UNLV CS456: Decide/Accept/Recognize

1. A deterministic machine $M$ accepts a string $w$ if, with input $w$, $M$ halts in an accepting state.

2. A non-deterministic machine $M$ accepts a string $w$ if, with input $w$, some computation of $M$ halts in an accepting state. Note: there could be many computations of $M$ with input $w$, perhaps only one which ends in an accepting state. We assume $M$ makes a correct guess at every step.¹

3. Let $L(M)$ be the set of all strings accepted by $M$. We call $L(M)$ the language recognized by $M$. (Some sources use the word “accepted” instead of “recognized.”) $L$ is called recognizable if it is recognized by some machine.

4. A deterministic machine $M \subseteq \Sigma^*$ decides a language $L$ if:
   
   (a) $L = L(M)$
   
   (b) $M$ halts with every input.

   We say a language is decidable if it is decided by some deterministic machine.

5. Let $T$ be a non-decreasing integral function on integers. A machine $M$ accepts $w \in L(M)$ in time $T$ if some computation of $M$ with input $w$ halts in an accepting state within $T(n)$ steps, where $n = |w|$.

Enumeration and Recursive Enumeration

An enumeration of a set $X$ is a sequence which includes each member of $X$. We say $X$ is enumerable if there exists an enumeration of $X$. The word countable means enumerable. We say a set is uncountable if it is not countable, i.e., has no enumeration. For example, $\mathbb{R}$, the set of all real numbers, is uncountable. Every subset of a countable set is countable, thus every language is countable, that is, enumerable. But that does not imply that an enumeration can be computed.

We say that a language $L$ is recursively enumerable, or RE, if there is a machine which writes an enumeration of $L$. If $L$ is infinite, the machine must run forever.

Theorem 1 A language is recursively enumerable if and only if it is recognizable.

Proof: Let $L \subseteq \Sigma^*$ be a language and a machine writes an enumeration of $M$, $w_1, w_2, \ldots$. The following program recognizes $L$.

read $w \in \Sigma^*$
for $i$ from 1 to $\infty$
    if $w = w_i$ write "yes" and halt.

Conversely, suppose $M$ is a machine which recognizes $L$. Let $w_1, w_2, \ldots$ be the canonical enumeration of $\Sigma^*$. The following program enumerates $L$.

¹Yogi Berra, the famous baseball player, once while giving directions to his house said, “When you come to a fork in the road, take it.”
for $t$ from 1 to $\infty$.
  for $i$ from 1 to $t$
    if $M$ accepts $w_i$ within the first $t$ steps
      write $w_i$.

By the Church-Turing thesis, every recognizable language is enumerated by some Turing machine.

**Canonical Order of a Language.** Given a language $L$ and two strings $u, v \in L$, we say that $u$ is before $v$ in canonical order, or simply $u < v$, if one of the following holds:

1. $|u| < |v|$  
2. $|u| = |v|$ and $u$ comes before $v$ alphabetically. (We assume the alphabet of $L$ is ordered.)

The canonical enumeration of $\{0, 1\}^*$ is $\{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$.

**Theorem 2** A language $L$ is decidable and only if some machine computes a canonical order enumeration of $L$.

**Proof:** Suppose $M$ decides a language $L \subseteq \Sigma$. Let $w_1, w_2, \ldots$ be the canonical enumeration of $\Sigma$. The following program enumerates $L$ in canonical order.

for all $i$ from 0 to $\infty$
  if $M$ accepts $w_i$ (Note that $M$ must halt.)
    write $w_i$,

Conversely, suppose a machine $M$ enumerates a language $L \subseteq \Sigma$ in canonical order. Let $w_1, w_2, \ldots$ be the enumeration of $L$ in canonical order.

If $L$ is finite, $L$ is trivially recursive. On the other hand, if $L$ is infinite, the following program decides $L$.

Read a string $w \in \Sigma^*$
for all $i$ from 1 to $\infty$
  if $w = w_i$
    write “yes” and halt
  else if $w > w_i$ // in canonical order
    write “no” and halt

By the Church-Turing thesis, every decidable language is enumerated in canonical order by some Turing machine.