CIS 511: Theory of Computation **Jan 24, 2017**

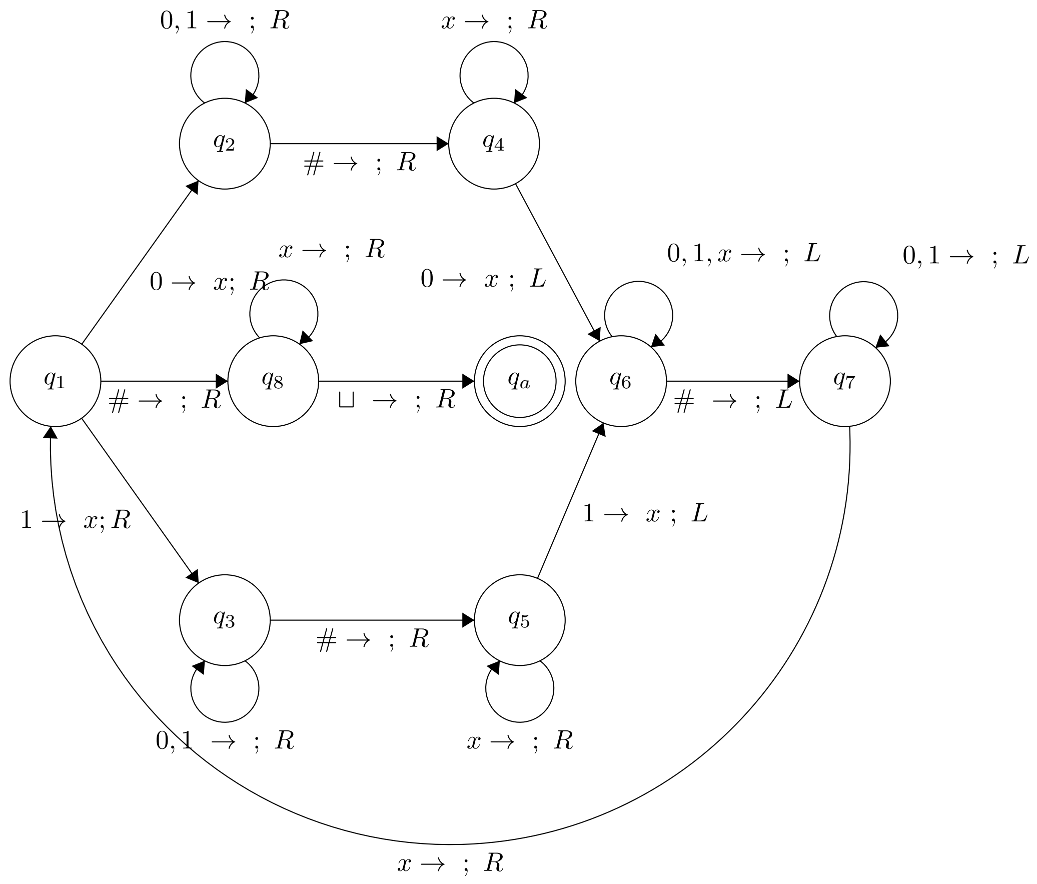
Lecture 4: More Turing Machines

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Turing Machines

Let $B = \{w\#w | w \in \Sigma^*\}$. Here is a TM to accept this language:



Variations on the Turing Machine

How robust is the TM? Turns out, very!

Definition 4.1 *Define a **stay**-TM as one with move commands expanded to $\{L, R, S\}$, where S denotes no-move.*

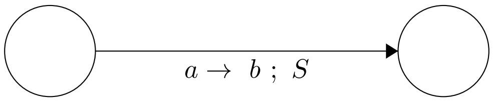
Claim 4.2 *A stay-TM is equal in power to the regular TM.*

Proof:

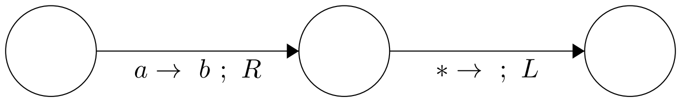
A regular TM is a stay-TM which does not use the S option, so clearly anything a TM can do, a stay-TM can as well.

Now, to see that a stay-TM does not recognize or decide any more languages than a regular TM, let M_s be a stay TM for some language. To convert it into a regular TM M', perform the following procedure.

Suppose M has a transition that uses the S option, of the form



for some $a, b \in \Gamma$. Replace this transition with an intermediate state and two transitions that read a character, perform the write, then move right, then move left without writing anything, as follows, where $*$ represents any character in Γ.



This pair of transitions performs the same modification to the state of M' as the original does to M, therefore stay-TMs are no more powerful than regular TMs.

■

Definition 4.3 *A **multi-tape TM** is a Turing machine with a finite number (k) of tapes. Each transition in δ reads the character under each head and writes and moves independently on each tape. Formally, δ is now a function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$.*

Claim 4.4 *A multi-tape TM is no more or less powerful than a regular TM.*

Proof:

One direction is trivial. A regular TM is a multi-tape TM with $k = 1$.

To see that a multi-tape TM is not more powerful than a regular TM, let M be a TM with k tapes. We will construct a single-tape TM S that simulates M.

Assume $\# \notin \Gamma_M$, which will be our delimiter symbol in S.

We will also expand Γ_S to include $a, \dot{a}$, for each $a \in \Gamma_M$.

Let S have a tape initialized to the concatenation of the contents of each of M's k tapes, separated by $\#$.

We will use our dotted symbols to track where the head of each tape is, and since the head can only be in one place, there is exactly one dotted symbol in each of the k segments.

Let S sweep to the right, entering a state corresponding to the k-tuple of dotted symbols it reads and the current state of M. Then, S needs to execute the proper transition in M. We now know that the state space of S is at least $|Q_M| \times |\Gamma_M|^k$. On the way back to the left end, S should execute M's transitions.

We now have to deal with the issue of S overwriting the bounds of the tape segments. We give S a routine that pushes everything to the right in order to make space.

This machine simulates M, completing the proof.

∎

Observe that if M has taken at most T steps, then it uses at most T cells on each of its tapes. Therefore, the maximum number of cells needed on S's single tape is $O(kT)$. We face no worse than quadratic slowdown, as each of the T steps of M take no more than T steps to simulate in S, leading to a slowdown on the order of T^2.

Is it possible for a multi-tape TM to have a truly quadratic speedup?

Claim 4.5 *Let $L = \{w\#w^R | w \in \Sigma^2\}$. This language gains a quadratic speedup on a 2-tape machine versus the naive implementation on a single tape machine.*

Proof:

First, copy the input to the second tape. Then, the head on the second tape can simultaneously match characters from right to left with the first tape going left to right.

How many steps does this take? If the input is of size m, we can copy the input in $O(m)$, move the first head back to start in $O(m)$, and step through the string in $O(m)$.

For the single tape machine, we need $O(m)$ steps to move back and forth to check each pair of characters, of which there are $\frac{m}{2}$, for $O(m)$ comparisons. Therefore, the 2-tape machine decides the language in $O(m)$ while the single tape needs $O(m^2)$.

∎

Can we do better by being clever? No! If we try to batch the symbols and compress, we lose information which might lead to incorrect acceptance/rejection.