cSplash, Event for High School Students, Courant Institute

What is the Numerical Halting Problem?

Chee Yap
Computer Science Dept, Courant, NYU

Mar 29, 2008

Abstract

The famous Halting Problem is the "representative problem" for what any conceivable physical computer can compute. We briefly introduce Computability Theory by using the elegant formulation of Turing Machines of Alan Turing (1937). Turing computability addresses computation over countable sets (like strings). But many computational problems in Mathematics involve real numbers. The set of real numbers is uncountable, and this causes a real strain on Turing Machines!! In fact, current theories of computation over real numbers are not satisfactory (we briefly mention two competing approaches). The source of this mystery is what we call the "numerical halting problem". Come hear why this problem s so central.

1 Introduction

- HANDOUT (This sheet)
- CLASS SURVEY: What is $\mathbb{R}$? Kinds of ∞? Know Turing Machines?
- Are there problems that a computer cannot do?
- Are there problems that we STILL do not know whether a computer can do?

2 What can be computed by a Computer? (ca. 1930)

- Notations: Natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, and strings S^* where S is any set of symbols. E.g., if $S = \{0, 1\}$, S^* is the set of binary strings.
- Finite Automata M (see Figure 1).

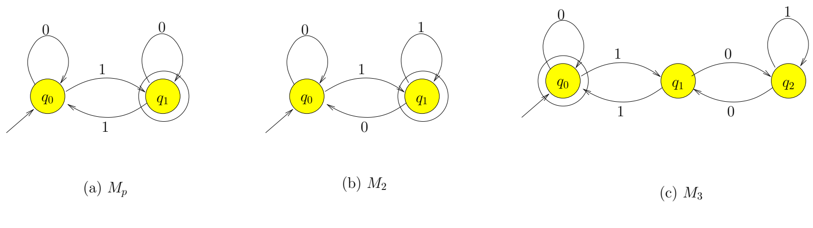

Figure 1: Finite Automata for Parity, Mod-2 and Mod-3 counters: try these for $w = 0110$

- EXERCISE: build M_5 to count Mod-5.
- So M has a set Q of **states**, a set S of **symbols**, and a set of **transition rules** $(q, b, q') \in Q \times S \times Q$.

Mom, can I buy an M_{23}?

$$\boxed{(q, b, q'):\ \text{“if in state } q \text{ and you see } b\text{, go to } q'\text{”}}$$

(Oh, also **start state** q_0 and **final state** q_f.)

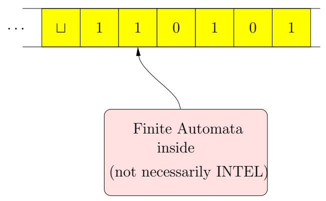

Figure 2: Turing Machine reading/writing an infinite roll of paper (tape)

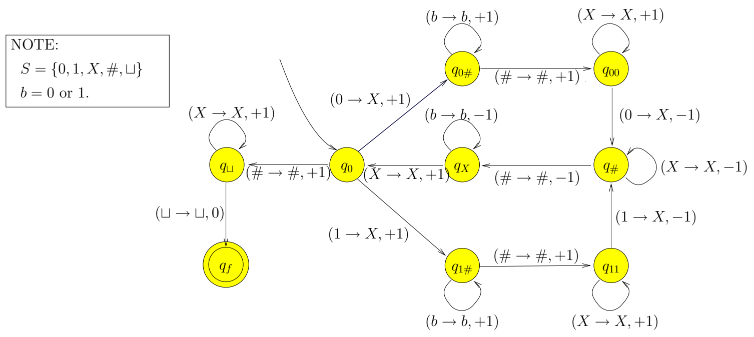

Figure 3: Turing Machine for duplicates: try it for $w = 10\#10$.

- Turing Machines (TM) is a finite automata that has an an infinite roll of paper (tape). See Figure 2 and Figure 3.

OK, *you* may call it toilet paper

- EXERCISE: build a TM to recognize "palindromes", $w\#w^{rev}$ where w^{rev} is the reverse of w.
- A TM is a finite automata, but has transitions has the form:

$$\boxed{(q, b, q', b', \pm 1): \text{ "if in state } q \text{ and you see } b\text{, go to } q'\text{, write } b' \text{ and move } \pm 1\text{"}}$$

- How does a TM computation end? Halts or Loops!

Great, TM never crashes!

- A Turing machine M **accepts** the set $L(M)$ of strings which leads to the final state. (M may not have to halt on inputs it does not accept.) A set L is **recursive** if it is accepted by a TM that halts on all inputs.
- (Gödel Numbers) Each TM's description can be viewed as a natural number $\mathbb{N} = \{0, 1, 2, \ldots\}$. Let TM_n be the TM whose description is n.
- The Halting Problem is to accept the set: $H = \{n \in \mathbb{N} : TM_n(n) \text{ halts}\}$.

The Halting Problem!

- THEOREM: The set H is not recursive.
  (Sketch: Suppose TM_{2008} accepts H. We construct another TM which imitates TM_{2008} but does the opposite outcome. This is $TM_{n'}$ for some n'. What will TM_{2008} do with n'? Get contradiction!)
- THEOREM: The set H is accepted by some non-halting TM. (Sketch: Easy – on input n, just simulated the TM_n

Great, a consolation prize

- CONCLUSION: In a certain sense, accepting H is the hardest problem for TMs. Any problem you can solve on a TM, you can "reduced" it to H.

3 What about computing over Reals? (ca. 2008)

- Most problems in scientific computing and engineering involve real numbers.
  Problem: $\mathbb{R}$ is uncountable. Finite strings are insufficient as TM inputs.
- Two current approaches:
  (Analytic Approach) Allow the TM to read infinite input sequences (going back to Turing).
  (Algebraic Approach) View real numbers as "atomic" and we have the capability to directly perform arithmetic operations on them, including testing its sign.
- Many computations reduce to **decisions** which is dependent on knowing if a number is 0.

The ZERO Problem!

- E.g., in computational geometry, we want to know if a point is on a line, or to the left or right of the line.
- Numbers have canonical names (e.g., $0, 1/2, \pi, \sqrt{2}$). But we are not given canonical names, but expressions!
  Are the following expressions zero?

Is there a problem?

$$1-1+1-1, \qquad 2^2+5-3^2, \qquad 1-\sum_{n=1}^{\infty} 2^{-n}, \qquad \sqrt{2}+\sqrt{3}-\sqrt{5+2\sqrt{6}}$$

$$3\log(640320)/\sqrt{163}-\pi,$$

$$\left(\sqrt{1000001}+\sqrt{1000025}+\sqrt{1000031}+\sqrt{1000084}+\sqrt{1000087} \quad +\sqrt{1000134}+\sqrt{1000158}+\sqrt{1000182}+\sqrt{1000198}\right)$$
$$-\left(\sqrt{1000002}+\sqrt{1000018}+\sqrt{1000042}+\sqrt{1000066}+\sqrt{1000113} \quad +\sqrt{1000116}+\sqrt{1000169}+\sqrt{1000175}+\sqrt{1000199}\right)$$

- Try using a hand calculator on the above (they all appear as zero)
- Precision: if $x, \widetilde{x}$ are real numbers, and $|x-\widetilde{x}| < 10^{-n}$ then $\widetilde{x}$ is an approximation of x with **precision** n. E.g., 3.142 is an approximation of π to precision 3.
- Assume we can approximate constants like π, and do operations like $+, -, \times, \div, \sqrt{\cdot}, \log, \ldots$ to any precision we choose. There are software for doing this. *Using such software, can we decide if an expression E is zero?* If $E \neq 0$, we will halt; but if $E \neq 0$, we can never halt!

Ahh, I see why you call it the the numerical halting problem!

- Problems with current approaches:
  (Analytic Approach) It is impossible to decide zero.
  (Algebraic Approach) Deciding zero is trivial (but it shouldn't be).
- CONCLUSION: So what do we know here? Not much more than the algebraic case (like $\sqrt{\cdot}$)! If the answer is NO (like the HALTING problem), it means that many of our continuous computations can never be guaranteed.

4 Conclusion

We talked about real numbers as a surrogate for continuous computation. Similarly, computing with finite strings is a surrogate for discrete computation.

Fundamental questions about the nature of discrete computation have been answered from the 1930's by logicians such as Turing, Church, Gödel, Kleene, etc.

But even a suitable computational model for continuous computation is unsettled in 2008. We invite you (who have a long career ahead) to think about such basic questions.

To learn further: `http://cs.nyu.edu/exact/`.