Lecture 10

Turing Machine & Computability

Zvonimir Rakamarić
University of Utah

Announcements

- Any questions about the homework assignment?

Topics Covered

- Propositional logic
- SAT solvers
- First-order logic
- First-order theories
- Nelson-Oppen theory combination
- SMT solvers

This Time

- Turing machine

Alan Turing [1912-1954]

- One of the founding fathers of CS
- His computer model –the Turing Machine– was inspiration/premonition of the electronic computer that came two decades later
- Was instrumental in cracking the Nazi Enigma cryptosystem in World War WII
- Invented the “Turing Test” used in AI
- Legacy
  - The Turing Award – pre-eminent award in theoretical computer science

A Thinking Machine [1936] I

- First Goal of Turing's Machine
  - A model that can compute anything that a human can compute
  - Before invention of electronic computers the term "computer" actually referred to a *person* who's line of work is to calculate numerical quantities!
- As this is a philosophical endeavor, it can't really be proved
- Turing's Thesis
  - Any "algorithm" can be carried out by one of his machines

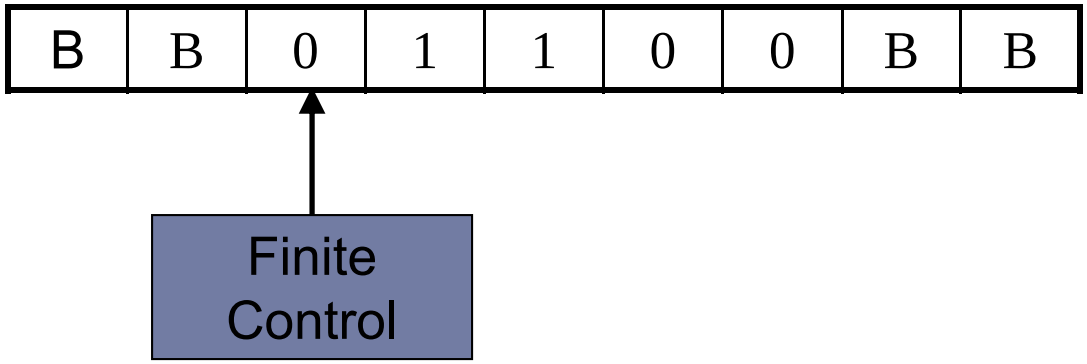
A Thinking Machine [1936] II

- Second Goal of Turing's Machine
  - A mathematical model that's so simple, that can actually prove interesting results.
  - Turing eyed Hilbert's 10^{th} problem, as well as a computational analog of Gödel's Incompleteness Theorem in Logic.
- Theory aside, Turing's programs for cracking the Enigma cryptosystem prove that he really was a true programmer/hacker
  - Turing's machine is actually easily programmable, if you really get into it.
  - Not practically useful though since it is very low-level…

A Thinking Machine [1936] III

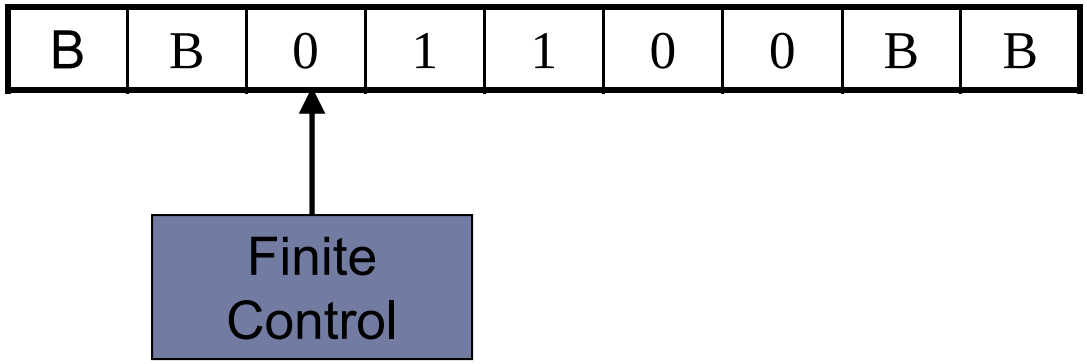
- Imagine a super-organized, obsessive-compulsive human “computer”.
- The computer wants to avoid mistakes so everything written down is completely specified one letter/number at a time.
- The computer follows a finite set of rules which are referred to every time another symbol is written down.
  - Rules are such that at any given time, only one rule is active so no ambiguity can arise.
  - Each rule activates another rule depending on what letter/number is currently read

Turing Machine I



- Two-way, infinite tape, broken into cells, each containing one symbol
- Two-way, read/write tape head
- Finite control, i.e., a program, containing the position of the read head, current symbol being scanned, and the current state.

Turing Machine II



- An input string is placed on the tape, padded to the left and right infinitely with blanks, read/write head is positioned at the left end of input string.
- In one move, depending on the current state and the current symbol being scanned, the TM
  1. changes state,
  2. prints a symbol over the cell being scanned, and
  3. moves its tape head one cell left or right.

A Thinking Machine [1936] IV

- Back then it was hard to believe that ***any*** algorithm could be carried out on such a simple device.
  - Today, it is much easier to believe, especially if you've programmed in assembly ☺
- More believable Church's lambda-calculus paradigm (on which lisp programming is based) proved equivalent to Turing machine

Formal Definition

- A Turing machine is defined as
  TM = <S, T, s_0, δ, H>
  where:
  - S is a set of TM states
  - T is a set of tape symbols
  - s_0 is the start state
  - H $\subset$ S is a set of halting states
  - δ : S x T $\rightarrow$ S x T x {L,R} is the transition function

Configuration

- A TM's next action is completely determined by current state and symbol read, so we can determine future actions if we know:
  1. current state
  2. current tape contents
  3. current position of TM's reading "head"
- Handy notation lists all of these in a single string:
  - A symbol representing current state is sandwiched between content of tape to left of head, and content of tape to right (including tape head)
  - The part of tape which is blank (to infinity) is ignored.

Computation

- Configuration is a triple (q, w, u)
  - state
  - tape content (string)
  - position of the head
- Configuration (q,w,u) yields (q’,w’,u’) in one step if we can transition from (q,w,u) to (q’,w’,u’)
- Execution: sequence of configurations, starting with the initial configuration, and i-th configuration yields the next one in one step

Example I

- Given a string of 1s on a tape (followed by an infinite number of 0s), add one more 1 at the end of the string:

  #111100000000… ➔ #1111100000000…

Example II

state	symbol	Δ(state, symbol)
S_0	b	(halt, a, stop)
S_0	a	(S_1 , a, right)
S_1	b	(halt, b, stop)
S_1	a	(S_0 , a, right)

Input = “aaaabb”
What is the output for this input?

Example III

- Make a TM that recognizes the language $L = \{ w\#w \mid w \in (0,1)^* \}$.
- That is, we have a language separated by a # symbol with the same string on both sides.

Example III Strategy

1. Scan the input to make sure it contains a single # symbol. If not, reject.
2. Starting with the leftmost symbol, remember it and write an X into its cell. Move to the right, skipping over any 0's or 1's until we reach a #. Continue scanning to the first non-# symbol. If this symbol matches the original leftmost symbol, then write a # into the cell. Otherwise, reject.
3. Move the head back to the leftmost symbol that is not X.
4. If this symbol is not #, then repeat at step 2. Otherwise, scan to the right. If all symbols are # until we hit a blank, then accept. Otherwise, reject.

Example IV

- Binary addition

Turing's Thesis I

Any mathematical problem solving that can be described by a mechanical procedure (algorithm) can be modeled by a Turing machine.

- All computers today perform only mechanical problem solving.
- They are no more expressive than a Turing machine.
  - If a TM can't solve it, neither can a computer

Turing's Thesis II

- Turing's thesis is not a "theorem" – there is no "proof" for the thesis.
- The theorem may be refuted by showing at least one task that is performed by a digital computer which cannot be performed by a Turing machine.
- Many contentions have been made to this end. However, there have still not been any conclusive evidence to refute Turing's thesis.

Next Time

- Decidability
- Complexity