COMPUTATION: DAY 1

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. What is computation

Wednesday, January 18, 2023.

The theory of computation is a very new subject. The Turing Machine, an important element of this theory, was defined by Alan Turing in about 1936. Its invention was a response to several important developments. However, we can look back to as far as 300 or 400 BC, to the geometric constructions, to see computation systems not tied to the artifacts of the modern world. At the time, their computer was the combination of a ruler and a compass.

We consider the the calculation of the square root of an integer n. We need to consider what an answer would look like. Today, I think probably would consider an answer to be something like,

$$\sqrt{5} = 2.23606798...$$

But that is not even an answer, just the suggestion of an approximation, that can be refined on demand. In the world of our geometric constructions, a value is the

length of a segment. Therefore a solution might be: given a line segment S of length $|S|$, in a finite sequence of application of ruler and compass actions, produce a line segment with length $\sqrt{|S|}$.

2. Ruler and Compass Constructions

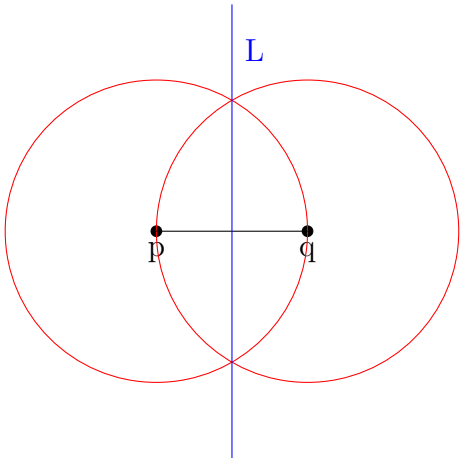
There will be only these ruler and compass actions allowed.

(1) A line can be draw by the edge of the ruler.
(2) An arc of a circle can be drawn by the compass.
(3) The ruler's edge can be aligned with one or two points.
(4) The compass can be sized to fit two points.

3. Perpendicular bisector

Just as we now write subroutines, which are generalized procedures to be invoked with parameters, our geometric constructions have subroutines, such as, *given two points, and a line connecting those points, construct a line perpendicular with passes through the midpoint of the two points.*.

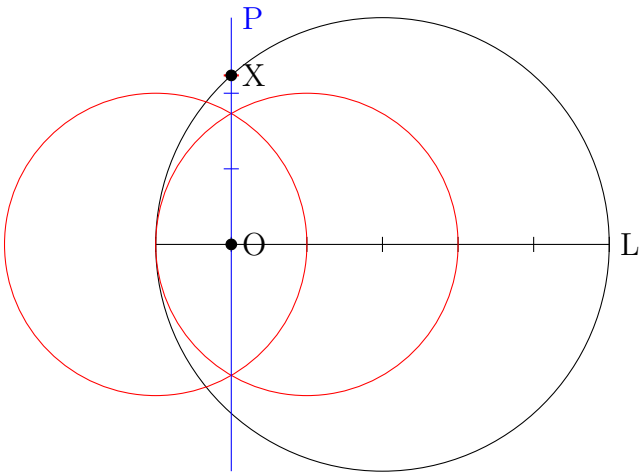
The input is have the points drawn on the paper, p and q. The program is the steps to take with ruler and compass, and the result is a new line L drawn on the paper. Here is the subroutine's result, with the red circles being constructed for, and the blue line the result.



4. The square root of 5

We give now the procedure for constructing the length $\sqrt{5}$. Create the line L with one unit to the left of point O, and 5 units to the right of point O. Invoke the perpendicular bisector subroutine result in a perpendicular P through O. Draw a

circle with diameter L. The intersection of the circle with P is X. The segment O to X has length $\sqrt{5}$.



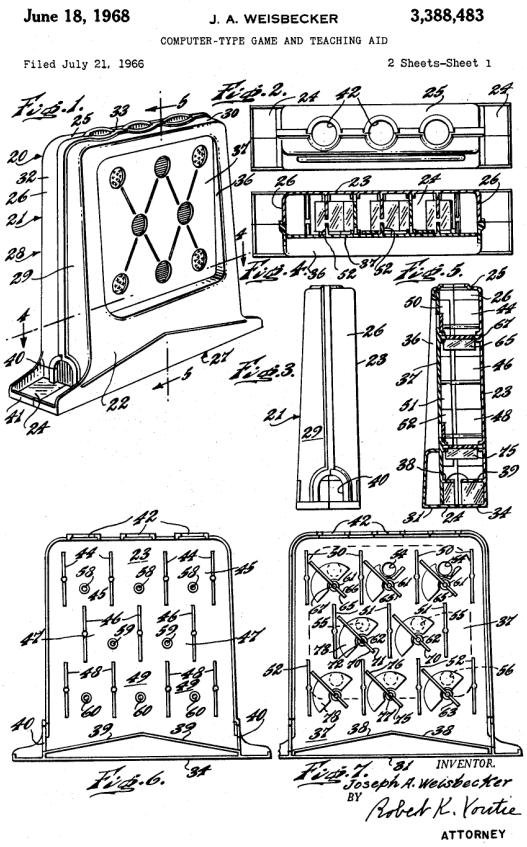
5. Angle Bisectors and Trisectors

Given two lines L_1 and L_2 intersecting at one point X, draw a line through X that bisects the angle between L_1 and L_2. However, it is not possible to trisect the angle. The rule and compass constructions can compute square roots, angle bisectors, but not angle trisectors. Constructs we sought for almost three millennia until proved impossible by Pierre Wantzel in 1837.

6. Think-A-Dot

The Think-A-Dot was created by Joseph Weisbecker and patented as US566881A filed 1966 with title *Computer-type game and teaching aid*. It illustrated the elements of a Finite State Automata.

The eight flip-flop elements show either yellow or blue. The drop of a marble into one of three holes at the top of the toy causes three of the flip-flop elements to change color. The color of the flip-flop determines the path of the marble.



7. Exercises

(1) Generalize from $\sqrt{5}$ to $\sqrt{n}$ for arbitrary positive integers n.
(2) Prove the construction gives the square root.
(3) Given an angle bisector construction.
(4) The Think-A-Dot has the property that if a marble is repeatedly dropped into the same hole, the pattern repeats at exactly every 8th marble. Prove this.

8. Set Membership

Wednesday, January 18, 2023.

A set is a mathematical object having a membership predicate. A set defined in any fashion that is sufficiently clear to determine membership. We are investigating which sets can be defined by a membership predicate calculated by a type of computer called a *finite automata*.

Given sets X, Y and U such that $X \subseteq U$, these new sets are obviously allowed, as the construction of a membership predicate follows easily,

$$\begin{aligned} X \cup Y &= \{\, z \,|\, z \in X \text{ or } z \in Y \,\} \\ X \cap Y &= \{\, z \,|\, z \in X \text{ and } z \in Y \,\} \\ \tilde{X} &= \{\, u \in U \,|\, u \notin X \,\} \\ X \times Y &= \{\, (x, y) \,|\, x \in X \text{ and } y \in Y \,\} \end{aligned}$$

While all of these should be familiar constructions, the *Kleene Star* is very important for us, and maybe be less familiar,

$$X^* = \bigcup_{i=0}^{\infty} X^i$$

We do not doubt the existence of such a set, because for an x of length i, we already have the predicate for $x \in X^i$.

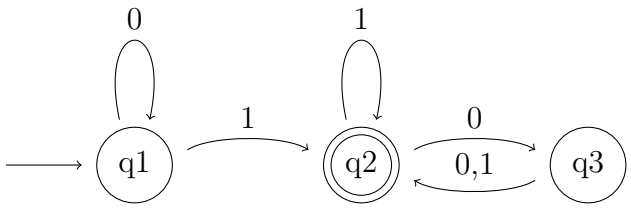
9. Finite Automata

A finite automata is the 5-tuple

$$M = \langle\, Q, \Sigma, \delta, q_o, F \,\rangle$$

where Q is a finite set of states, Σ is a finite set called the alphabet, δ is the transition function, $\delta : Q \times \Sigma \longrightarrow Q$, a distinguished state at which the computation begins, $q_o \in Q$ and a set of states $F \subseteq Q$ such that, if the computation ends at a state in F the computation is *accepting*.

There is also a set of visual conventions for drawing a finite automata as a labeled, directed graph, with the nodes as states, the edges as transitions, labelled with the alphabet letter, and double circled nodes for final states, and an arrow towards the start state.



10. Regular Languages

Given a finite automata $M = \langle\, Q, \Sigma, \delta, q_o, F \,\rangle$, a computation $M(s)$, with $s \in \Sigma^*$ is defined by lifting the transition funtion δ from the domain Σ to the domain Σ^* using the rules,

(1) $q \xrightarrow{\epsilon} q$, where ϵ is the empty string,
(2) $q \xrightarrow{\sigma} q'$, where $\sigma \in \Sigma$ and $\delta(q, \sigma) = q$;,
(3) $q \xrightarrow{s} q'$, where $s \in \Sigma^*$, and $q \xrightarrow{\sigma} q'' \xrightarrow{s'} q'$, with $s = \sigma s'$ and $\sigma \in \Sigma$.

The computation is used as a membership predicate for a set,

$$\mathcal{L}(M) = \{\, s \in \Sigma^* \mid q_o \xrightarrow{s} f,\ f \in F \,\}$$

A language $L \subseteq \Sigma^*$ is *regular* is there is a finite automata M such that $L = \mathcal{L}(M)$.

11. Exercises

(1) Given the set $X = \{\, 1, 2, 3 \,\}$ and the set $Y = \{\, 2, 4, 6, \,\}$, what is $X \cup Y, X \cap Y$ and $X \times Y$.
(2) For the finite automata in the diagram above, complete this description of its transition function δ: $\delta(q_1, 0) = q_1, \delta(q_2, 0) = q_3, \ldots$.

12. How to Design an FA

Monday, 23 January 2023

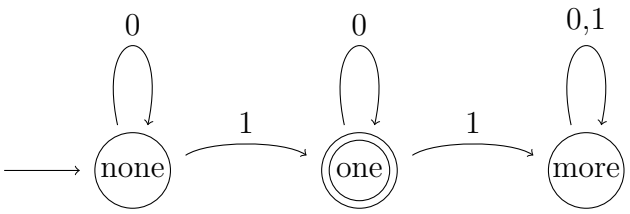
Consider the regular language,

$$J_1 = \{\, w \in \{\, 0, 1 \,\} \mid w \text{ contains exactly one } 1 \,\}.$$

Here are remarks on designing this, and other, finite automata.

(1) A great help to the design process is to locate and clarify the set of states needed. Once the states are identified, the machine builds itself.
(2) It is helpful to figure out how many states are needed.
(3) If the machine accepts something, there must be an accept state. If the machine also rejects something there has to be a distinct state that is not an accept state.
(4) Because the state set is finite, but the possible input without bound, some state has to summarize an infinity of inputs.

(5) States need not be distinguished if once in that state, it remains either accepting or non-accepting, not matter what the future transitions. These are *absorbing states*, with all arrows leaving returning back — once entered, never left.
(6) Once the states are identified, the start state is the state which is the outcome of computing on the empty string.
(7) In this machine, we need to be able to count one 1, hence we need the state that represents exactly that.
(8) Of the states that represent the situation of not one 1, we have to distinguish between those with less than one 1, because they have various future possibilities (some will end up as accepting and some will not) and more than 1 one which have a fixed future possibility (they are non-accepting now and will forever be so).
(9) Hence the machine has three states; the start state corresponds to the empty string of zero 1's, and the accept state the unique state representing one 1; and the third state of more than one 1 is absorbing.
(10) In this machine, since 0's do not affect the outcome, the states will not be conditioned on any statement about zeros.
(11) Hence the machine as shown below. Once the states are are written down, the machine builds itself.



Wednesday, 25 January 2023

The machines M1 and M4 from the class textbook were presented, and the Project 2 assignment and the code for the `SimpleFiniteAutomata` class was presented, see github class (not public). We ended with some considerations concering regular languages of the form *"the string contains the substring s"*.