MA/CSSE 474 Day 31 Summary

1) **TMs as language recognizers**. Let $M = (K, \Sigma, \Gamma, \delta, s, \{y, n\})$.
   a) M ***accepts*** a string w iff $(s, qw) \mid\!-_M^* (y, w')$ for some string w'.
   b) M ***rejects*** a string w iff $(s, qw) \mid\!-_M^* (n, w')$ for some string w'.
   c) M ***decides*** a language $L \subseteq \Sigma^*$ iff for any string $w \in \Sigma^*$ it is true that:
   i) if $w \in L$ then M accepts w, and
   ii) if $w \notin L$ then M rejects w.
   d) A language L is ***decidable*** iff ____.
   e) We define the set ***D*** to be the set of all decidable languages.
   f) M ***semidecides*** L iff, for any string $w \in \Sigma_M^*$:
      i) $w \in L \rightarrow M$ accepts w
      ii) $w \notin L \rightarrow M$ does not accept w. M may either _________________ or ______________________.
   g) A language L is **semidecidable** iff there is a Turing machine that semidecides it.
   h) We define the set SD to be the set of all semidecidable languages.
   i) Another term that means the same thing as semidecidable: recursively enumerable.
   j) Regular languages ⊂ CFLs ⊂ D ⊆ SD ⊆ all languages. [The last two ⊆s are realy ⊂s, but we still need to show it].
2) **TMs can compute functions**. Let $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$.
   a) $M(w) = z$ iff $(s, \square w) \mid\!-_M^* (h, \square z)$.
   b) Let $\Sigma' \subseteq \Sigma$ be M's output alphabet, and let f be any function from Σ^* to Σ'^*.
      i) M ***computes*** f iff, for all $w \in \Sigma^*$:
         (1) if w is an input on which f is defined, then $M(w) = f(w)$.
         (2) otherwise $M(w)$ does not halt.
      c) A function f is ***recursive*** or ***computable*** iff there is a Turing machine M that computes it and that always halts.
      d) **Computing numeric functions**:
         i) For any positive integer k, ***value$_k$(n)*** returns the nonnegative integer that is encoded, base k, by the string n.
         ii) TM M computes **a function *f* from $\mathbb{N}^m$ to $\mathbb{N}$** iff, for some k, $value_k(M(n_1;n_2;\ldots n_m)) = f(value_k(n_1), \ldots value_k(n_m))$.

         Notice that the TM's function computes with strings ($\Sigma^* \mapsto \Sigma'^*$), not directly with numbers.
3) **TM extensions**. For each extension, we can show that every extended machine has an equivalent basic machine.
   a) Multi-track TM. Input symbols are tuples of the input symbols from the tracks
   b) Multiple-tape TM
      i) The transition function for a k-tape Turing machine:
4) ***Theorem** (adding tapes adds no computing power):* Let $M = (K, \Sigma, \Gamma, \delta, s, H\}$ be a k-tape Turing machine for some $k > 1$. Then there is a standard TM M' where $\Sigma \subseteq \Sigma'$, and:
      (1) On input x, M halts with output z on the first tape iff M' halts in the same state with z on its tape.
      (2) On input x, if M halts in n steps, M' halts in $O(n^2)$ steps.
         (a) Proof by construction:
            (i) Treat the single tape as if it were multi-track. This gives M' a large number of tape symbols:
5) Example: Use two tapes to add two natural numbers (represented in binary)

6) Exercise: Use multiple tapes to multiply two natural numbers (represented in binary)

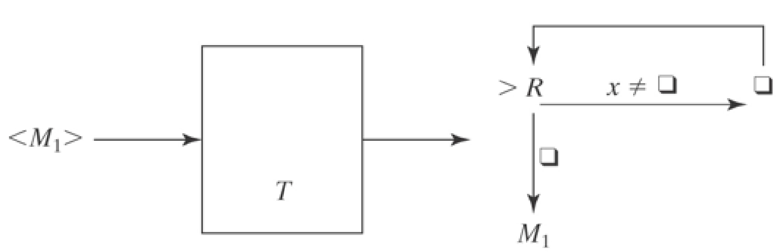
7) **Encoding a TM** M = (K, Σ, Γ, δ, s, H) as a string <M>:
    - i) **Encoding the states**: Let i be $\lceil \log_2(|K|) \rceil$.
        - (1) Number the states from 0 to $|K|$-1 in binary (i bits for each state number):
        - (2) The start state, s, is numbered 0; Number the other states in any order.
        - (3) If t' is the binary number assigned to state t, then:
            - (a) If t is the halting state y, assign it the string yt'.
            - (b) If t is the halting state n, assign it the string nt'.
            - (c) If t is the halting state h, assign it the string ht'.
            - (a) If t is any other state, assign it the string qt'.
    - ii) **Encoding the tape alphabet**: Let j be $\lceil \log_2(|\Gamma|) \rceil$.
        - (1) Number the tape alphabet symbols from 0 to $|\Gamma|$ - 1 in binary.
        - (2) The blank symbol is number 0.
        - (3) The other symbols can be numbered in any order
    - iii) **Encoding the transitions**:
        - (1) (state, input, state, output, direction to move)
        - (2) Example: (q000,a000,q110,a000,$\rightarrow$)
    - iv) **Encoding s and H** (already included in the above)
    - v) A **special case** of TM encoding
        - (1) One-state machine with no transitions that accepts only ε is encoded as (q0)
    - vi) **Encoding other T**Ms: It is just a list of the machine's transitions:
        - (1) Detailed example on slide
    - vii) Consider the alphabet Σ = {(,), a, q, y, n, *h*, 0, 1, comma, $\rightarrow$, $\leftarrow$}. Is the following question decidable?
        - (1) Given a string w in Σ*, is there a TM M such that w = <M> ?

8) We can **enumerate all TMs**, so that we have the concept of "the ith TM"

9) We can have **processes** (TMs?) **whose input and outputs are TM encodings**:

Input: a TM M_1 that reads its input tape and performs some operation P on it.

Output: a TM M_2 that performs P on an empty input tape.



10) **Encoding multiple inputs**: <x_1, x_2, ...x_n>