CSE 431 Theory of Computation **Spring 2014**

Lecture 2: April 3

Lecturer: James R. Lee *Scribe: Riley Klingler and Alexa Rust*

2.1 Finite State Machines

The simplest model of computation is a *finite state machine* or a *finite automoton*. Informally, it is a set of states with transitions between the states based on an input string from some fixed alphabet. There is a start state and a set of accept states. The machine accepts a string if the machine ends in an accept state after processing the string.

The simplicity of finite state machines limits their computational power. It can be proven that the languages which can be recognized by a finite state machine are exactly the regular languages, languages which can be expressed using a regular expression.

Example This language $bab(a|b)^*|ab^*a(a|b)*$ is accepted by a finite state machine.

Example This language $L = \{a^n b^n : n \geq 0\}$ cannot be accepted by a finite state machine.

2.2 Pushdown Automata

A *pushdown automaton* is a finite state machine with the addition of a stack. This provides more computational power, because the stack allows an unlimited amount of memory. Thus pushdown automata can recognize more languages than finite state machines, for example the language $L = \{a^n b^n : n \geq 0\}$. Pushdown automata like finite state machines are still a weak form of computation. For example the language $L' = \{a^n b^n c^n : n \geq 0\}$ cannot be accepted by a pushdown automaton with only one stack (it can be recognized by an automaton with two stacks).

2.3 Turing Machines

Turing machines represent a universal model of computation. Informally a Turing Machine (TM) consists of a finite control, an infinitely long tape (possibly in both directions), a read/write head that can read and write symbols on the tape and move left and right.

Church-Turing Thesis 2.1 *Any reasonable model of computation is no stronger than a Turing machine.*

Since "reasonable" does not have a mathematical definition the Church-Turing thesis is not a statement that can be proved or disproved.

Definition 2.2 *A Turing machine is a 7-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ *such that*
1. Q *is is a finite set of states,*
2. Σ *is the input alphabet where the bank symbol* $\sqcup \notin \Sigma$*,*
3. Γ *is the tape alphabet where* $\Sigma \subset \Gamma$ *and* $\sqcup \in \Gamma$*,*
4. δ *is the transition function where* $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$*,*
5. $q_0 \in Q$ *is the start state,*
6. q_{accept} *is the accept state,*
7. q_{reject} *is the reject state.*

We assume that if the head of the TM is at the left most tape entry, attempting to move left well keep it in the same location.

Definition 2.3 *If* M *is a TM over* Σ*, then* $\mathcal{L}(M) = \{x \in \Sigma^* | M \text{ accepts } x\}$ *is the language of* M*.*

Definition 2.4 *A language* $\mathcal{L}$ *is Turing-recognizable if* $\mathcal{L}$ *is the language accepted by some TM.*

Definition 2.5 *A TM is a decider if it halts on every input.*

Definition 2.6 *A language* $\mathcal{L}$ *is Turing-decidable if it is accepted by a decider.*

Example We give an example of a TM M. The TM M decides the language $A = \{0^{2^n} | n \geq 0\}$, i.e. strings of 0's that are of length a power of 2. The TM works by crossing out 0's on the tape to simulate repeated division by 2. We do this by crossing out every other 0 on the tape (ignoring already crossed out 0's) starting with the second 0 on the tape. We accept if there is one 0 left on the tape, we reject if at any time there is a non-crossed out 0 at the end of the string.

We give a formal definition of this TM: $M = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$, where $Q = \{q1, q2, q3, q4, q5, q_{\text{accept}}, q_{\text{reject}}\}$, $\Sigma = \{0\}$, $\Gamma = \{\sqcup, 0, x\}$, q_1 is the start state, and the transition function δ is given by Figure 2.3.

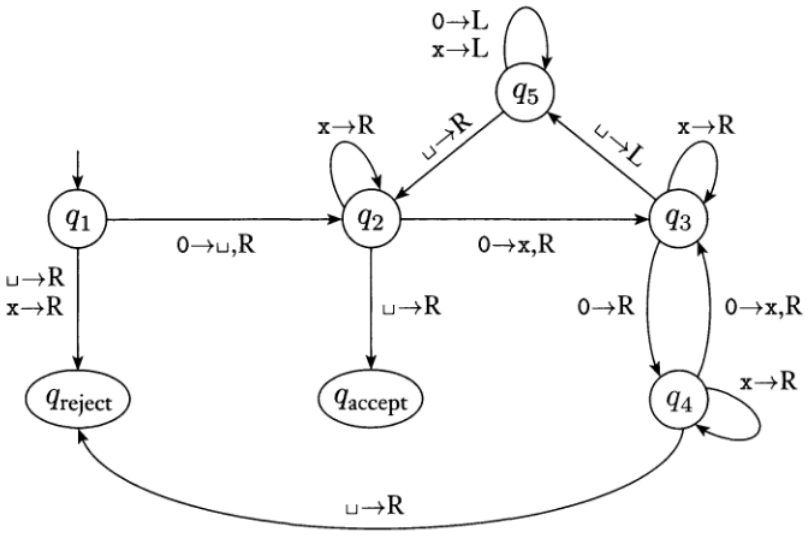

Figure 2.1: Transition function for TM that decides language $A = \{0^{2^n} | n \geq 0\}$.

We write '$x_1x_2\ldots x_mq_ix_{m+1}x_{m+2}\ldots x_n$' to denote that $x_1, x_2, \ldots, x_m, x_{m+1}, x_{m+2}, \ldots, x_n$ are the first n symbols on the tape where x_1 is the left most symbol and all symbols after x_n are blanks (n may be 0), the TM is in state q_i, and the head is looking at symbol $x_m + 1$.

On input '0000', M starts in state q_1 with 0000 written on the tape with blanks following on the right and the head looking at the leftmost 0. Hence, the initial configuration of the TM is '$q_1$0000'. We demonstrate the TM M's computation on the input '0000':

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3$
$\sqcup x 0 q_5 x$
$\sqcup x q_5 0 x$
$\sqcup q_5 x 0 x$
$q_5 \sqcup x 0 x$
$\sqcup q_2 x 0 x$
$\sqcup x q_2 0 x$
$\sqcup x x q_3 x$
$\sqcup x x x q_3$
$\sqcup x x q_5 x$
$\sqcup x q_5 x x$
$\sqcup q_5 x x x$
$q_5 \sqcup x x x$
$\sqcup q_2 x x x$
$\sqcup x q_2 x x$
$\sqcup x x q_2 x$
$\sqcup x x x q_2$
$\sqcup x x x \sqcup q_{\text{accept}}$

The input '0000' is accepted by TM M.

We note that we overwrite the first symbol on the tape with a blank so that when moving back to the left from the end of the input to the start of the input we know we have found the start when we see a blank.