8 Turing Machines

The question of what computers can do/solve, or equivalently, what languages can be defined/recognized by any computational device whatsoever.

8.1 Problems that computers cannot solve

- Two types of problems: “Solve this” and “decide this”.
- Decision problems (the “decide this” type) have a yes/no solution. They are just as hard as their “solve this” version in the sense of dealing with important questions in complexity theory.
- A problem is said to be unsolvable/undecidable if it cannot be solved/decided using a computer.
- Recall that a decision problem is really membership of a string in a language. For example, the problem of primality testing is actually the language of all prime numbers in binary representation.
- The number of problems/languages over an alphabet with more than one symbol is uncountably infinite. However, the number of programs that a computer may use to solve problems is countably infinite. Therefore, there are more problems than there are programs. Thus, there must be some undecidable problems.
- An undecidable problem (the halting problem):
  - Input: Any program $P$ and any input $I$ to the program;
  - Output: “Yes” if $P$ terminates on $I$ and “No” otherwise.

8.2 The Turing machine

*Reading: Sipser 3.1 (pp. 137-147)*

- A Turing machine includes a control unit, a read-write head, and a one-way infinite tape.
- TM $M = (Q, Σ, Γ, δ, q_0, q_{\text{accept}}, q_{\text{reject}})$, where
  - $Q$: The finite set of states for the control unit.
  - $Σ$: An alphabet of input symbols, not containing the “blank symbol” $\sqcup$ (or $B$).
  - $Γ$: The complete set of tape symbols. $Σ ∪ \{\sqcup\} ⊂ Γ$.
  - $δ$: The transition function from $Q × Γ$ to $Q × Γ × D$, where $D = \{L, R\}$.
  - $q_0$: The start state.
  - $q_{\text{accept}}$: The accept state.
  - $q_{\text{reject}}$: The reject state.
- Configuration: $X_1 \cdots X_{i-1}qX_i \cdots X_n$ is a configuration (snapshot) of the TM in which $q$ is the current state, the tape content is $X_1 \cdots X_n$, and the head is scanning $X_i$.
  - If $δ(q, X_i) = (p, Y, L)$, then $X_1 \cdots X_{i-1}qX_i \cdots X_n \vdash X_1 \cdots X_{i-2}pX_{i-1}YX_{i+1} \cdots X_n$.
  - If $δ(q, X_i) = (p, Y, R)$, then $X_1 \cdots X_{i-1}qX_i \cdots X_n \vdash X_1 \cdots X_{i-1}YpX_{i+1} \cdots X_n$.
- Starting configuration $q_0w$, accepting configuration $u_{\text{accept}}$, and rejecting configuration $u_{\text{reject}}$, where the latter two are called the halting configurations.
- Language of a Turing machine $M$ (or language recognized by $M$) is $L(M) = \{w ∈ Σ^* | q_0w \vdash αq_{\text{accept}}β$ for $α, β \in Γ^*\}$.
- For any given input, a TM has three possible outcomes: accept, reject, and loop. Accept and reject mean that the TM halts on the given input, but loop means that the TM does not halt on the input.
• A language $A$ is Turing-recognizable (or recursively enumerable) if there is a TM $M$ such that $A = L(M)$. In other words, $\forall w \in A$, $M$ accepts $w$ by entering $q_{\text{accept}}$. However, $\forall w \notin A$, $M$ may reject or loop.

• A language $A$ is Turing-decidable (or decidable, or recursive) if there is a TM $M$ such as $A = L(M)$ and $M$ halts on all inputs. In other words, $\forall w \in A$, $M$ accepts $w$ and $\forall w \notin A$, $M$ rejects $w$. Such TM’s are a good model for algorithms.

**Example:** Give a TM $M$ with $L(M) = \{0^n1^n | n \geq 0\}$.

**Example (Sipser p. 143):** Give a TM $M$ that decides $A = \{0^n | n \geq 0\}$.

### 8.3 Properties of TDLs and TRLs

**Theorem:** A Turing-decidable language is also Turing-recognizable, but not vice versa.

**Theorem:** $A$ and $\overline{A}$:

- If $A$ is Turing-decidable, so is $\overline{A}$.
- If $A$ and $\overline{A}$ are both Turing-recognizable, then $A$ is Turing-decidable.
- For any $A$ and $\overline{A}$, we have one of the following possibilities: (1) Both are Turing-decidable; (2) Neither is Turing-recognizable; (3) One is Turing-recognizable but not decidable, the other is not Turing-recognizable.

Some additional closure properties: TRLs and TDLs both are closed under

- Union
- Intersection
- Concatenation
- Star

### 8.4 Variations of TM’s

**Reading:** Sipser 3.2 (pp. 148-159)

- TM with multi-tapes (and multi-cursors) $(\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k)$.
- TM with multi-strings (and multi-cursors).
- TM with multi-cursors.
- TM with multi-tracks.
- TM with two-way infinite tape.
- TM with multi-dimensional tape.
- Nondeterministic TM’s $(\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times D})$.

**Theorem:** The equivalent computing power of the above TM’s: For any language $L$, if $L = L(M_1)$ for some TM $M_1$ with multi-tapes, multi-strings, multi-cursors, multi-tracks, two-way infinite tape, multi-dimensional tape, or nonde terminism, then $L = L(M_2)$ for some basic TM $M_2$.

**Theorem:** The equivalent computing speed of the above TM’s except for nondeterministic TM’s: For any language $L$, if $L = L(M_1)$ for some TM $M_1$ with multi-tapes, multi-strings, multi-cursors, multi-tracks, two-way infinite tape, or multi-dimensional tape in a polynomial number of steps, then $L = L(M_2)$ for some basic TM $M_2$ in a polynomial number of steps (with a higher degree). Or in other words, all reasonable models of computation can simulate each other with only a polynomial loss of efficiency.

Note: The speed-up of a nondeterministic TM vs. a basic TM is exponential.

**Example:** True of false: (1) A language is Turing-recognizable if and only if some nondeterministic Turing machine recognize it. (2) A language is Turing-decidable if and only if some nondeterministic Turing machine decides it.

**The Church-Turing Thesis:** Any reasonable attempt to model mathematically algorithms and their time performance is bound to end up with a model of computation and associated time cost that is equivalent to Turing machines within a polynomial. (The power of TM.)