13. Turing Machines/Extensions of

Ways to extend Turing machines:

1. Allow two-way infinite tape.

   Figure 4-18 here

2. Allow multiple tapes.

   Figure 4-23 here

3. Allow multiple heads.

4. Allow two-dimensional tape.

5. Combination of some or all of the above.

Theorem 1 *The operation of a machine allowing some or all of the above extensions can be simulated by a standard Turing machine. The extensions do not give us machines more powerful than Turing machines.*

Note: The extensions are helpful in designing Turing machines to solve particular problems.

Two-way infinite tape:

- Let $M_1 = (K_1, \Sigma_1, \delta_1, s_1)$ be a two-way infinite tape Turing machine.
- It can never hang.
- This machine can be simulated by a standard Turing machine M_2 whose tape is divided into two tracks:

  Figure 4-19

  - Upper track represents the "non-negative" part of the tape of M_1.
  - Lower track represents "negative" part of the tape of M_1.

Two-way infinite tape: Algebraically,

- $K_2 = K_1 \times \{1, 2\}$.
  - $< p, 1 >$ means M_1 in state p and its head in the "non-negative" part of its tape.
  - $< p, 2 >$ means M_1 in state p and its head in the "negative" part of its tape.
- $\Sigma_2 = \{\$\} \cup \Sigma_1 \times \Sigma_1$.
  - (a, b) being the symbol at the $i + 1$th position of the tape of M_2 means
    * a is the symbol at position $i + 1$ of the tape of M_1 and
    * b is the symbol at position $-i$ of the tape of M_1.
  - \$ is a special symbol always at the left most square of the tape of M_2.

- The transition rules for M_2:
  - (Simulate M_1 when its head is at the "non-negative" part of its tape.) If $\delta_1(q, a_1) = (p, b)$,

$$\delta_2(< q, 1 >, (a_1, a_2)) = \begin{cases} (< p, 1 >, L) & \text{if } b = L \\ (< p, 1 >, R) & \text{if } b = R \\ (< p, 1 >, (b, a_2) & \text{otherwise} \end{cases}$$

  - (Simulate M_1 when its head is at the "negative" part of its tape.) If $\delta_1(q, a_2) = (p, b)$,

$$\delta_2(< q, 2 >, (a_1, a_2)) = \begin{cases} (< p, 2 >, R) & \text{if } b = L \\ (< p, 2 >, L) & \text{if } b = R \\ (< p, 2 >, (a_1, b) & \text{otherwise} \end{cases}$$

  - (Change tracks.)

$$\begin{aligned} \delta_2(< q, 1 >, \$) &= (< q, 2 >, R) \\ \delta_2(< q, 2 >, \$) &= (< q, 1 >, R) \end{aligned}$$

k-tape Turing machines:

- There is a read/write head for each tape.
- In each step, it reads the symbols pointed to by all its heads.
- Depending on its current state and the current symbols, it decides for each head whether to
  - rewrite the square.
  - Move left/right.

Such a machine M_1 can be simulated by using a standard Turing machine M_2 whose tape is divided into $2k$ tracks. For each $i = 1$ to k,

- The $2i - 1$th track records the contents of the ith tape of M_1, and
- The $2i$th track records the position of the ith head.

A transition in M_1 is simulated by a series of transitions in M_2 (example).

Figure 4-26 here

The language

$L = \{w \in \{a, b, c\}^* :$ w contains equal number of a's, b's, and c's

can easily be decided by a 4-tape machine.

- Assume input inscribed on tape 1 and all other tapes are blank.
- Scan tape 1 from right to left, move a's to tape 2, b's to tape 3, and c's to tape 4, with one # at the leftmost square to mark the left end of the strings.
- Scan tapes 2, 3, and 4 from left to right in parallel.
- If all head reach # at the same time, return Y. Otherwise, return N.

One more extension: Allow nondeterministic transitions.

- In a Turing machine $M = (K, \Sigma, \delta, s)$, δ is a mapping

$$\delta : K \times \Sigma \to (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$$

- A **nondeterministic Turing machine** is a quadruple $M = (K, \Sigma, \Delta, s)$, where Δ is a subset of

$$K \times \Sigma \times (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$$

- A nondeterministic Turing machine M **accepts** a string w if there **exists** one way of computation in which M halts on w.

Example:

$$L = \{w \in \{a, b\}^* : w \text{ contains substring } abaab\}$$

The following nondeterministic Turing machine accepts L by "guessing" where the substring starts.

Figure 4-27 here

Theorem 2 *If M_1 is a nondeterministic Turing machine, we can construct a standard Turing machine such that for every string w that does not contain blanks*

$$M_1 \text{ halts on } w \text{ iff } M_2 \text{ halts on } w.$$

Nondeterminism does not give us machines that are more powerful than standard Turing machines.

Ideas of proof:

- M_1 halts on w if some computation of M_1 halts on w.
- Design M_2 to
  - Perform all length 1 computation of M_1 on w, then
  - Perform all length 2 computation of M_1 on w, then
  - Perform all length 3 computation of M_1 on w, then
  - and so on and so forth.
- Then M_2 halts on w iff some computation of M_1 halts on w.

Church's Thesis:

Turing machines are formal versions of algorithms; no computational procedure will be considered an algorithm unless it can be presented as a Turing machine.