Alex Hubers
CSC:341 Spring 2024

Lecture Notes #6A

#1 Motivation

Turing machines originate with Alan Turing in answer to a handful of questions in formal logic—in particular, the halting problem and the decision problem. (The optional reading by Erickson has a lot more on this.)

I should also mention that this course does not present its material with much respect for chronological accuracy: Turing machines much predate finite automata. You can think of finite automata more as generalized restrictions of TMs. That being said, let's continue with this chronology as a convenient narrative.

Limitations of finite automata?

1. restricted to regular / context-free languages
2. finite memory
3. Only binary output (accepts / does not accept)
4. What else?

#2 Turing machines

Let's start with the formal definition.

Note that there are a dozen plus one different (but ultimately equivalent) ways of defining TMs. Here is Sipser's.

Definition 1 (Turing machine)**.** *A Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ where Q, Σ, Γ are all finite sets and*

1. *Q is the set of states,*
2. *Σ is the input alphabet s.t. $\square \notin \Sigma$,*
3. *Γ is the tape alphabet where $\square \in \Gamma$ and $\Sigma \subset \Gamma$,*
4. *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,*
5. *$q_0 \in Q$ is the start state,*
6. *$q_A \in Q$ is the accept state,*
7. *$q_R \in Q$ is the reject state, where $q_A \neq q_R$.*

N.b. Sipser uses (i) a different blank symbol, (ii) q_{accept} instead of q_A, and (iii) q_{reject} instead of q_R.

#3 Running the machine

It will be hard to continue with more definitions until you have an intuition of what the hell it is we are describing. So let's first try a very simple example.

Let's first think about how such a machine works.

1. Start with a "tape"—an infinite array of cells with each cell marked with the blank symbol $\square$.
2. Write the "input" (with length n) onto the first n cells of the tape.
3. Now we begin: let q_0 be the start state, Place the TM "head" at the leftmost position, and "read" the first cell.
4. Now, use δ to
   (a) transition the head,
   (b) write to the tape,
   (c) advance the states.
5. Repeat until we reach state q_A or q_R. (Or, run forever).

#4 Example

Consider a Turing machine that recognizes the following language.

$$A = \{0^n \mid n \geq 0\}$$

In class, let's think on what A should "do" before defining it formally. Consider input 00000. We start at the leftmost character. If it's a zero, move the head 'right and check if the next character is a zero. Repeat until we see the blank symbol $\square$. If this happens, accept the string. Otherwise, on any other input, reject.

Now, let's give a formal definition of the TM that recognizes A.

1. $Q = \{q_0, q_{seek}, q_A, q_R\}$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{0, 1, \square\}$
4. Define δ by the following equations:

$$\delta(q_0, 0) = (q_{seek}, 0, R) \qquad \text{Start with 0? Look for more.} \tag{1}$$
$$\delta(q_0, \square) = (q_R, \square, R) \qquad \text{Or, if the string is empty, reject.} \tag{2}$$
$$\delta(q_0, 1) = (q_R, 1, R) \qquad \text{If the string starts with 1, reject.} \tag{3}$$
$$\delta(q_{seek}, 0) = (q_{seek}, 0, R) \qquad \text{Keep looking for zeros.} \tag{4}$$
$$\delta(q_{seek}, 1) = (q_R, 1, R) \qquad \text{If we find a 1, reject.} \tag{5}$$
$$\delta(q_{seek}, \square) = (q_A, \square, R) \qquad \text{If we reach the end of the input, accept.} \tag{6}$$

#5 More Definitions

We need the next set of definitions to begin to discuss important properties of Turing machines.

1. A configuration of a Turing machine is represented by a triple $(q, x, i) \in Q \times \Gamma^* \times \mathbb{N}$ indicating that
   (a) the machine's internal state is q,
   (b) the tape contains the string x followed by an infinite sequence of blanks, and
   (c) the head is located at position i.

   A Sipser configuration is the string uqv where (1) the state is q, (2) the string is uv, and (3) the head is located at the position immediately after prefix u.

2. We write $(p, x, i) \Rightarrow (q, y, j)$ to indicate that the Turing machine M transitions from the first configuration to the second in one step. We say that the first yields the second. We can write $(p, x, i) \Rightarrow^* (q, y, j)$ to indicate that the first yields the second in 1 or more steps.

3. Let $A \subseteq \Sigma^*$ be a language.
   - We say that M recognizes or accepts A if and only if M accepts every string in A but nothing else.
   - We say that M decides A if and only if M accepts every string in A and rejects every string in $\Sigma^* \setminus A$. Equivalently, M decides A iff M recognizes A and halts on all inputs.

#6 Formal, implementation, and high-level descriptions of TMs

A valid question: in what level of detail do I specify Turing machines in this course? Sipser makes the following distinction.

1. A formal description tells me (at minimum) the components of (Q, Γ, q_0, δ). The description we gave above for the language that decides $\{0^n \mid n \geq 0\}$ was a formal description.

2. An implementation-level description uses English prose to describe the way that the TM moves its head and the way it stores data on its tape, but may omit details of state or transition functions.

3. A higher-level description uses English prose to describe the behavior of the TM, ignoring how the machine manages its tape or head.

I will try to be clear which you should use. In general, the hope is to "graduate" slowly from formal descriptions to higher-level. Once you show to me that you can write formal descriptions, it becomes unnecessary to do so, and so forth. Ultimately, what I ask of you is that your description of the TM is *clear*.