Lesson 03 - Recursive Fuction Theory

Recursive function theory is a theory about what it means for there to be effective (computable) procedures. We will introduce these ideas in the context of formal language theory; that is, in the context of subsets of the set of all possible strings of finite length over some fixed alphabet Σ. These classes are more typically discussed in the context of arithmetic.

Primitive Recursive sets

Primitive recursive sets are ones that can be generated using particular base functions and particular ways of recursively combining them. Jean Gallier provides has some good notes on them in the context of formal languages.

The functions have domain $(\Sigma^*)^n$ and co-domain $□^*).

The basic idea (which should be familiar from how we defined various grammatical formalisms in comp ling 2) is to define a set of basic functions as well as a couple operators which allows us to obtain new functions from existing ones. Without going into too much detail, the base cases basically consist of the following.

- the constant (zero) function (e.g. $f(w) = \lambda$)
- successor functions (e.g. $f_a(w) = wa$)
- projection functions (e.g. $f_2(u, v, w) = v$)

Each of these basic functions is a primitive recursive (p.r.) function.

The entire class of primitive recursive functions is obtained by closing the set of primitive recursive functions under two operations: extended composition and primitive recursion.

- extended composition. If g is a m-ary p.r. function and there are m n-ary p.r. functions $h_i(1 \leq i \leq m)$, then $f(w_1 \ldots w_n) = g(h_1(w_1 \ldots w_n), \ldots h_m(w_1 \ldots w_n))$ is also a p.r. function.
- If If g is a $m - 1$-ary p.r. function and there are $|\Sigma|$ $m + 1$-ary p.r. functions $h_a(a \in \Sigma)$, then the following function f is a m-ary function defined recursively as follows:
  - $f(□, w_2, \ldots w_m) = g(w_2, \ldots w_m)$
  - $f(ua, w_2, \ldots w_m) = h_a(u, f(u, w_2, \ldots w_m), w_2, \ldots w_m)$ for each $a \in \Sigma$.

Theorem: The primitive recursive sets are a proper subset of the recursive sets.

Theorem: The primitive recursive sets are a proper subset of the recursive sets.

Historically, these functions were thought to include all functions definable with recursive operators. However, Ackermann discovered in 1934 that this was not the case. The Ackermann function bears his name. Around the same time Alonzo Church was developing the lambda calculus, and his student Alan Turing was developing what would later be called Turing machines. Both of these models provided models of computation that exceeded the capacity of primitive recursive functions.

A set of strings S is primitive recursive if it constitutes the range of some primitive recursive function.

Recursively enumerable sets

Recall that a Turing machine is a general computing device. By the Church-Turing thesis, it is the most general kind of computing device. It takes an input, which is written as a sequence of symbols on a 'tape'. The Turing machine processes the tape (reads and writes or rewrites) by mechanically following instructions. Eventually it may halt, in which case what is now written on the tape is its output. However, it may not halt, inc which case following the instructions on some input has the consequence that is just runs forever. A formal language L is *semi-decidable* if for all $w \in L$, there is a Turing Machine M such that $M(w)$ halts (and outputs "Yes").

A *recursively enumberable* (r.e.) set is one for which an effective enumeration ϕ exists. In other words, S is recursively enumerable provided for all $w \in S$, there exists $i \in \mathbb{N}$ such that $\phi(i) = w$. Recursively enumerable sets are also called *computably enumerable* (c.e.).

Theorem: The set of r.e. sets is exactly the set of semi-decidable sets.

Recursive sets

A *recursive* set is an r.e. set whose complement is also r.e. In other words, a set S is recursive provided there are effective enumerations for both it and its complement. Recursive sets are also called *computable*.

A formal language L is *decidable* if for all $w \in L$, there is a Turing Machine M such that $M(w)$ halts (and outputs "Yes") if and only if $w \in L$ and halts (and outputs "No) if and only if $w \notin L$.

It is also possible to define the recursive sets as the range of **recursive** functions, which are definied similarly to the class of p.r. functions.

Theorem: The set of recursive sets are exactly the set of decidable sets.

Theorem: The recursive sets are a proper subset of the r.e. sets.

Theorem: The p.r. sets are a proper subset of the recursive sets.

Theorem: The context-sensitive sets are a proper subset of the p.r. sets.