CS 3719 (Theory of Computation and Algorithms) – Lecture 2

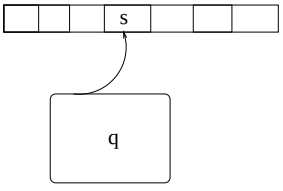
Antonina Kolokolova *

January 9, 2013

Turing Machines

In 1936, Alan Turing gave the first definition of an algorithm, in the form of (what we call today) a Turing machine. He wanted a definition that was simple, clear, and sufficiently general. Although Turing was only interested in defining the notion of "algorithm", his model is also good for defining the notion of "efficient" (polynomial-time) algorithm.

His inspiration likely was watching "human computers" decrypting intercepted German messages at Bletchley park during the war. Such a "computer", usually a lady, would start with a message to be decoded and a stack of blank paper, on which she would write her intermediate calculations. Sometimes she would have to leaf through a stack to find a page with something computed before. The idea here is that each piece of paper can only hold a finite amount of information; moreover, the person doing the computation also has "finite attention span" and can only keep finite amount of information in her memory. Each piece of paper would be modeled with a "tape cell", with the whole tape being the stack of paper (assuming one can always go and get more blank paper). The "states of the mind" of the person doing the calculation is modeled by states. And the "head position on the tape" is the piece of paper currently being used.



A Turing machine consists of an infinite tape and a finite state control. The tape is divided up into squares, each of which holds a symbol from a finite tape alphabet that includes the blank symbol $\not b$. Some definitions give two-way infinite tape; Sipser's book uses tape infinite to the right (so it has a start cell, but no end cell).

*The material in this set of notes came from many sources, in particular "Introduction to Theory of Computation" by Sipser and course notes of U. of Toronto CS 364. Special thank you to Richard Bajona for sharing scribed lecture notes.

The machine has a read/write head that is connected to the control, and that that scans squares on the tape. Depending on the state of the control and symbol scanned, it makes a move, consisting of

- printing a symbol
- moving the head left or right one square
- assuming a new state

The tape will initially be completely blank, except for an input string over the finite input alphabet Σ; the head will initially be pointing to the leftmost symbol of the input.

A Turing machine M is specified by giving the following:

- Σ (a finite input alphabet).
- Γ (a finite tape alphabet). $\Sigma \subseteq \Gamma$. $\not\flat \in \Gamma - \Sigma$.
- Q (a finite set of states). There are 3 special states:
  - q_0 (the initial state)
  - q_{accept} (the state in which M halts and accepts)
  - q_{reject} (the state in which M halts and rejects)
- $\delta : (Q - \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ (the transition function)
  (If $\delta(q, s) = (q', s', h)$, this means that if q is the current state and s is the symbol being scanned, then q' is the new state, s' is the symbol printed, and h is either L or R, corresponding to a left or right move by one square.

So formally, a Turing machine M is defined to be equal to the tuple $\langle \Sigma, \Gamma, Q, \delta, q_0, q_{accept}, q_{reject} \rangle$. Sometimes you see definitions that group together q_{accept} and q_{reject} into "halting states" F, or specify the blank symbol separately.

The Turing machine M works as follows on input string $x \in \Sigma^*$.
Initially x appears on the tape starting from its left end, with infinitely many blanks to the right, and with the head pointing to the leftmost symbol of x. (If x is the empty string, then the tape is completely blank and the head is pointing to the leftmost square.) The control is initially in state q_0.
M moves according to the transition function δ.
M may run forever, but if it halts, then it halts either in state q_{accept} or q_{reject}. (We will mainly be interested in whether M accepts or rejects; there are definitions of what it means

for a Turing machine to output a string: e.g., the output is what is left on the tape when the machine halts)

The Turing machine M works as follows on input string $x \in \Sigma^*$.
Initially x appears on the tape starting from its left end, with infinitely many blanks to the right, and with the head pointing to the leftmost symbol of x. (If x is the empty string, then the tape is completely blank and the head is pointing to the leftmost square.) The control is initially in state q_0.
M moves according to the transition function δ.
M may run forever, but if it halts, then it halts either in state q_{accept} or q_{reject}. (We will mainly be interested in whether M accepts or rejects; there are definitions of what it means for a Turing machine to output a string: e.g., the output is what is left on the tape when the machine halts)

Definition 12. M accepts *a string $x \in \Sigma^*$ if M with input x eventually halts in state q_{accept}. We write $\mathcal{L}(M) = \{x \in \Sigma^* | M \text{ accepts } x\}$, and we refer to $\mathcal{L}(M)$ as the language accepted by M.*

(Notice that we are abusing notation, since we refer both to a Turing machine accepting a string, and to a Turing machine accepting a language, namely the set of strings it accepts.)

Example 1. PARITY= the set of all binary strings that have an even number of 1s $= \{x | x \in \{0,1\}^*, x \text{ has even number of occurrences of symbol } 1\}$.

A very simple Turing machine recognizes the set of such strings in time linear in the number of symbols in the strings. For that, it moves once through the string from left to right, at each point remembering whether the number of 1s it has seen so far is even or odd. If by the time it reaches a blank symbol it has seen an even number of 1s, the Turing machine accepts, and if it has seen an odd number of 1s, it rejects.

State q	*Symbol s*	*Action $\delta(q,s)$*
q_0	0	$(q_0, \not b, R)$
q_0	1	$(q_1, \not b, R)$
q_0	$\not b$	$(q_{accept}, \not b, L)$
q_1	0	$(q_1, \not b, R)$
q_1	1	$(q_0, \not b, R)$
q_1	$\not b$	$(q_{reject}, \not b, L)$

Definition 13. *Let us define more formally what is a* computation *of a Turing machine. To describe a computation of a Turing machine we need to specify, at every point in time, its state, head position, and (non-blank) content of the tape. This information we will call a* configuration *of a Turing machine. For convenience, we will write a configuration as the string that is the sequence of non-blank symbols of the tape (assuming there are no blanks between non-blanks), with the symbol for the current state in a position right before the cell it points to. For example, the starting configuration of a Turing machine on input 0011 is $q_0 0011$; after a transition $(q_0, 0) \to (q_5, 1, R)$ the machine goes to the configuration $1q_5011$.*

Now, a Turing machine M accepts *a string w if there is a sequence of configurations starting*

from $q_0 w$ and ending in a configuration containing q_{accept}, with every configuration in the sequence resulting from a previous one by a transition in δ of M.