



Introduction to the Theory of Computation

(Second Edition), Michael Sipser

Lecture Notes developed by Teodor Rus ^a

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

^aThis notes have been developed by Teodor Rus. They are copyrighted materials and may not be used in other course settings outside of the University of Iowa in their current form or modified form without the express written permission of one of the copyright holders. During this course, students are prohibited from selling notes to or being paid for taking notes by any person or commercial firm without the express written permission of one of the copyright holders.

Contents

- History;
- Current trends of computation theory;
- Problem solving methodology;
- Polya Four Steps Methodology;
- Problem Classification;
- Problem Solving and Areas of Computation Theory;
- Problem Solving Tools;
- Relationship to Theory of Computation;
- How to Approach It.

History

The three sources of theory of computation are:

1. Works performed by mathematicians such as Leibnitz and Pascal, trying to mechanize calculus and logical deduction;
2. Work performed by logicians such as Kurt Gödel, Alfred Tarski, Alan Turing, Alonzo Church, A. A. Markov, toward the development of a formal concept of algorithm;
3. Development of computer based problem solving methodology.

Gottfried Wilhelm, Leibnitz

- German mathematician, 1646–1716;
- Invented Calculus (independent of Newton)
(Leibnitz's notation is still used);
- Discovered binary system.

Blaise, Pascal

- French mathematician, 1623–1662;
- Invented mechanical calculators;
- Worked on Projective geometry and Probability Theory (with Fermat);
- Since 1654 to the end he did philosophy and theology;
- Known to us through "Pascal language"
(Developed by Nicholas Wirth 1970).

Kurt, Gödel

- Austrian American Logician, 1906 (Brün, Bruno, Czech), 1978 (Princeton, NJ);
- Proved Incompleteness theorem thus challenging Hilbert's Program (1900).

Alfred, Tarski

- Polish American Mathematician+Logician, 1901 (Warsaw, Poland) – 1983 (Berkeley, Ca);
- Invented the formal concept of truth and model theory.

Alan, Turing

- English Mathematician, 1912–1954, known as the father of CS;
- Invented the formal concept of an algorithm (Turing Machine);
- Invented Enigma machine, a code-breaker during Second World War.

Alonzo, Church

- American Logician, 1903–1995;
- Invented the λ -calculus;
- Known for Church's thesis:

everything that is computable is computable
by a recursive functions (TM) !

Andrey Andreevich, Markov

- Russian Mathematician 1903–1979
(sun of A.A. Markov 1856–1922);
- Developed Russian school of constructive mathematics and logic;
- Invented normal algorithms and proved $NA \simeq RF \simeq TM \simeq \lambda C$.

Current trends

- Algorithm development and information (computation) technology, IT;
- Solving problems related to computation complexity;
- Putting information technology on firm mathematical basis;
- Developing domain specific languages (DLS) to express computations;
- Developing domain specific computer based problem solving methodology.

Facts

1. Most of the current texts dedicated to the theory of computation disregard the influence computer based problem solving methodology has(d) on the development of theory of computation.
2. Provide (please) rationale for this situation!
3. We will try to correct this in this offering of 22C:131 by choosing a textbook that can be best used to connect the theory of computation with problem solving methodology.

Unfortunately: following tradition this class will have no programming component!

Problem solving methodology

- **Problem:** (paraphrasing G.Polya, Stanford 1956)
to have a problem means to search consciously for some action appropriate to attain a given aim.
- **Solution:** to solve a problem means to find such an action.

See G. Polya *How To Solve It* 1956.

Observations

- A problem is characterized by three principal components:
 - **unknowns**,
 - **data**, and
 - **conditions**.
- To solve a problem means to find a way of determining the unknowns from the given data such that the conditions of the problem are satisfied.

Example problem

Find all real numbers x that for a given triple (a, b, c) of real numbers satisfy the relation

$$ax^2 + bx + c = 0$$

- Unknown: real numbers x
- Data: real numbers a, b, c
- Conditions: $a \neq 0$ and $ax^2 + bx + c = 0$

Solving the problem

Use the properties of the data and conditions to deduce a relation $x = f(a, b, c)$

`a x^2 + b x + c = 0 /* by problem statement */`

`a x^2 + b x = -c /* by adding -c to both terms of equality */`

`(x^2 + b/a x) = -c/a /* by dividing both term by a */`

`(x + b/2a)^2 = (b^2 - 4ac)/4a^2 /*by using binomial identity */`

`(x + b/2a) = +/- sqrt(b^2-4ac)/2a /*by taking square-root */`

`x=(-b +/- sqrt(b^2-4ac))/2a /*by adding -b/2a to equality */`

Solutions:

$$x_1 = (-b + \sqrt{b^2 - 4ac})/2a$$

$$x_2 = (-b - \sqrt{b^2 - 4ac})/2a$$

Difficulties in solving a problem

- In finding a way of determining unknowns; this belongs to the very notion of a problem.

Citing Polya:

“where there is no difficulty there is no problem”.

- It seems that what best characterizes human activity is **solving problems by finding ways around or out of difficulties.**

Two classes of difficulties

- Difficulties that pertain to finding the action;
- Difficulties that pertain to performing the action.

Example: solving linear systems of equations

1. One can use Cramer's rule or Gaussian elimination method;
2. One may know how and one can do it for small systems (system size < 10);
3. One could do it for large systems too (system size > 1000) but perhaps she will not finish during her entire life.

Distinction between difficulties

The distinction between the two classes of difficulties is in solution mechanisms:

- Finding the way around difficulties in discovering the action seems to be a characteristic of intelligent human behavior.

What could one use to help this?

- Finding the way around difficulties of performing the action can be done both creatively and mechanically.

What could one use to help this?

Mathematical methodology

Polya's four steps methodology:

1. Formalize the problem;
2. Develop **an algorithm** to solve the problem;
3. Perform the algorithm on the data characterizing the problem;
4. Validate the solution.

Question: is there any difference between this and computer based methodology?

Problem formalization

- Express the three components of the problem, *unknown*, *condition*, and *data* as mathematical objects;
- Define the problem as a parameterized element of a mathematical language, i.e. *use a model*;
- Parameters of problem model are condition and data;
- An instance of the problem model is obtained by individualizing the condition and data.

Example formalization

Consider the problem of solving quadratic equations:

- Express unknown, data, and condition
- The model:
- Parameters:
- Instance:

Result

The result of “problem formalization” step depends upon mathematical knowledge and problem understanding.

Note: one cannot reasonably hope to solve a problem that one does not understand.

The solution algorithm

The solution algorithm is a sequence of coordinated operations starting from the data and ending in the required unknowns.

It could be:

- Mathematically defined over a class of problems a member of which the problem at hand is.

Example please: ?

- A heuristic that is used for one given problem when no good algorithm is known.

Example please: ?

Performing the algorithm

- Instantiate the problem by appropriate data, conditions, and unknown;
- Apply the algorithm !

Example?

Validation of the solution

- Prove that the solution satisfies the conditions in the instance of the problem;
- Test the solution on all configurations of data and conditions.

Question: what is the difference between *prove* and *test*?

Classification

- Analysis problems;
- Synthesis problems.

Analysis problems

- Problem model does not represent a relation between unknowns, data, and conditions;
- There is no general solution algorithm to determine the unknowns;
- The solution algorithm is a heuristic.

Note: most of the problems faced by Computer Science are analysis problems!

Synthesis problems

- Problem model does represent a relation between unknowns, data, and conditions;
- There exists a general solution algorithm that determines the unknowns.

Note: according to the model and the algorithm we have problems “to find” and problems “to prove”

Synthesis problems are characteristic to the **well-formalized application domains !**

Problems “to find”

Problem model expresses the unknown in terms of data under conditions and the algorithm constructs the solution by algebraic operations on data abstractions.

Example: The algorithm to find the solutions of the equation $ax^2 + bx + c = 0$

Problems “to prove”

Problem model expresses the unknown as a statement to be proved or disproved and the algorithm constructs the logical inferences.

Example: Pythagoras' Theorem:

in a right triangle the square of the side opposing the right angle (the hypotenuse) is the sum of the square of the sides opposing the other two angles

Areas involved

The three traditionally central areas of a *theory of computation* are:

1. **Automata theory:** provides problem solving tools (PST) Example: DFA, NFA, TM, etc..
2. **Computability theory:** provides the framework to characterize computation power of PST-s.
3. **Complexity theory:** provides the framework to characterize the complexity of a computation performed by PST-s.

Facts

1. The complexity of a computation is characterized in terms of **time and space required by the PST** to perform it.
2. Computability of a computation is characterized in terms of the **existence of a PST** to perform it.
3. PST that perform a computation are characterized in terms of **human ability to understand and formalize computation!**

Problem Solving Tools

- Problem solving tools of general interest for computation theory are *computers*.
- Yet computers are not the topic of interest for computation theory.

Can somebody explain this?

Relationship

- The three areas involved in the theory of computation are all about PST-s.
- All PST are seen here as theoretical models of computers.
- Hence, the three areas involved are linked by the question:

What are the fundamental capabilities and limitations of computers?

More on the history of question

what are the fundamental capabilities and limitations of computers?

- This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation;
- Technological advances since that time have increased our ability to compute and evolved our intuition of computation;
- This has brought this question into the world of practical concern!

Interpretation

- Each of the three areas – automata, computability, complexity – interprets this question differently and provides answers according to its interpretation.
- The natural relationship between the three areas is:

automata \rightarrow *computability* \rightarrow *complexity*

Note: interpret \rightarrow as “large implies”

Our approach

To better understand CS implications of this relationship we introduce the three fundamental areas for computation theory in reverse order.

Complexity theory

- Real life problems come in different varieties: some are easy to solve and some are hard!
- **Example:** sorting numbers is easy while scheduling events is hard.
- The central question for *Complexity theory* is: *what makes some problems computationally hard and others easy?*
- There is no yet answer to this question, though it has been intensively researched for the last 25 years!

Major achievements

- The scheme for classifying problems according to their computational difficulty using **time** and **space** required for problem solving algorithm:
linear, polynomial, exponential with problem size
- **Note:** this scheme assumes implicitly that an algorithm to solve the problem is available.

Goodness of this scheme

- This scheme is analogous to the periodic table for classifying elements according to their chemical properties;
- Using this scheme we can demonstrate a method for giving evidence that certain problems are computationally hard, even if we are unable to prove that they are so!

However: the discovery of new algorithms and computers can change the complexity of some problems.

Confronting problem to scheme

1. Understand which aspect of the problem is the root of difficulty;
2. Settle for less than perfect solution to reduce the difficulty;
3. One may be satisfied with a procedure that works for most cases (avoiding the worst case scenario);
4. One may consider alternative types of computation, such as randomized computation, that can speed up certain tasks.

Areas affected: cryptography

- In most cases an easy computational problem is preferred to a hard one because an easy one is cheaper to solve.
- Cryptography is unusual because it specifically requires hard computational problems.
- The secret codes should be hard to break without the secret key or password.
- Complexity theory has pointed cryptographers in the direction of computationally hard problems around which they have designed revolutionary new codes.

Computability theory

- Range of problems expand from very simple, whose solution algorithm can be executed by “hand”, to very complex, whose solution algorithm cannot be carried out even by the most powerful computers.
- Mathematicians identified such problems and called them “undecidable” or “unsolvable”.
- Example is the problem of determining if a mathematical statement is true or false.

It has been shown that no computer algorithm can perform this task.

Consequences

- Development of ideas concerning theoretical model of computers that led to the construction of actual computers.
- Avoid putting expensive effort in solving problem that cannot be solved.
- Develop alternative methodology for problem solving that accept less than ideal solutions.

Note: new models of computation are developed (*hypercomputation*) where conventionally unsolvable problem become solvable.

Relationship

- Theories of computability and complexity are closely related:
 1. In complexity theory the objective is to classify problems as easy and hard;
 2. In computability theory the objective is to classify problems as solvable and unsolvable (by the current computers);
- Computability theory introduces concepts used in complexity theory.

Example: measure the computation complexity in terms of number of operations required by the PST to perform it.

Automata theory

- Automata theory deals with the definition and properties of the mathematical models of computation.
- These models provide the tools used to solve problems (to develop solution algorithms) in several areas of computer science.

Questions are: **What is a computation?**

Can we construct models of computations?

Example models and tools

- The model provided by **Finite Automata (FA)**. FA are used for the development of software tools used for text-processing, compiler construction, hardware design, etc.
- The model provided by **Homomorphism Computation (HC)**. HC is used for the development of tools (such as logarithm computation) used in early technology (tabular computation) and more recently for parallel computation.
- The model provided by **Context-Free Grammars (CFG)**. CFG are used as specification and implementation tools for PL and in artificial intelligence.

Relationship

- Automata theory is an excellent place to begin the study of theory of computation;
- Theories of computability and complexity require a precise definition of a *computer* which is provided by automata theory;
- Automata theory allows practice with formal definitions of computation as it introduces concepts relevant to other nontheoretical areas of computer science.

Framework

- We will start our study of the theory of computation by first providing the mathematical notions and terminology. This will be an open-ended task allowing us to expend as necessary.
- Then we will use these notions in the second part where we will study automata theory and its applications to CS.
- The third part of the class will be dedicated to the computability theory.
- The fourth part of the class will be complexity theory!