AB}$ and $\overrightarrow{AC}$. Since $(b^x - a^x, b^y - a^y, b^z - a^z)$ and $(c^x - a^x, c^y - a^y, c^z - a^z)$ are vectors of two lines determined with points A and B, and points C and D, the first three polynomials are derived using the formula:

$$(\pi^{v_x}, \pi^{v_y}, \pi^{v_z}) = \begin{matrix} i & j & k \\ b^x - a^x & b^y - a^y & b^z - a^z \\ c^x - a^x & c^y - a^y & c^z - a^z \end{matrix}$$

The last parameter of the plane π^d is obtained from the condition that the point A is in the plane π, and satisfies its equation. Note that this equation uses the symbolic parameters π^{v_x}, π^{v_y} and π^{v_z} that are introduced in this construction step.

Very similar methods can be used to construct the plane that contains the given line and is orthogonal or is parallel to the given plane.

Solid construction Many exercises in solid geometry have objects and usually start with a sentence of the form *"In a given cube..."*, *"In a given pyramid..."*, *"In a given prism..."* and so on. The first approach was to construct such solids using construction rules presented above. This means that there were many rules (around 20) constructing only one cube and thus many new variables and polynomials which made polynomial systems very complex. Also, some important construction steps could be forgotten. Therefore, it is very natural to introduce solid constructions as elementary construction steps (instead of ten or twenty rules that construct their points, lines and planes). Such construction steps automatically introduce only the points of the solid, whereas lines (edges) and planes (faces) can be introduced by the previous construction rules only when necessary.

Da li uvesti izometrijske transformacije?

We decide to support only the construction of solids placed in some *canonical* positions — for example, when constructing a cube, one vertex is placed in the origin, and the other three are placed on the axes (x-axis, y-axis and z-axis). However, this approach has a drawback. For example, it is not possible to construct more than one different cube using the elementary cube construction step. The points of other cubes that are not in the canonical position can be constructed using point construction steps or by applying isometric transformations to the canonical cube. On the other hand, in most geometric problems encountered in education there is usually only one free solid, and without loss of generality it can be assumed that it is in the canonical position. When other solids are introduced in the problem text, they are usually dependent, so their points (vertices) have to be constructed. Therefore, supporting only canonical solid constructions was not the obstacle for the problems encountered in most exercises.

▷ `make_cube`

Description: Construct the cube in the canonical position, with the edge length equal to 1.

Input: none
Objects and parameters: Points $A(0,0,0)$, $B(1,0,0)$, $C(1,1,0)$, $D(0,1,0)$, $A_1(0,0,1)$, $B_1(1,0,1)$, $C_1(1,1,1)$ and $D(0,1,1)$.
Polynomials: No polynomials are generated.
Explanation: Since the cube is in the canonical position, no symbolic variables are introduced.

▷ `make_tetrahedron`
Description: Construct the tetrahedron in the canonical position.
Input: none
Objects and parameters: The vertices of the tetrahedron have the coordinates $A(0,0,0)$, $B(1,0,0)$, $C(c^x, c^y, 0)$ and $D(c^x, d^y, d^z)$, with the four fresh parameters.

Polynomials: $poly_1 = 2 \cdot c^x - 1$
$poly_2 = 2 \cdot c^{y\,2} - 3$
$poly_3 = 3 \cdot d^y - c^y$
$poly_4 = 3 \cdot d^{z\,2} - 2$

Explanation: $c^x = \frac{1}{2}$, $c^y = \frac{\sqrt{3}}{2}$, $d^y = \frac{\sqrt{3}}{6} = \frac{c^y}{3}$, $d^z = \frac{\sqrt{2}}{\sqrt{3}}$. Note that all our objects always have either symbolic or integer parameters and polynomials must always have integer coefficients, so irrational values (and even fractions) must be introduced using polynomials.

▷ `make_pyramid_4side`
Description: Construct a regular pyramid in the canonical position – the base is a unit square in the xOy plane, its lateral edges are equal in length, and the height is not constrained.
Input: none
Objects and parameters: Points $A(0,0,0)$, $B(1,0,0)$, $C(1,1,0)$, $D(0,1,0)$ and $S(s^x, s^y, s^z)$, for three fresh parameters s^x, s^y and s^z.

Polynomials: $poly_1 = 2 \cdot s^x - 1$
$poly_2 = 2 \cdot s^y - 1$

Explanation: The projection of the apes is $(s^x, s^y, 0)$ and it lies in the middle of the square, so $s^x = s^y = \frac{1}{2}$. Note that s^z is not constrained.

▷ `make_square`
Description: Construct a square in the canonical position – the base is a unit square in the xOy plane, its lateral edges are equal in length, and the height is not constrained.
Input: none
Objects and parameters: Points $A(0,0,0)$, $B(1,0,0)$, $C(1,1,0)$, $D(0,1,0)$.
Polynomials: no new polynomials are created.

▷ `make_point_on_line` A l
Description: Construct a point that belongs to the given line.
Input: Line l with symbolic parameters with parameters $l(l^{v_x}, l^{v_y}, l^{v_z}, l^{p_x}, l^{p_y}, l^{p_z})$.
Objects and parameters: Point $A(a^x, a^y, a^z)$, for three fresh parameters a^x, a^y and a^z.

Polynomials: $poly_1 = l^{p_x} \cdot l^{v_y} - a^x \cdot l^{v_y} - l^{p_y} \cdot l^{v_x} + a^y \cdot l^{v_x}$
$poly_2 = l^{p_x} \cdot l^{v_z} - a^x \cdot l^{v_z} - l^{p_z} \cdot l^{v_x} + a^z \cdot l^{v_x}$
$poly_3 = l^{p_y} \cdot l^{v_z} - a^y \cdot l^{v_z} - l^{p_z} \cdot l^{v_y} + a^z \cdot l^{v_y}$
`make_point_on_line` A l = `incident` A l

▷ `make_point_in_plane` A π
Description: Construct a point that belongs to the given plane.
Input: Plane π with fresh parameters $(\pi^{v_x}, \pi^{v_y}, \pi^{v_z}, \pi^d)$.
Objects and parameters: Point $A(a^x, a^y, a^z)$, for three fresh parameters a^x, a^y and a^z.

Polynomials: $poly = \overrightarrow{\pi} \circ \overrightarrow{A} + \pi^d$
`make_point_in_plane` A π = `point_in_plane` A π

▷ `translate_z` A O q
Description: Construct a point given by translating another point by some parameter along z-axis.
Input: Point O with symbolic coordinates (o^x, o^y, o^z) and parameter q.
Objects and parameters: Point $A(o^x, o^y, a^z)$, for one fresh parameter, a^z.

Polynomials: $poly = a^z - q - o^z$
$\overrightarrow{AO} = (0, 0, q)$

▷ `equilateral_triangle` A B C
Description: Construct a equilateral triangle in the canonical position – it is in xOy plane, one point is in origin, and another point is on x-axis.
Input: none
Objects and parameters: Points $A(0,0,0)$, $B(1,0,0)$, $C(c^x, c^y, 0)$, for two fresh parameters c^x and c^y.

Polynomials: $poly_1 = 2 \cdot c^x - 1$
$poly_2 = 4 \cdot c^y - 3$

▷ `regular_hexagon` A_1 A_2 A_3 A_4 A_5 A_6
Description: Construct a regular hexagon in the canonical position – it is in xOy plane,

one point is in origin, and another point is on x-axis.
Input: none
Objects and parameters: Points $A_1(0,0,0)$, $A_2(1,0,0)$, $A_3(a_3^x, a_3^y, 0)$, $A_4(1, a_4^y, 0)$, $A_5(0, a_4^y, 0)$ and $A_6(a_6^x, a_3^y, 0)$, for four fresh parameters a_3^x, a_3^y, a_4^y, and a_6^x.

Polynomials: $poly_1 = 2 \cdot a_3^x - 3$
$poly_2 = 4(a_3^y)^2 - 3$
$poly_3 = a_4^y - 3$
$poly_4 = 2a_6^x - 1$

6. Conclusion and future work