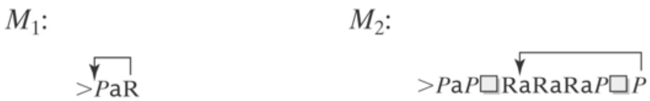
MA/CSSE 474 Day 36 Summary

1) Summary of results from last session:
   a) **The language H** = {<M, w> : TM *M* halts on input string *w*} is in SD but not in D.
   b) If H were in D, then SD would equal D
   c) Every CF language is in D.
   d) D is closed under complement
   e) SD is *not* closed under complement.
   f) A language L is in D iff both L and its complement are in SD.
   g) The language $\neg H$ = {<M, w> : TM *M* does not halt on input string *w*} is not in SD.
2) **Dovetailing:** Run an infinite number of computations "in parallel". S[i, j] represents step j of computation i.
   a) S[1, 1]
   b) S[2, 1] S[1, 2]
   c) S[3, 1] S[2, 2] S[1, 3]
   d) S[4, 1] S[3, 2] S[2, 3] S[1, 4]
   e) For every i and j, S[i, j] will eventually happen.
3) A language is **Turing-enumerable** iff there is a Turing machine that enumerates it.



   a) A language is SD iff it is Turing-enumerable (TE).
      i) TE→SD. Given M that enumerates L, construct M' that semidecides L.
         (1) Save w. Use M to enumerate L. As each string is enumerated, compare to w. If they match, accept.
      ii) SD→TE. Given M that semidecides L, construct M' that enumerates L.
         (1) Enumerate all $w \in \Sigma^*$ lexicographically. As each is enumerated, use *M* to check it.
         (2) The problem with this approach?
         (3) Solution:

4) *M* ***lexicographically enumerates*** *L* iff *M* enumerates the elements of *L* in lexicographic order.
5) *L* is ***lexicographically Turing-enumerable*** iff there is a Turing machine that lexicographically enumerates it.
6) A language is in D iff it is **lexicographically Turing-enumerable**.
   a) D→LTE. Given M that decides L, construct M' that lexicographically enumerates L
      i) *M'* lexicographically generates the strings in Σ^* and tests each using *M* (*M* halts and accepts or rejects each).
      ii) It outputs those that are accepted by *M*.
   b) LTE→D. Given M that lexicographically enumerates L, construct M' that decides L.
      i) Save w. Use M to start enumerating L. As each string is enumerated, compare to w. If they match, accept.
      ii) If M ever generates a string that comes after w in lexicographic order, reject.
7) Problem P_1 is **reducible** to problem P_2 (written $P_1 \leq P_2$) if there is a Turing-computable function f that finds, for an arbitrary instance I of P_1, an instance f(I) of P_2, and
   a) f is defined such that for every instance I of P_1,
   b) I is a yes-instance of P_1 if and only if f(I) is a yes-instance of P_2.
   c) So $P_1 \leq P_2$ means "if we have a TM that decides P_2, then there is a TM that decides P_1.

> In some sense, ≤ means "is no harder than" or "is at least as decidable as"

8) **Special case:** Language L_1 (over alphabet Σ_1) is **reducible** to language L_2 (over alphabet Σ_2) and we write $L_1 \leq L_2$ if there is a Turing-computable function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that $\forall x \in \Sigma_1^*$, $x \in L_1$ if and only if $f(x) \in L_2$
   a) If P_1 is reducible to P_2, then
      i) If P_2 is decidable, so is P_1.
      ii) If P_1 is not decidable, neither is P_2.
   b) The second part is the one that we will use most.

9) Another way to say it:
   a) A ***reduction*** *R* from language L_1 to language L_2 is one or more Turing machines such that:
   b) If there exists a Turing machine *Oracle* that decides (or semidecides) L_2,
   c) then the TMs in *R* can be composed with *Oracle* to build a deciding (or semideciding) TM for L_1.
10) **Using Reduction for Undecidability**
   a) (R is a reduction from L_1 to L_2) $\wedge$ (L_2 is in D) $\rightarrow$ (L_1 is in D)
   b) Contrapositive: If (L_1 is in D) is false, then at least one of the two antecedents of that implication must be false. So: If (R is a reduction from L_1 to L_2) is true and (L1 is in D) is false, then (L_2 is in D) must be false.
   c) **Application:** If L2 is a language that is known to not be in D, and we can find a reduction from L2 to L1, then L1 is also not in D.
11) A framework for using reduction to show undecidability. To show language L_2 undecidable:
   a) Choose a language L_1 that is already known not to be in D, and show that L_1 can be reduced to L_2.
   b) Define the reduction *R and show that it can be implemented by a TM.*
   c) Describe the composition *C* of *R* with *Oracle* (the purported TM that decides L_1).
   d) Show that *C* does correctly decide L_1 iff *Oracle* exists. We do this by showing that *C* is correct. I.e.,
      i) If $x \in L_1$, then $C(x)$ accepts, and
      ii) If $x \notin L_1$, then $C(x)$ rejects.

12) **Example:** H_ε = {<*M*> : TM *M* halts on ε}. Show that it is not in D by showing $H \leq H_\varepsilon$.
   a) H_ε is in SD.
   b) H_ε is not in D.