Post

Algorithms, Systems, and Computable Functions

Emil Post

- The Mathematician Emil Post developed a number of computing techniques
  - Post Algorithms
    - Transforms 1 string into another string
  - Post Systems
    - Generates (a possibly infinite) sets of strings (like a Regular Expression, etc)
  - Post computable functions
    - Formalizes what a computable function is (like a Turing machine)

Post Algorithm

- An alphabet S
- A set of variables V
- A finite set of productions
  - Of the form s -> t
  - Both s and t elements of $(S \cup V)^*$
    - If ***v*** appears in ***t*** then ***v*** must also appear in ***s***
  - Some production are labeled with Halt
  - No order is implied by the productions

Steps in a Post Algorithm

- The is a single string that is the focus of the algorithm. It is given an initial value as input.
- If the focus matches the lhs (s) of some production s -> t, then the focus is updated to match the rhs (t) of that production.
- Any variables appearing in s or t can match any string of symbols in S* (including Λ).
- Example
  - Let S = {a,b} let V = {S,T}
  - Production = SaT -> SbT
  - Focus = abac
  - There are two possibilities
    1. SaT matches abac, where S=Λ and   T = bac
       - This changes the focus to: bbac
    2. SaT matches abac, where S= ab   and   T = c
       - This changes the focus to abbc

Termination

- Steps are repeated until
  1. A production is used that is labeled halt
  2. No production applies to the focus.

1. When the steps terminate, the value of the focus becomes the output of the process
2. A post algorithm transforms a string into another string
3. The process is non-deterministic (why?)

Example

- S = {a,b,x,y}
- V = {S,T}
- Prod =
  1. SaT -> SxxT
  2. SbT -> SyT
  3. SxyT -> SyxT

aba	by 1
xxba	by 1
xxbxx	by 2
xxyxx	by 3
xyxxx	by 3
yxxxx	halt

A different path
Xxya
Xyxa
Yxxa
yxxxx

Post Systems

- Post systems define (possible infinite) sets of strings (just like regular expressions).
- Post systems are strictly more powerful than the Regular languages or the Context Free Languages.
- Post systems are equivalent to Turing Machines
- Post systems are a generalization of Post Algorithms.

Post System Definition

- An alphabet S
- A finite set of axioms in S*
  - Think of these as the initial set of strings in the set of strings we are generating
- A set of inference rules, which are Post style productions.
  - Think of these as rules to add new things to the set of strings we are generating
  - An inference rule may multiple lhs
    - Eg. s_1,s_2,s_3 -> t

- For some systems the inference rules may add an infinite number of new strings, so the set produced may be infinite.

Steps in a Post system

- Given a set of axioms (strings in the focus)
- Given an inference rule
  - s_1, s_2, s_3 -> t
  - If all the s_i match some string in the axiom set
  - Add the matching rhs (t) to the axiom set
- Repeat until no inference rule applies.
  - The steps may never terminate, in which case the set generated will be infinite.
- Post systems are non-deterministic (why?)

Example: a(b+c)*d

- Axioms = {ad}
- Inference rules
  1. aTd -> abTd
  2. aTd -> acTd

{ad}
{ad,acd}
{ad,acd,abd}
{ad,acd,abd,abcd}
…

Other examples

- See the text for
  - Balanced parentheses
  - Palindromes

Post Computable functions

- Suppose we have a function f: with type A* -> A*
- We say f is “post computable”, if there exists a post system that computes the pairs
  - {(x,f(x)} | x in A*}
- We encode tuples (x,f(x)) as the strings
  - X ++ “#” ++ f(x)
  - Where ++ is string concatenation
- The set of post computable functions are equivalent to the functions computable by a Turing Machine.