Canonical Ordering and Recursive Enumeration

We define *canonical ordering* on strings as follows.

1. If u, v are strings and $|u| < |v|$, then u is less than v in the canonical ordering.
2. If u, v are string of the same length over an ordered alphabet Σ, we say that u is less than v in the canonical ordering if u comes before v in alphabetic order.

Every language is enumerable, in the sense that an enumeration of the the language exists, but that doesn't mean that an enumeration can be calculated. A language L is defined to be *recursively enumerable* ($\mathcal{RE}$) if there is a machine which enumerates L. There are uncountably many languages over a given alphabet Σ, but only countably many of those are recursively enumerable.

Theorem 1 *If a language L is enumerated in canonical order by some machine, then L is decidable.*

Proof: Let $w_1, w_2, ..$ be the canonical enumeration of L, which is given by some machine. The following program decides L.

Read w
For i = 1 to ∞
  If $w_i = w$ HALT ACCEPT
  If $w_i > w$ in the canonical ordering HALT REJECT

∎

Theorem 2 *If a language L is recursive (decidable), then L is enumerated in canonical order by some machine.*

Proof: Let Σ be the alphabet of L. Let $w_1, w_2, \ldots$ be the canonical enumeration of Σ^*. The following program write an enumeration of L in canonical order.

For i = 1 to ∞
If($w_i \in L$) write w_i

∎

Theorem 3 *If a language L is recursively enumerable, then L is accepted by some machine.*

Proof: Suppose L is recursively enumerable. Then there is some program which (possibly running forever) which writes an enumeration of L. Let $w_1, w_2, \ldots$ be that enumeration. The following program accepts L:

Read w
For i = 1 to ∞
  If($w = w_i$) HALT and ACCEPT

∎

Theorem 4 *If a language L is accepted by some machine M, then L is recursively enumerable.*

Proof: Let M be a machine which accepts a language L over an alphabet *Sigma* The following program enumerates L. Let $w_1, w_2, \ldots$ be the canonical ordering of Σ^*. The following program enumerates L.

```
For t = 1 to ∞
    For i = 1 to t
        If (M accepts w_i within t time units)
            write w_i
```

∎