CS 530: Theory of Computation

Based on Sipser (second edition): The Church-Turing Thesis

Definition 1 (Definition 3.3, page 140; minor modifications) *A **Turing machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and*

1. *Q is the set of states,*
2. *Σ is the input alphabet not containing the special **blank** symbol $\_$,*
3. *Γ is the tape alphabet, where $\_ \in \Gamma$ and $\Sigma \subseteq \Gamma$,*
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. *$q_{accept} \in Q$ is the accept state, and*
7. *$q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.*

As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a **configuration** of the Turing machine. Configurations often are represented in a special way. For a state q and two strings u and $v \neq \epsilon$ over the tape alphabet Γ we write uqv for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v. The tape contains only blanks following the last symbol of v. Also, v's last symbol is blank only if $|v| = 1$.

Say that configuration C_1 **yields** configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step. Suppose that we have a, b, and c in Γ, as well as u and v in Γ^* and states q_i and q_j. In that case uaq_ibv and uq_jacv are two configurations. Say that

$$uaq_ibv \text{ yields } uq_jacv$$

if in the transition function $\delta(q_i, b) = (q_j, c, L)$. That handles the case where the Turing machine moves leftward. For a rightward move, say that

$$uq_ibv \text{ yields } ucq_jv$$

if $\delta(q_i, b) = (q_j, c, R)$.

There are three exceptions:

1. if $\delta(q_i, b) = (q_j, c, L)$, then configuration q_ibv yields q_jcv (the machine cannot move left from the left-most position of the tape),
2. if $\delta(q_i, b) = (q_j, c, R)$, then configuration uq_ib yields $ucq_j\_$ (a blank is inserted at the end of the configuration if the head moves right from a symbol that is followed only by blanks).
3. if $\delta(q_i, b) = (q_j, \_, L)$, then configuration uaq_ib yields uq_ja (we erase from a configuration $u'qv'$ the blanks at the end of v').

The **start configuration** of M on input w is the configuration q_0w, which indicates that the machine is in the start state q_0 with its head at the leftmost position on the tape. In an **accepting configuration** the state of the configuration is q_{accept}. In an **rejecting configuration** the state of the configuration is q_{reject}. Accepting and rejecting configurations are **halting configurations** and accordingly do not yield further configurations. A Turing machine M **accepts** input w if a sequence of configuration $C_1, C_2, ..., C_k$ exists where

1. C_1 is the start configuration of M on input w,
2. each C_i yields C_{i+1}, and
3. C_k is an accepting configuration.

The collection of strings that M accepts is **the language of** M, denoted $L(M)$.

Definition 2 (Definition 3.5, page 142) *Call a language* ***Turing-recognizable*** *if some Turing machine recognizes it*[1]

[1] There is a purely algebraic definition of a *recursively enumerable* set of natural numbers. This definition corresponds (based on proofs we do not have time to do) to Turing-recognizable languages.

When we start a TM on an input, three outcomes are possible. The machine may *accept*, *reject*, or *loop*. By **loop** we mean that the machine simply does not halt. It is not necessarily repeating the same steps in the same way forever as the connotation of looping may suggest. Looping may entail any simple or complex behavior that never leads to a halting state.

Sometime, we prefer Turing machines that halt on all inputs; such machines never loop. These machines are called **deciders** because they always make a decision to accept or reject. A decider that recognizes some language also is said to **decide** that language.

Definition 3 (Definition 3.6, page 142) *Call a language* ***Turing-decidable*** *or simply* ***decidable*** *if some Turing machine decides it*[2]*.*

Definition 4 *A* ***multitape Turing machine*** *is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape* 1*, and the others start out blank. The transition function is changed to allow for reading, writing, and moving the heads on all the tapes simultaneously. Formally, it is*

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$

where k *is the number of tapes. The expression*

$$\delta(q_i, a_1, ..., a_k) = (q_j, b_1, ..., b_k, L, R, ..., L)$$

means that, if the machine is in state q_i *and heads* 1 *through* k *are reading symbols* a_1 *through* a_k*, the machine goes to state* q_j*, writes symbols* b_1 *through* b_k*, and moves each head to the left or right as specified.*

Theorem 5 (Theorem 3.13, page 149) *Every multitape Turing machine has an equivalent single-tape Turing machine.*

Corollary 6 (Corollary 3.15, page 150) *A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.*

[2]The purely algebraic equivalent (again, no proof) is a *recursive* set of natural numbers.

Corollary 7 (implicit in the textbook) *A language is decidable if and only if some multitape Turing machine decides it.*

Definition 8 *A* ***nondeterministic Turing machine*** *is defined just as a deterministic Turing machine, except for the transition function, which has the form:*

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

Theorem 9 (Theorem 3.16, page 150) *Every nondeterministic Turing machine has an equivalent deterministic Turing machine.*

Corollary 10 (Corollary 3.18, page 152) *A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it.*

Theorem 11 (Corollary 3.19, page 152) *A language is decidable if and only if some nondeterministic Turing machine decides it.*

Definition 12 *An* ***enumerator*** *E is defined just as a k-tape deterministic Turing machine, except it also has a (non-final) special state q_{print}, and one of the tapes is called the* print tape. *E* prints *string $w \in \Sigma^*$ if a sequence of configurations exist, starting with all tapes blank, such that E enters state q_{print} with w on the print state. The collection of strings that E prints is* ***the language of enumerator*** *E, denoted $L(E)$.*

Theorem 13 (Theorem 3.21, page 153) *A language is Turing-recognizable if and only if some enumerator enumerates it.*

A **polynomial** is a sum of terms, where each **term** is a product of certain variables and a constant called a **coefficient**. A **root** of a polynomial is an assignment of values to its variables so that the value of the polynomial is 0.

Recall that a graph is **connected** if every node can be reached from every other node by travelling along the edges of the graph.