MA/CSSE 474 Day 30 Announcements and Summary

Announcements:

1) Exam 3 is a week from today.

Main ideas from today

1) Review of Macro language; look at some example machines.

2) Exercise: Initial input on the tape is an integer written in binary, most significant bit first (110 represents 6).

Using Elaine Rich's macro language notation, design a TM that replaces the binary representation of *n* by the binary representation of *n*+1.

3) **TMs as language recognizers**. Let $M = (K, \Sigma, \Gamma, \delta, s, \{y, n\})$.
   a) *M* ***accepts*** a string *w* iff $(s, \underline{q}w) \mid\!-_M{}^* \ (y, w')$ for some string w'.
   b) *M* ***rejects*** a string *w* iff $(s, \underline{q}w) \mid\!-_M{}^* \ (n, w')$ 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].

4) **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*, $\textbf{\textit{value}}_k\textbf{(\textit{n})}$ 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.

5) **TM extensions**. For each extension, we can show that every extended machine has an equivalent basic machine.

a) **Multiple-tape TM**

i) The transition function for a *k*-tape Turing machine:

$$((K\text{-}H),\ \Gamma_1,\ \Gamma_2,\ \ldots,\ \Gamma_k) \quad \text{to} \quad (K,\ \Gamma_{1'},\ \{\leftarrow, \rightarrow, \uparrow\},\ \Gamma_{2'},\ \{\leftarrow, \rightarrow, \uparrow\},\ \ldots,\ \Gamma_{k'},\ \{\leftarrow, \rightarrow, \uparrow\})$$

ii) ***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.

iii) Proof by construction:

(1) Treat the single tape as if it were multi-track. This gives M' a large number of tape symbols:

(a) Alphabet (Γ') of $M' = \Gamma \cup (\Gamma \times \{0, 1\})^k$ "The Representation" slide contains an example.

b) **Non-deterministic TM** (later …)