Lecture 26

Homework #26: 4.2.1, 4.2.2, 4.2.3

Convention: Recall from our introduction to Turing Machines that unless specified otherwise, the input to a TM is placed immediately to the right of the left end marker, with the tape head on the first symbol of the input w. Let's add a blank to the front now. This will help us if we want to use some of the handy TMs introduced last time. So now the initial configuration of a TM M on input w is:
(s, >#w)

Def. Let M = (K, Σ, δ, s, H) be a TM, such that H = {y, n} consists of two distinguished halting states (y and n for "yes" and "no", respectively).

- Any halting configuration whose state component is y is called an accepting configuration.
- A halting configuration whose state component is n is called a rejecting configuration.

We say that M accepts $w \in (\Sigma\text{-}\{\#,>\})^*$ iff (s, >#w) yields an accepting configuration;
We say that M rejects w iff (s, >#w) yields a rejecting configuration.

Let $\Sigma_0 \subseteq \Sigma - \{\#,>\}$ be an alphabet, called the input alphabet of M (note that this allows M to have other symbols that it uses in computation).

M **decides** a language $L \subseteq \Sigma_0{}^*$ if for any $w \in \Sigma_0{}^*$ the following is true:

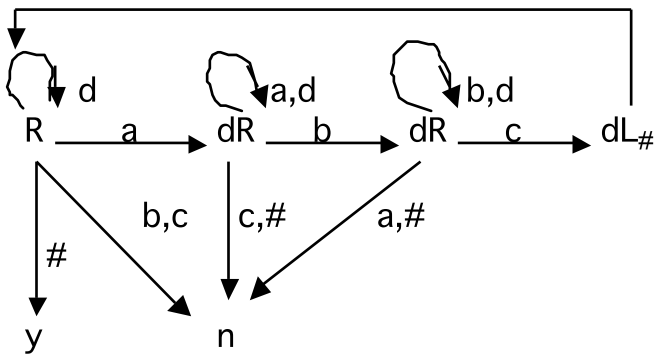
If $w \in L$ then M accepts w; if $w \notin L$ then M rejects w.

A language L is called **recursive** if there is a TM that decides it.

Example. $L = \{a^n b^n c^n : n \geq 0\}$

Let y be a TM that, on any input, immediately transitions to state y;
Let n be a TM that, on any input, immediately transitions to state n.

L is decided by the following TM



An important point: Even if a TM has H = {y, n}, it does not guarantee that the TM decides a language. It may still fail to halt.

Computing with Turing Machines

Let M = (K, Σ, δ, s, {h});
Let $\Sigma_0 \subseteq \Sigma - \{\#, >\}$
Let $w \in \Sigma_0{}^*$

Suppose M halts on w, and that $(s, >\#w) \vdash^* (h, >\#y)$ for $y \in \Sigma_0{}^*$.
y is the **output** of M on **input** w.

Denote it M(w).

Let f: $\Sigma_0{}^* \rightarrow \Sigma_0{}^*$

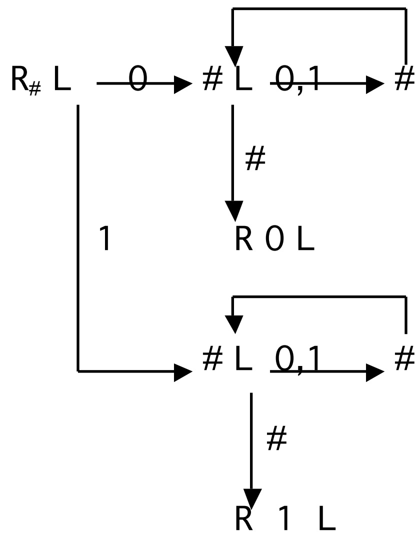
M computes f if, for all $w \in \Sigma_0{}^*$, M(w) = f(w).

A function f is called **recursive**, if there is a TM M that computes f.

Example. Consider the function f: $\{0,1\}^* \rightarrow \{0,1\}^*$, defined as

$$f(n) = \begin{cases} 0, & \text{if } n \text{ is the binary representation of an even number} \\ 1, & \text{if } n \text{ is the binary representation of an odd number} \end{cases}$$

f is computed by the following TM



A function f: $N^k \to N$ is recursive if there is a TM M that computes f.

Note that we can't, in general, determine whether a TM decides a given language or computes a given function (because we can't tell whether it will halt on all input).

Def. Let M = (K, Σ, δ, s, H) be a TM.
Let $\Sigma_0 \subseteq \Sigma - \{\#, >\}$ be an alphabet.

M **semidecides** a language $L \subseteq \Sigma_0{}^*$ if for any $w \in \Sigma_0{}^*$ the following is true: M halts on input w iff $w \in L$.

A language L is **recursively enumerable** iff there is a TM that semidecides L.

Note that if $w \notin L$, the TM does not halt.

We'll write M(w) = $\nearrow$ if M fails to halt on input w.

<u>Thm.</u> If a language is recursive, then it is recursively enumerable.

<u>Thm.</u> If L is a recursive language, then its complement is also recursive.

Pf. If L is decided by a TM M = (K, Σ, δ, s, {y, n}), then its complement is decided by a TM that is identical to M, except that the roles of the states y and n are reversed.