The Euclid Abstract Machine

JERZY MYCKA[1][*], JOSÉ FÉLIX COSTA[2], FRANCISCO COELHO[3]

[1] *Institute of Mathematics,*
University of Maria Curie-Skłodowska
Lublin, Poland
§

[2] *Department of Mathematics,*
I.S.T., Universidade Técnica de Lisboa
and CMAF – Centro de Matemática e Aplicações Fundamentais,
Lisboa, Portugal
||

[3] *Department of Mathematics,*
Universidade de Évora, Portugal
#

He took the golden Compasses, prepar'd
In Gods Eternal store, to circumscribe
This Universe, and all created things:
One foot he center'd, and the other turn'd
Round through the vast profunditie obscure,
And said, thus farr extend, thus farr thy bounds,
This be thy just Circumference, O World.

Milton, Paradise Lost
Most famous painting by William Blake

§ email: `Jerzy.Mycka@umcs.lublin.pl`
|| email: `fgc@math.ist.utl.pt`
# email: `fc@uevora.pt`
⋆ Corresponding author

Abstract. Concrete non-computable functions are usually related to the halting function. Is it possible to present examples of non-computability, which are unrelated to the halting problem and its derivatives? We built an abstract machine based on the historic concept of compass and ruler constructions (a compass construction would suffice) which reveals the existence of non-computable functions not related with the halting problem. These natural, and the same time, non-computable functions can help to understand the nature of the uncomputable and the purpose, the goal, and the meaning of computing beyond Turing.

Key words: unconventional computing, Euclid machine, undecidable problems.

1 INTRODUCTION

The three most famous problems of ancient Greeks were to *square* a circle (i.e., to find a square with the same area — problem also known as the quadrature of a circle), to trisect an angle, and to duplicate a cube, by the use of only an unmarked ruler and a compass. Greek philosophers, in particular Plato, viewed the straight line and the circle as the basic and perfect curves that be sufficient to accomplish the geometric constructions, namely to fully explain the motions of the heavenly bodies, like the sun, the moon, the fixed stars and the erratic planets which trajectories were thought later as compositions of circles and uniform movements.

The Greeks were not able to solve these three problems (see [2]). Hippocrates of Chios (about 460–380 B.C.) became famous for the major contributions on (a) the quadrature of lunes that can be seen as an attempt to square the circle and (b) the solution of the duplication of the cube. Menaechmus (about 350 B.C., tutor of Alexander the Great) became famous for his attempt to duplicate the cube.*

The ancient Greeks were interested in the construction of different angles. Starting with a few fundamental angles such as angles of 60° and 108° (the angle of a regular pentagon), in a similar way to modern trigonometry, they could construct other specific angles by (a) adding two given angles, (b) subtracting one given angle from another, and (c) bisecting a given angle. Attempts to solve the trisection problem are due to Archimedes (287–212 B.C.),

* Legend says that in response to Alexander's request for a short account of geometry, Menaechmus replied *O King, for traveling over the country there are royal roads and roads for common citizens; but in geometry is one road for all.*

Nicomedes (about 240 B.C.), and Hippias (about 420 B.C.). Sliding linkage based on Archimedes trisection process, although well known is not reducible to ruler and compass and the other attempts failed also to meet Plato's program. The most well know attempt to square the circle is due to Hippias.

The duplication and trisection problems are closely related: (a) they can both be solved by using conic sections (known to the Greeks), (b) when expressed algebraically, they both lead to cubic equations, and (c) the proofs that they can not be solved with ruler and compass alone make use of the same approach and were given at the same time.

The proofs of the impossibility of duplicating a cube and trisecting an angle rely in the theory of cubic equations, and were developed over many centuries and formally proved by Wantzel ([8]). The proof of impossibility of the quadrature was developed when Lindemann (1852–1939) proved that π is a transcendental number.

Assuming that the general form of a polynomial equation is

$$a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n = 0,$$

the cubic equation takes the form

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 = 0.$$

We have the following classic definition:

Definition 1.1 *By a constructible root of an equation we mean a root that has the following property: if a unit length is given, we can construct with ruler and compass a line segment with the length equal to the root.*

Then we have two main results:

Proposition 1.2 *A cubic equation with integral coefficients that has no rational root has no constructible root.*

Proposition 1.3 *If the coefficients of a polynomial equation are integers, then any rational root of the equation can be written in the form p/q, where p is a factor of a_0 and q is a factor of a_n.*

With these two propositions we can prove the main impossibility results:

Proposition 1.4 *The duplication of the cube and the trisection of the angle can not be accomplished by ruler and compass alone.*

For the proofs and further discussions see [4] and [16]. For more technical proofs see [14], chapter 5 (in chapter 6 the reader can find the proof of Lindemann's theorem on the transcendence of π).

It is interesting to notice that, in 1672, Mohr showed that

- every construction with ruler and compass can be performed with a compass alone.[†]

Of course, there can be considered also constructions using ruler alone. Poncelet suggested that instead of a compass a singled fixed circle, together with its center, should suffice. If the center of the circle is not known, then fewer constructions are possible. Hilbert asked how many circles must be given in order for the center of one of them to be constructed using ruler alone. Cauer, in 1912, showed that for two general circles this is impossible. Grossmann showed that three linearly independent circles suffice for geometric constructions with only a ruler. It is also known that all ruler and compass constructions can be performed using a two-edged ruler, whose edges are either parallel or meet at a point. More ideas can be read in [7].

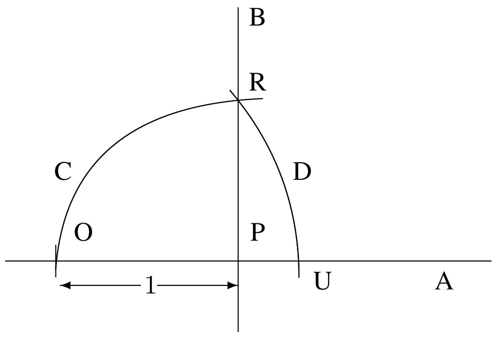

FIGURE 1
Computing $\sqrt{2}$ at point U.

We start our discussions about the purpose of this paper with the illustrations of some Euclidean constructions. We will make use of the Euclidean plane together with unscaled, unmarked Cartesian orthogonal axis (that can

[†] Cf. William Blake's most famous painting and Milton's description of creation.

be constructed in the beginning of each session on Euclid machines). Let us also suppose that we use the x axis to represent numbers as multiples of a suitable unit, by means of an unmarked ruler and a compass accordingly to what was said above.

Example 1.5 *We can obtain irrational numbers along the x axis by simple algebraic construction, such as $\sqrt{2}$. The following is a possible Euclid machine for the purpose. The construction is illustrated in Figure 1.*

01 :: *draw two different points with labels O and P*
02 :: *draw the line trough O and P and give it the label A*
03 :: *draw the perpendicular to A going through P and give it label B*
04 :: *draw the circumference with center P going through O and give it the label C*
05 :: *give the label R to the intersection of B and C over P*
06 :: *draw the circumference with center O going through R and give it the label D*
07 :: *give the label U to the intersection of A and D on the right of O*

If the point O is taken as the origin of the Cartesian coordinate system and the length of $\overline{OP}$ is taken as unit, then the length of $\overline{OU}$ is $\sqrt{2}$. This is the kind of programs we want to write using ruler and compass.

Example 1.6 *As another example, an Euclidean program for the multiplication can be given by means of the well known Thales' theorem. All details are illustrated in Figure 2.*

01 :: *draw two different points with labels O and P*
02 :: *draw the line through O and P and give it the label A*
03 :: *draw the perpendicular to A going through P and give it label B*
04 :: *draw along A any point to the right of O and give it the label S*
05 :: *draw along B any point and give it the label R*
06 :: *draw the line going through O and R and give it label D*
07 :: *draw the perpendicular to A going through S and give it label E*
08 :: *label the intersection point of D and E by T*

If the point O is taken as the origin and the length of $\overline{OP}$ is taken as unit, then the length of $\overline{ST}$ is the length of $\overline{PR}$ times the length of $\overline{OS}$. It means that

$$|\overline{ST}| = |\overline{PR}| \times |\overline{OS}|.$$

In the next section we introduce a suitable language to write programs using the above instruction types. We provide examples and we will discuss

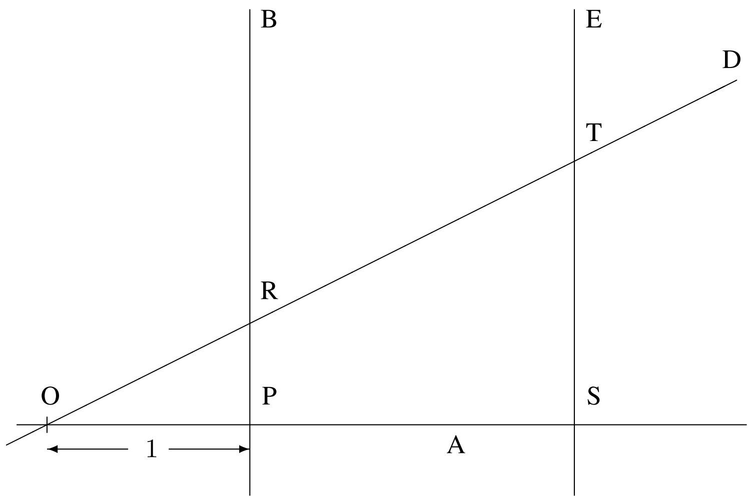

FIGURE 2
Computing the product $|\overline{PR}| \times |\overline{OS}| = |\overline{ST}|$.

two different ways of looking to these programs: as constructs or as proofs. In Section 2 we recall concepts of the unlimited ideal register machine URM and simulate URM programs by Euclidean constructions. In Section 4 we discuss algebraic numbers and its subset of reachable points by unmarked ruler and compass. Finally, in Section 5 we define some undecidable predicates.

The last aim of this paper is to put in correspondence the impossibility of some Euclidean geometric constructions, like (a) the duplication of the cube or (b) the trisection of the angle, with some undecidable problems in classical computability theory.

2 OPERATIONS

Let us imagine a construction of algorithms acting in the framework of Euclid's geometry. We can use an infinite (in reality sufficiently large) sheet of paper, an unmarked ruler and a compass. Now we need to specify the list of possible operations.

- $P(P_1, \ldots, P_n)$ — Draw a finite number of distinct points $P_1, \ldots, P_n$.[‡]
- $C(P, Q)$ — Draw the circle with the center P and going through the point Q.
- $LC(P, Q; A)$ — Give the label A to the circle with the center P and going through the point Q.
- $L(P, Q)$ — Draw the line passing through P and Q.
- $LL(P, Q; A)$ — Give the label A to the line passing through P and Q.
- $LP(O_1, O_2; A, B)$ — Give the label A to the point of the intersection of the objects (lines or circles) O_1 and O_2, in the case of two intersections choose freely the order of labeling by A and B.
- $D(A)$ — Delete the label A.
- $X \in C : n$ — If the point X is in the circle C, then execute the n-th instruction; otherwise go to the next instruction.

Of course, from the first operation we see that points are always labeled, unless labels are ultimately removed through an instruction of the type D. Let us add that each label can be used only in the unique way, i.e., one label can identify exactly one object. This does not mean that some objects can not have two or more labels.

Let us comment some aspects of the above list of operations. The number of starting points is arbitrary and related to the purpose of a program. For example, we can think about a construction of the circle going through the set of given points — then at least three points are needed at the beginning. Of course, any configuration of these points is possible, but some of them can lead to unexpected behavior of the program (as incorrect data supplied for some algorithm working on numbers).

Let us add that in a natural way for the operation $C(P, Q)$ the points P, Q can be identical, then we obtain zero-sized circles (i.e., the same point P). In the case of $L(P, Q)$ we need to have P and Q distinct in order to construct a well-defined line. If some operation is impossible to execute (as $L(P, P)$ or $LP(O_1, O_2; A, B)$ when O_1, O_2 do not intersect or are not distinct) then our program incorrectly halts (like in the case of division by zero for numbers).

[‡] We can think about this operation as a weak version of the axiom of choice — we can always choose finite set of different points from the Euclidean plane.

Some subtlety is connected with the operation D. It is not necessary to use this operation in any program, which is known to have a fixed number of steps. But in some situations we cannot foresee how many steps are needed to complete the program. If we want to have some label chosen for the constructed point we can be forced to rename points during computation. It will be clearly visible, when we describe how to represent variables as points. This necessity leads us to an introduction of the operation D.

The last operation $X \in C : n$ is a conditional form of an instruction. It is needed to build programs with non-fixed numbers of steps during constructions. We interpret $X \in C : n$ as X inside the circle C or on the circumference C. The interpretation is equivalent to the relation $\leq$ on real numbers and is chosen as comfortable in our constructions.

A program is a (implicitly numbered) list of operations of the above types. After the n-th operation the next one (with the number $n + 1$) is executed, unless it is the last operation or it is the test $X \in C : n$.

Example 2.1 *Let us consider the construction of two perpendicular lines. We need to start with two points P, Q, then draw the line through these points. Next we need two circles to construct a perpendicular line.*

Here is the code for the construction of a perpendicular line to PQ through P (see Figure 3):

01 :: $P(P, Q)$
02 :: $L(P, Q)$
03 :: $LL(P, Q; A)$
04 :: $C(P, Q)$
05 :: $LC(P, Q; C)$
06 :: $D(Q)$
07 :: $LP(A, C; Q_1, Q_2)$
08 :: $C(Q_1, Q_2)$
09 :: $LC(Q_1, Q_2; C_1)$
10 :: $C(Q_2, Q_1)$
11 :: $LC(Q_2, Q_1; C_2)$
12 :: $LP(C_1, C_2; S_1, S_2)$
13 :: $L(S_1, S_2)$

By the similarly constructed programs we can give Euclid machines, which draw equilateral triangles or the line that bisects some angle.

Let us consider an analogy which exists between programs of Euclid machines and theorems of Euclidean geometry.

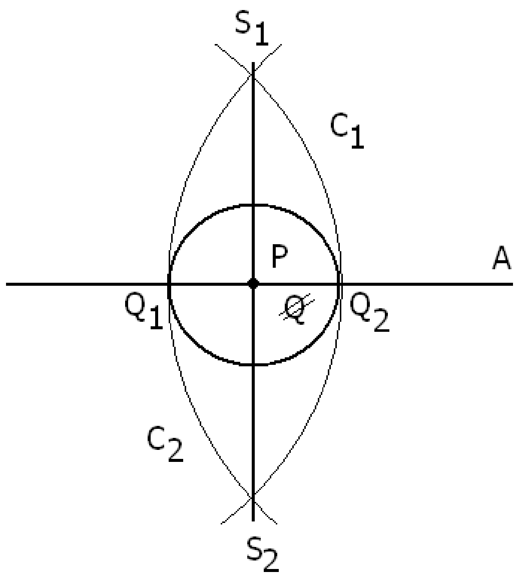

FIGURE 3
Construction of a perpendicular

Example 2.2 *We can start by recalling the following Thales' Theorem:* An inscribed angle in a semicircle is a right angle. *How can this fact be checked by means of Euclid machines? Let us imagine the following construction of Figure 4.*

01 :: *Draw three different, non-collinear points O, A, X*
02 :: *Draw the circle C with the center O and going through A*
03 :: *Draw the line L through O and A*
04 :: *Label the other point of intersection of L and C by B*
05 :: *Draw the line through O and X*
06 :: *Label the points of the intersection of this line and C as P, Q*
07 :: *Draw the line going through A and P*
08 :: *Draw the line going through B and P*
09 :: *Draw the perpendicular L' to the line BP through P*
10 :: *Label the intersection of L' and L as A'.*

Now let us analyze the above part of the program (which can be translated into instructions of the Euclid machine in the obvious manner). For the correct construction only a requirement that the starting points are non-collinear is sufficient.

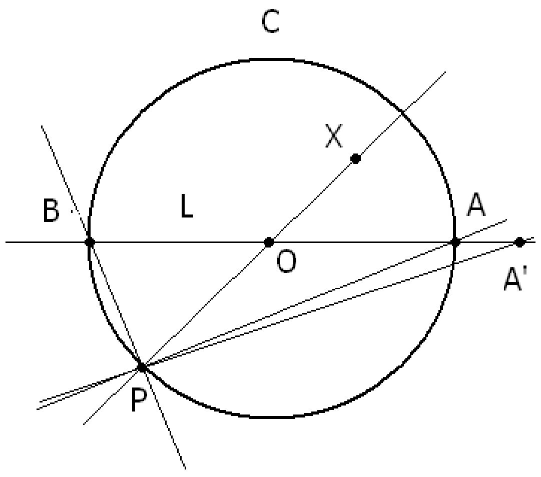

FIGURE 4
Thales theorem

We have constructed the angle $\angle APB$ and, after that, we have added the perpendicular L' to PB in the point P. Thus the fact that $\angle APB$ is a right angle is equivalent to the fact that the points A (the intersection of AP with L) and A' (the intersection of L' and L) are identical. We can use the test operation to check the last statement, let us assume that every point is a circle with a radius of the length 0.

- $A' \in A : n$

We can use this situation to build some kind of output. For example, the program would end its activity if this condition is true; otherwise it would go *into infinite loop. Or we can draw some previously chosen labels for some point (e.g. O): $+$ for the positive test; $-$ for the negative one.*

In the light of the above example we can translate proposed proofs of Euclidean geometry in equivalent programs; the proof is correct if for all initial configurations we obtain the previously chosen special sign (e.g., $+$) of an acceptance.

The reader surely noticed that our geometric constructions with a ruler and a compass are non-deterministic in the sense that they depend on the order of labeling. This fact is not truly relevant for our proofs. Indeed, the non-determinism of labeling can be removed by imposing some conventions on drawing circles.

We assume a Cartesian (unmarked) reference frame (see [14], chapter 5). Such a frame can always be imposed by some arbitrary choice on Euclidean plane. Acceptable convention is: the instruction $LP(O_1, O_2; A, B)$ is executed in the following way:

- If both objects O_1 and O_2 are lines, then always choose label A whenever there is such an intersection.
- If either O_1 or O_2 is a line, then the intersections (if any) are labeled from left to right and from above to bottom along the line.
- The same as in the previous item along the line of intersections if both O_1 or O_2 are circumferences.

3 REGISTER MACHINES

In this section we present the Turing completeness of the above described geometrical machine (i.e., we prove that to each computable — partial recursive — function corresponds at least one Euclid machine). We use for this purpose not the Turing machine but an equivalent formulation, the unlimited register machine [13]. Every unlimited register machine (URM for short) program is a finite sequence of instructions acting on (potentially) infinite number of registers containing natural numbers. The instructions of URM machines programs can be chosen in the following way.

- $Z(n)$ — Put 0 into the n-th register.
- $S(n)$ — Increment (adding 1) the current value of the n-th register.

- $J(n, m, k)$ — If the values in the n-th and m-th registers are equal, jump to the k-th instruction, else continue with the execution of the next instruction in the list.

We use the following conventions: (a) inputs are registered in the first registers of the memory, let us say, giving two inputs x and y, we consider R_1 for x and R_2 for y, (b) the other registers hold the number 0, and (c) when the machine halts the output may be read in the register R_1. With these conventions in mind, to add numbers x and y we can execute the following URM program:

01 :: $J(2, 3, 5)$	03 :: $S(3)$
02 :: $S(1)$	04 :: $J(1, 1, 1)$

The previous program halts on all inputs. The next program only halts on even inputs giving as output half of the input.

01 :: $J(1, 2, 6)$	04 :: $S(2)$	07 :: $J(1, 3, 10)$
02 :: $S(3)$	05 :: $J(1, 1, 1)$	08 :: $S(1)$
03 :: $S(2)$	06 :: $Z(0)$	09 :: $J(1, 1, 7)$

There are many variants of machines with registers (compare [13], [9]). Some of them are extended with a special set of instructions which gives good complexity properties for the model [11]. The example of a simulation of Turing machines by register machines can be found in [1].

In the subsections 3.1, 3.2, and 3.3 we translate each URM instruction in a program into a small Euclid machine; the final Euclid machine is obtained by concatenating the translations of all instructions, redefining jumps of the type $X \in C : n$ through appropriate shifts.

3.1 The emulation of registers

Lines will be used to remember values of registers as Figure 5 presents. The distance of point X to P, where X, *lying somewhere in the line R between P and the circumference of the circle with the center P and through the point Q*, informs us about the current value of some register which is put equal to

$$\log_2 \frac{|\overline{PQ}|}{|\overline{PX}|}.$$

In the case we need to put zero into some register we should move the point X to the intersection of R and C again. Representing numbers is thus obtained by successive bisections of a corresponding line segment.

The reader may quest the efficiency of such division process. The answer is subtle but simple: the unbound character of a tape in a Turing machine corresponds to the increasing precision needed to store a given natural number — as much tape needed for a computation as much precision needed to compute in bounded space. This simulation issue is well-known from dynamical systems that simulate Turing machines. However, if a bound on space is provided together with the Turing machine, then finite precision will suffice to compute over the reference circle and equality between two points can always be established by finite means.

Let us observe that without any space restriction we can start always with sufficiently large line segments. Then, after a finite process of successive bisections, we could obtain sufficiently large resulting line segments to be compared. So, the problem of efficiency for this representation is connected with the problem space resources for Euclid machines and of restrictions for a tape length.

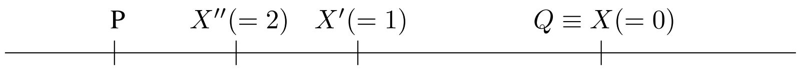

FIGURE 5
Values in one register

Let us consider a URM program P, and let r be a number of registers used in this program (which can be obtained by syntactical inspection of P). Then we will use a pencil of r lines (i.e., a set of straight lines meeting at some point) to emulate these registers. The line R_i is used to remember values of the i-th register as in Figure 5. The construction (which is presented in Figure 6) will be done in the following way:

- Draw two distinct points P, Q.
- Draw the line through P and Q.
- Label this line as R_1.
- Draw the perpendicular to R_1 in P.
- Label this perpendicular as R_n.

- Construct the bisector of R_1 and R_n.
- Label it as R_{n-1}.
- Construct the consecutive bisectors of R_1 and $R_{n-1}, R_{n-2}, \ldots, R_2$ and label them as $R_{n-2}, \ldots, R_1$.
- Draw the circle with the center P and going through the point Q.
- Label this circle as C.
- Label the intersections of C and $R_1, \ldots, R_n$ as $X_1, Y_1, \ldots, X_n, Y_n$.

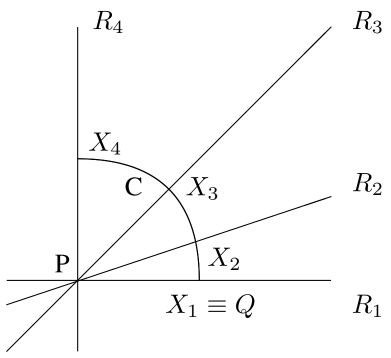

FIGURE 6
Simulation of 4 registers.

Let us add an important remark. During the whole computation (or rather drawing) the labels of the main elements of our system, i.e., the starting points P, Q, register lines R_1, …, R_n, and the circle C will be not removed or changed.

3.2 The translation of URM instructions

Let us describe the translation of URM instructions into operations of Euclid machines

$Z(n)$: move the point X_n to the intersection of R_n and C

$k + 00 :: D(X_n)$
$k + 01 :: LP(R_n, C; Y_n, X_n)$
$k + 02 :: D(Y_n)$

$S(n)$: divide the segment $\overline{PX_n}$ into two sub-segments with the same length and label the center point as X_n (look at Figure 7)

$k + 00 :: C(P, X_n)$
$k + 01 :: LC(P, X_n; C_1)$
$k + 02 :: C(X_n, P)$
$k + 03 :: LC(X_n, P; C_2)$
$k + 04 :: LP(C_1, C_2; P_1, P_2)$
$k + 05 :: L(P_1, P_2)$
$k + 06 :: LL(P_1, P_2; L)$
$k + 07 :: D(X_n)$
$k + 08 :: LP(L, R_n; Y_n, X_n)$
$k + 09 :: D(C_1)$
$k + 10 :: D(C_2)$
$k + 11 :: D(P_1)$
$k + 12 :: D(P_2)$
$k + 13 :: D(L)$
$k + 14 :: D(Y_n)$

$J(n, m, p)$: test whether the point X_n is in the circle with the center P and the radius PX_m and whether the point X_m is in the circle with the center P and the radius PX_n

$k + 00 :: C(P, X_n)$
$k + 01 :: LC(P, X_n; C_n)$
$k + 02 :: C(P, X_m)$
$k + 03 :: LC(P, X_m; C_m)$
$k + 04 :: X_n \in C_m : k + 06$
$k + 05 :: P \in C : k + 7$
$k + 06 :: X_m \in C_n : k + 10$
$k + 07 :: D(C_n)$
$k + 08 :: D(C_m)$
$k + 09 :: P \in C : k + 13$
$k + 10 :: D(C_n)$
$k + 11 :: D(C_m)$
$k + 12 :: P \in C : q$

where q is the starting number of the Euclid corresponding instruction, equivalent to the p-th instruction of the URM program.

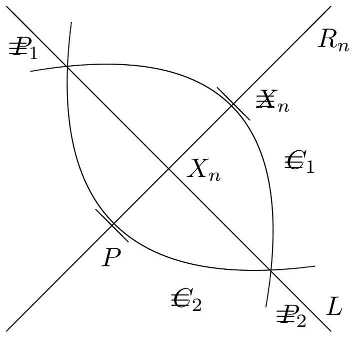

FIGURE 7
Simulation of a $S(n)$ instruction.

Note that, in the machine above, we used the *unconditional jumping instruction $P \in C$*. This unconditional jumping could have been translated directly from the URM language into an appropriate geometrical instruction. Indeed, whenever such an instruction $J(n, n, q)$ is found we can translated directly to an Euclid instruction $P \in C : q$, where q is suitably chosen.

3.3 Euclid machines are Turing complete

Let us add the we have to proceed with the re-enumeration of the instructions due to the fact that every Z instruction needs 2 operations, every S instruction needs 14 operations and J needs 13 operations. With this re-enumeration we have a complete description how to translate any URM program into some Euclid machine. So, we obtain the following proposition.

Proposition 3.1 *Every URM program can be simulated by some Euclid machine.*

Now we proceed to exemplify with one URM program — the binary addition, considered in the beginning of Section 3. More illustrative examples can be built straightforward by the reader.

Example 3.2 *Consider simple example of the sum of two natural numbers. We start with a preparation of* 3 *lines* R_1*,* R_2*,* R_3 *of the same pencil with the center* P*, and the circle* C *going through these lines with the points of intersections called* X_1*,* X_2*,* X_3*.*

\\ *draw the line* R_1
01 :: $P(P, Q)$
02 :: $L(P, Q)$
03 :: $LL(P, Q; R_1)$
04 :: $C(P, Q)$
05 :: $LC(P, Q; C)$
06 :: $D(Q)$
07 :: $LP(R_1, C; Y_1, X_1)$
\\ *draw the perpendicular line* R_3
08 :: $C(X_1, Y_1)$
09 :: $LC(X_1, Y_1, C_1)$
10 :: $C(Y_1, X_1)$
11 :: $LC(Y_1, X_1, C_2)$
12 :: $LP(C_1, C_2; S_2, S_1)$
13 :: $L(S_1, S_2)$
14 :: $LL(S_1, S_2; R_3)$
15 :: $D(C_1)$
16 :: $D(C_2)$
17 :: $D(S_1)$
18 :: $D(S_2)$
19 :: $D(Y_1)$
20 :: $LP(R_3, C; Y_3, X_3)$
21 :: $D(Y_3)$,
\\ *draw the bisector of the angle* $\angle\ R_3PR_1$ *and call it* R_2
22 :: $C(X_1, P)$
23 :: $LC(X_1, P; C_1)$
24 :: $C(X_3, P)$
25 :: $LC(X_3, P; C_3)$
26 :: $LP(C_1, C_3; S_2, S_1)$
27 :: $L(S_1, S_2)$
28 :: $LL(S_1, S_2, R_2)$
29 :: $D(S_1)$
30 :: $D(S_2)$
31 :: $D(C_1)$
32 :: $D(C_3)$
33 :: $LP(R_2, C; Y_2, X_2)$
34 :: $D(Y_2)$

To start a computation for some $n, m \in \mathbb{N}$ *we need to place the points* X_1*,* X_2 *on the lines* R_1*,* R_2 *in such a way that the following conditions hold:*

$$|\overline{PX_1}| = \frac{|\overline{PX_1'}|}{2^n},$$

$$|\overline{PX_2}| = \frac{|\overline{PX_2'}|}{2^m},$$

where X_0'*,* X_1' *represent the initial position of* X_1*,* X_2*. For this purpose we need to use* n *times the operation* $S(1)$ *and* m *times the operation* $S(2)$*.*

We can use the URM *program introduced above to implement the problem of an addition. We assume the arguments are in the registers* 1 *and* 2*; the rest of registers is initially equal to zero.*

Now this sequence of the URM *instructions can be translated into operations of the Euclid machine in the following manner.*

\\$J(2, 3, 6)$
01 :: $C(P, X_2)$
02 :: $LC(P, X_2; C_2)$
03 :: $C(P, X_3)$
04 :: $LC(P, X_3; C_3)$
05 :: $X_2 \in C_3 : 07$
06 :: $P \in C : 08$
07 :: $X_3 \in C_2 : 11$
08 :: $D(C_2)$

```
09 :: $D(C_3)$
10 :: $P \in C : 14$
11 :: $D(C_2)$
12 :: $D(C_3)$
13 :: $P \in C : 43$
\\$S(1)$
14 :: $C(P, X_1)$
15 :: $LC(P, X_1; C_1)$
16 :: $C(X_1, P)$
17 :: $LC(X_1, P; C_2)$
18 :: $LP(C_1, C_2; P_2, P_1)$
19 :: $L(P_1, P_2)$
20 :: $LL(P_1, P_2; L)$
21 :: $D(X_1)$
22 :: $LP(L, R_1; Y_1, X_1)$
23 :: $D(C_1)$
24 :: $D(C_2)$
25 :: $D(P_1)$
26 :: $D(P_2)$
27 :: $D(L)$
28 :: $D(Y_1)$
\\$S(3)$
29 :: $C(P, X_3)$
30 :: $LC(P, X_3; C_1)$
31 :: $C(X_3, P)$
32 :: $LC(X_3, P; C_2)$
33 :: $LP(C_1, C_2; P_2, P_1)$
34 :: $L(P_1, P_2)$
35 :: $LL(P_1, P_2; L)$
36 :: $D(X_3)$
37 :: $LP(L, R_3; Y_3, X_3)$
38 :: $D(C_1)$
39 :: $D(C_2)$
40 :: $D(P_1)$
41 :: $D(P_2)$
42 :: $D(L)$
43 :: $D(Y_3)$
\\$J(1, 1, 1)$
44 :: $P \in C : 14$
```

4 COORDINATES OF POINTS

What we have shown in the preceding sections is that a suitable encoding of URM programs exist in the Cartesian plane, by performing geometric constructions using an unmarked ruler and a compass. Many other such encodings exist, possibly more efficient. We did not really define computable functions in the sense of an Euclid-computable analogous to, e.g., the Turing-computable concept. In fact, we didn't need of that concept.

However, we can have it directly over the plan, as we are going to show in this section.

Let us recall some useful notions. A field $\mathbb{F}'$ is said to be a field extension of a field $\mathbb{F}$, if $\mathbb{F}$ is a subfield of $\mathbb{F}'$. Given some field we can extend it by several methods, for us the most natural one is to pick some elements p_1, ..., p_k, not in $\mathbb{F}$, and then to define $\mathbb{F}' = \mathbb{F}(p_1, \ldots, p_k)$ as the smallest field containing $\mathbb{F}$ and all p_j, with $j = 1, \ldots, k$. For instance, the real numbers can be extended by $i = \sqrt{-1}$ to the field of complex numbers.

In our case we are interested in points on the Euclidean plane with good (from the computational point of view) coordinates. The most convenient choice is the field $\mathbb{A}$ of algebraic numbers, which are computable and enumerable. Because we want to start with completely freely chosen points we need to extend this field by the set of all initial points (strictly speaking by the set of real, non-algebraic coordinates). Hence, for the starting points $P_1 = (x_1, y_1)$, ..., $P_k = (x_k, y_k)$ we obtain the extended field $\mathbb{A}(x_1, y_1, \ldots, x_k, y_k)$.

We can enumerate elements of such field $\mathbb{A}(x_1, y_1, \ldots, x_k, y_k)$ by natural numbers, hence the problem of any construction of points on Euclidean plane can be seen as some computation on natural numbers.

Let us make the above remark precise. Every construction available with Euclid machines is done by drawing circles, lines, and finding intersections. Hence, we can obtain coordinates of these newly constructed points from the previously constructed by solving systems of equations of at most second degree. This means that new points will be also in $\mathbb{A}(x_1, y_1, \ldots, x_k, y_k)$. In this way we have the following theorem.

Proposition 4.1 *For any Euclid machine, with the initial points $P_1 = (x_1, y_1)$, ..., $P_k = (x_k, y_k)$, all points reachable have their coordinates in the field $\mathbb{A}(x_1, y_1, \ldots, x_k, y_k)$.*

If we start with points with algebraic coordinates (in $\mathbb{A}$), then all constructed points will be also (with respect to their coordinates) in $\mathbb{A}$.

Proposition 4.2 *The set of algebraic points on the plane is effectively recursively enumerable.*

Proof. Consider any polynomial $p(x) = a_0 + a_1x + ... + a_nx^n$. We start by listing its integer coefficients

$$a_0, \ldots, a_n$$

and then transform this list into a list of natural numbers

$$b_0, \ldots, b_n,$$

using the encoding bijective map

$$\xi(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}.$$

The previous list of non-negative integers can then be encoded into a natural number by the well-known bijective map

$$\tau(b_0, \ldots, b_n) = 2^{b_0} + 2^{b_0+b_1+1} + \ldots + 2^{b_0+\ldots+b_n+n} - 1,$$

providing recursive enumeration (with repetitions) of all polynomials.

Now we apply Sturm's algorithm (see [6] for an overview and [3] for a detailed account) to effectively count the number of zeros of the given polynomial p. Let p_0, ..., p_r be the sequence of polynomials given by (a) $p_0 = 0$,

$p_1 = p'$ (the derivative of p), (c) for $0 < i < r$, there is a polynomial q_i such that $p_{i-1} = p_i q_i - p_{i+1}$ with $p_{i+1} \neq 0$ and $degree(p_{i+1}) < degree(p_i)$,[¶] and (d) $p_{r-1} = p_r q_r$. For any number y we denote by $\delta(y)$ the number of sign changes in the sequence $p_0(y)$, ..., $p_r(y)$ (ignoring zeros). Suppose that a and b are numbers that are not zeros of p, and $a < b$. Sturm's theorem states that the number of zeros z in the interval $[a, b]$ is $\delta(a) - \delta(b)$ (each zero being counted once only).

Consider again the original polynomial $p(x) = a_0 + a_1 x + ... + a_n x^n$. We take the natural number

$$M = |a_0| + \ldots + |a_n|,$$

such that all the zeros of p lie in the interior of the interval $[-M, M]$ (in fact, the rational number $M = 1 + \frac{1}{|a_n|}(|a_0| + \ldots + |a_{n-1}|)$ suffices). Then by Sturm's theorem the number of zeros of p is $\delta(-M) - \delta(M)$ which may be calculated effectively.

We can then encode together the code of polynomial p, the list number n of divisions of $[-M, M]$ necessary to separate all the roots of p (using repeatedly Sturm's theorem to successive divisions of the interval $[-M, M]$; if there are no roots, then n is taken to be 0) and the index k of the a root (if k is greater than $\delta(-M) - \delta(M)$, then index k is taken to be the degree of the polynomial plus 1), using the Cantor pairing coding function applied successively to these numbers in this order.

Let ζ be the number of roots of a polynomial given by its code in the way described above.

If ω is the enumeration established so far, then the new enumeration that eliminates polynomials with no roots can be given by ν as follows:

$$\nu(0) = \omega(\mu_y(\zeta(\omega(y)) \neq 0 \text{ and } \zeta(\omega(y)) \leq n)),$$

$$\nu(n+1) = \omega(\mu_y(y > \nu(n) \text{ and } \zeta(\omega(y)) \neq 0 \text{ and } \zeta(\omega(y)) \leq n)).$$

With ν we finally enumerate with repetitions all algebraic numbers by eliminating those codes that refer to polynomials without zeros and other situations.

Ultimately, each algebraic number correspond to a polynomial and that interval containing the algebraic number as the only root of that polynomial in that interval. Applying Newton's method, we can always specify a Turing

[¶] So that q_i and $-p_{i+1}$ are the quotient and remainder respectively when p_{i-1} is divided by p_i.

machine that calculates the successive digits of this algebraic number in its decimal expansion.

The final number is the code of an algebraic number. Coding together two algebraic numbers we get the code of a point in the plane. □

Thus, to any natural number corresponds an algebraic point in the plane and any algebraic point in the plane corresponds to a set of natural numbers (its infinitely many codes).

Now, let us observe this fact closer for its connection with computability. Let us denote by P_n the algebraic point P having index n.[§]

Definition 4.3 *We say that a $(n{+}1)$-ary relation $\mathcal{R}$ over $\mathbb{N}$ is Euclid-decidable if there exists a Euclid machine $E_{\mathcal{R}}$ such that, for all k_1, ..., k_n, k, $\mathcal{R}(k_1, \ldots, k_n, k)$ holds if and only if $E_{\mathcal{R}}$ starting from the initial points P_{k_1}, ..., P_{k_n} reaches P_k.*

5 UNDECIDABLE PROBLEMS

Let us clarify one important point. We can think about two different types of activity for Euclid's machines. The first one is connected with the described method of computation on encodings (given by points) of natural numbers. The second type of activity is simply drawing of points with a ruler and a compass. Now we need to distinguish carefully these two levels: a simulation of computations and drawings.

Let us exemplify this problem by means of the trisection problem. Angle trisection is the division of an arbitrary angle into three equal angles. It was one of the three famous geometric problems of antiquity for which solutions using only compass and ruler were sought (the other two were: cube duplication and circle squaring — see Section 1 for a full account). The construction was proved to be impossible by Wantzel (see [8]) only in 19th century. From this result we can infer an obvious corollary.

Proposition 5.1 *The problem of an angle trisection can not be solved by any Euclid machine.*

Proof. This statement can be more properly presented as a corollary to the following theorem 5.2. □

Namely, we can even ask about the possibility of the trisection by ruler and compass alone but restricted to points with algebraic coordinates. However, within infinitely many other cases,

[§] It n does not code for an algebraic number, then take the code of the first root of the first polynomial with a root as convention.

Proposition 5.2 *The angle $\pi/3$ can not be trisected.*

Proof. See the proof in [14], Theorem 5.4. The angle $\pi/3$ is the classical refutation instance of trisection. □

Since we can simulate Turing machines by Euclid machines, being the outputs of Turing computations reachable by Euclid machines, we may well wonder

Question 5.1 *What is the computational interpretation of such unsolvability?*

But now, we can reformulate the question about trisection. We can represent any angle $\angle AOB$ by three points A, O, B as in Figure 8. If we restrict ourselves to points from field $\mathbb{A}$, then with the use of the above mentioned coding we obtain the following new problem: does there exist such Euclid machine that given three numbers m, n, r, it finds a number representation s of the point of the trisection of $\angle AOB$, i.e., $\angle AOB = 3\angle AOP$ (e.g., with respect to the main circle).

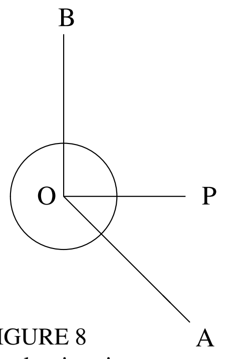

FIGURE 8
Angle trisection.

The first claim needing justification in this problem is the existence of such point P with algebraic coordinates. But this fact can be obtained by simple algebra taken from analytic geometry.

Proposition 5.3 *For any points A, O, B, with algebraic coordinates, there exists the point P with algebraic coordinates too, such that*

$$\angle AOB = 3\angle AOP.$$

Proof. We use simple methods of analytic geometry to prove that for an angle placed in the center of a given circle (where the center of this circle and the points of intersections of the angle with this circle are given by algebraic coordinates), then the point which gives a solution of the trisection problem on this circle has also algebraic coordinates.

Without any loss of generality we can identify the point O with the origin $(0,0)$, because we can always use a translation with algebraic parameters to obtain such a situation. Now, we have two lines: OA and OB, for $A = (x_A, y_A)$, $B = (x_B, y_B)$ (x_A can always be made different from 0 by some rotation) they have the equations: $x_A y - y_A x = 0, x_B y - y_B x = 0$. We can find now $\tan(\angle AOB) = \frac{y_B x_A - y_A x_B}{x_A x_B + y_A y_B}$. Of course, $\tan(\frac{1}{3}\angle AOB)$ can be found from the equation

$$\tan(\angle AOB) = \frac{3\tan(\frac{1}{3}\angle AOB) - \tan^3(\frac{1}{3}\angle AOB)}{1 - 3\tan^2(\frac{1}{3}\angle AOB)},$$

which means that $\tan(\frac{1}{3}\angle AOB)$ is an algebraic number.

The next step is devoted to compute the coefficient of the line OP given in the Cartesian plane by $y = ax$, with a given by

$$\frac{\frac{y_A}{x_A} + \tan(\frac{1}{3}\angle AOB)}{1 - \frac{y_A}{x_A}\tan(\frac{1}{3}\angle AOB)}.$$

And now to find coordinates of P all we need is a solution of the following system of equations with algebraic coefficients: $x_P^2 + y_P^2 = x_A^2 + y_A^2$ and $y_P = a x_P$, such systems have always algebraic solutions. □

In similar way, looking through the proof of last proposition, we can prove that:

Proposition 5.4 *For any points A, O, P, with algebraic coordinates, there exists the point B with algebraic coordinates too, such that*

$$\angle AOP = \frac{1}{3}\angle AOB.$$

So, now we are concerned with the crucial question. Can a code of P be computed? Our first observation is that if it would be possible for some URM machine, then this process of computation could be presented in the well known manner by Euclid machines. By observation of the proof of Proposition 5.4 we have such a method which can be performed on (possibly infinite) decimal expansion of the coordinates (for example, by machines of Type Two Theory [15] which can work on a tape filled with infinite number of digits). But our problem needs a computation on natural numbers, not on infinite sequences of digits. And, let us recall, that even if we can generate from a natural label of some algebraic number x its decimal expansion (see Section 4), it is impossible to obtain from finite subsequences of this expansion that natural number, which represents x (from density of the set of algebraic numbers we can always find infinite number of natural descriptions of algebraic numbers which agree with given finite sequence of digits).

But the above paragraph does not solve our problem. We can not compute a code of an algebraic point P from its decimal expansion, but maybe there is some direct method to solve this problem.

Theorem 5.5 *Let us define the trisection relation $T : \mathbb{N}^4 \to \mathbb{N}$ if and only if*

$$\angle P_m P_n P_r = 3\angle P_m P_n P_s.$$

Then T is not Euclid-decidable.

What about decidability of relation T?[||] in the following way $T(m, n, r, s)$ holds

Theorem 5.6 *T is not URM-computable (of course, neither Turing-computable).*

Proof. This is rather trivial, because if T was decidable, then we could decide equality between algebraic numbers given by procedures, i.e., given by Turing machines that calculate their decimal expansions. But this procedure can not be done in finite time.

It is sufficient to consider two arbitrary algebraic points P_m and P_n with codes, respectively, m and n. Let u_1, u_2 are the minimal points such that their trisection can be done by the point P_m. This can be described as:

$$u_1 = (\mu_u T((u)_1, 0, (u)_2, m))_1,$$

$$u_2 = (\mu_u T((u)_1, 0, (u)_2, m))_2.$$

Now the condition $T(u_1, 0, u_2, n)$ means that both P_n, P_m are trisection points for $\angle P_{u_1} P_0 P_{u_2}$. Let us observe that the same point can be used on the arms of some angle and as a trisection point if and only if this angle has a measure 0 degrees.

[||] With codomain in $\{0, 1\}$, since T is characteristic function.

Hence $T(n, 0, m, m)$ means that P_m lies on the line through P_0, P_n. But this condition does not exclude the possibility that P_n and P_m are two different points. So, let us demand that P_n, P_m are both on the different line $P_{u_1}P_n$ — in this case these points P_n and P_m have to be identical. Now we obtain the following condition

$$P_m = P_n$$

$$\text{iff}$$

$$T(u_1, 0, u_2, n) \wedge T(n, 0, m, m) \wedge T(n, u_1, m, m) \ holds.$$

□

The above theorem creates a question about a source of non-computability of the trisection problem. However the following explanation can be given.

There exist constructions trisecting the angle to an arbitrary degree of precision (see [14], Chapter 5). Given m, n, r, and s, such that $T(m, n, r, s)$ holds, there exist an infinite sequence of numbers $(s_i)_{i \in \mathbb{N}}$ such that an accessibility decidable predicate $U(m, n, r, s_i)$ exists, with $(P_{s_i})_{i \in \mathbb{N}} \to P_s$, although such a procedure for $T(m, n, r, s)$ does not exist, i.e., P_{s_i} is reachable from P_m, P_n, and P_r, for all i, but not P_s. Thus, the undecidability of T is a consequence of the fact that, although infinitely many algebraic points can be obtained as whole (entire) entities (the reachable ones) — objects in themselves with no need of a decimal expansion —, infinitely many others can only be approximated.

So far, it seems that the *halting problem* in this case is not a reason for not achieving decidability, since Turing machines that compute algebraic numbers have to operate forever.

As a consequence, we can state that to handle operations with algebraic numbers and questions on the computability of these operations the framework of Type Two Turing machines is much more suitable than that of Turing machines.

6 CONCLUDING REMARKS

It is very interesting to observe that the trisection function seems does not have a character of a self-referential problem (like, e.g., the halting problem). It would be worth of explanation whether such function has any connection to classical uncomputable functions and to find the proper place of such problem in the hierarchy of Turing degrees.

Let us also add that Fourier series can be interpreted as sums of circles with decreasing *radii*. E.g., planetary orbits in the plane can be studied from a computational point of view by means of possible approximations in the Euclidean plane. This could be used to obtain another (functional) interpretation of Euclid machines.

We can also ask the natural question: is every Euclid computable function also Turing computable? The obvious suggestion to this question is the answer **yes**, by Church's thesis. Of course, we can interpret this model as a model with infinite precision, which leads us to comparison with such constructions as BSS machines. Whatever, the fully mathematical answer will need a precise construction of a proof.

The next problem is to determine whether the non-deterministic character of labeling has an impact on Euclid algorithms. In other words: let us fix some convention of labeling in a given algorithm A. Can we always give some equivalent algorithm A' for any order of labeling? We can also ask what is a characterization of the class of Euclid algorithms invariant with respect to labeling.

The last (but not least) field of research is connected to the question about meaning of such geometrical machines in different geometries. Let us recall five postulates of Euclid in a formulation taken from [5].

1. A straight line may be drawn from any one point to any other point.
2. A finite straight line may be produced to any length in a straight line.
3. A circle may be described with any center at any distance from that center.
4. All right angles are equal.
5. If a straight line meet two other straight lines, so as to make the two interior angles on one side of it together less than right angles, the other straight lines will meet if produced on that side on which the angles are less than two right angles.

It is obvious to observe that the three first postulates are in a close correspondence with our set of instructions. But it is not so obvious how to put the fifth postulate in a construction of geometric machines. Let us use a different statement, which is equivalent to the fifth postulate.

- Given any straight line and a point not on it, there exists one and only one straight line which passes through that point and never intersects the first line.

With this formulation in mind we can try to do the following modification on the set of instructions. We can replace $L(P, Q)$: *draw the line through P and Q* by $L'(P, Q, R)$: *draw the line through R parallel to this line which could be drawn through P and Q* (this line through P and Q need not to exist).

Let us justify the fact that these operations are equivalent on Euclidean plane. To simulate $L(P, Q)$ by L' let us use the following program:

```
01 :: Draw the circle $C_1$ with a center $P$ and through $Q$
02 :: Draw the circle $C_2$ with a center $Q$ and through $P$
03 :: Label one of the intersection points of $C_1$ and $C_2$ as $R$
04 :: $L'(P, Q, R)$ // we have a parallel line to our desirable line through $P, Q$
05 :: Label this line $L_1$
06 :: Draw the circle $C_3$ with a center $R$ and through $P$
06 :: Label the intersection point of $C_3$ and $L_1$ as $S$
07 :: $L'(R, S, P)$ // this is the line through $P$, $Q$
```

We can also give a short description of the converse simulation. To simulate $L'(P, Q, R)$ by the instruction L we should draw the line through P, Q (which is done by $L(P, Q)$), label this line as L_1, now draw the perpendicular L_2 to L_1 through R. It is clear that the perpendicular to L_2 through R is the line parallel to P, Q and containing the point R.

Maybe a similar modification on instructions drawing circles is also possible. Namely, we could think about $C'(P, Q, R)$ — draw the circle through the points P, Q, R instead of $C(P, Q)$. The instruction C' is based on another version of the fifth postulate.

- The three non-collinear points always lie on a circle.

In this or that way we could have the operation based on the famous Euclid's fifth postulate. With a change of a type of an underlying geometry we will obtain new properties of the syntactically identical geometric algorithms. In this framework we could ask: what is a connection of non-Euclidean geometric machines with classical computational non-determinism?

7 ACKNOWLEDGEMENTS

A previous geometrical model was presented at York to the Conference *Unconventional Computation '06* (see [10]).

We thank to the many participants with whom we discussed the Euclid model, their criticism and their suggestions.

A special thank goes to Christian S. Calude, from the south hemisphere, and to Andy Adamatzky, our editor, for having invited us to submit an extended post-conference paper. We also thank to the anonymous referees and their competent suggestions.

José Félix Costa is very indebted to the *United Grand Lodge of England* for providing him with the DIVINE ARCHITECT, symbol of *first principle*.

REFERENCES

[1] Boolos, G. and Jeffrey, R.C. *Computability and Logic*, Cambridge University Press, 1989.

[2] Bunt, L.N.H., Jones, P.S. and Bedient, J.D. *The Historical Roots of Elementary Mathematics*, Dover, 1988.

[3] Cohn, P. M. *Algebra*, volume 2, Wiley, 1977.

[4] Courant, R. and Robbins, H. *What is Mathematics?*, Oxford University Press, New York, 1947.

[5] Coxeter, H.S.M, *Non-Euclidean geometry*, Mathematical Association of America, 1998.

[6] Cutland, Nigel J., *Cumputability, An Introduction to Recursive Function Theory*, Cambridge University Press, 1980, 1992.

[7] Klein, F. *et al*. *Famous Problems and other Monographs*, Chelsea, New York, 1962.

[8] Martin, G.E. *Geometric Constructions*, Springer-Verlag, 1998.

[9] Minsky, M. L. Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in Theory of Turing Machines. *Ann. Math.*, 74, 437-455, 1961.

[10] Mycka J., Coelho F., and Costa J.F. Euclid abstract machine: the trisection of the angle and the halting problem. In C. S. Calude, M. J. Dinneen, G.Paun, G. Rosenberg and S. Stepney (eds), *Unconventional Computation (UC 2006)*, volume 4135 of Lecture Notes in Computer Science, 195-206, Springer-Verlag, 2006.

[11] Papadimitriou, Ch. H. *Computational Complexity*, Addison Wesley, 1993.

[12] Plouffe, S. The computation of certain numbers using a ruler and compass. *Journal of Integer Sequences*, 1, 1998.

[13] Shepherdson, J.C. and Sturgis, H.E. Computability of recursive functions. *Journal of the ACM*, 10(2), 217-255, 1963.

[14] Stewart, I. *Galois Theory*, Chapman & Hall, 1973, 1995.

[15] Weihrauch, K. *Computable Analysis, An Introduction*, Springer-Verlag, 2000.

[16] Yates, R. C. *The Trisection Problem*, National Council of Teachers of Mathematics, Washington, D. C., 1971.