**MA/CSSE 474 Day 37**

1) **Reducibility Special case:** *Language* $L_1$ (over alphabet $\Sigma_1$) is **reducible** to language $L_2$ (over alphabet $\Sigma_2$) 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.
2) 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 $L_1$.
3) **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.
4) 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.

5) **Example:** $H_\varepsilon = \{<M> : \text{TM } M \text{ halts on } \varepsilon\}$.
   a) $H_\varepsilon$ is in SD.

   b)

   c) $H_\varepsilon$ is not in D. Prove this by showing $H \leq H_\varepsilon$. Details on slides. A place for notes:

   d) The block diagram for C gives some insight into how the reduction shows that $H_\varepsilon$ decidable implies H decidable.

   e) R can be implemented by a TM.
   f) Languages we are dealing with: H, $H_\varepsilon$, the language on which some machine M halts.
   g) Machines we are dealing with: Oracle, R, C, M, M#
   h) View the reduction as a C-like procedure. Reads input, writes output, both of which involve TM encodings/

6) Important elements of a reduction proof
   a) A clear declaration of the reduction "from" and "to" languages.
   b) A clear description of *R*.
   c) If *R* is doing anything nontrivial, argue that it can be implemented as a TM.
   d) Note that machine diagrams are not necessary or even sufficient in these proofs. Use them as thought devices, where needed.
   e) Run through the logic that demonstrates how the "from" language is being decided by the composition of *R* and *Oracle*. You must do both accepting and rejecting cases.
   f) Declare that the reduction proves that your "to" language is not in D.
7) Don't do the reduction backwards!

8) $H_{ANY}$ = {<*M*> : there exists at least one string on which TM *M* halts}
   a) In SD

   b) Not in D. Two different reductions.

Undecidable problems and languages:

| **The Problem View** | **The Language View** |
|---|---|
| Does TM *M* halt on *w*? | H = {<*M*, *w*> : *M* halts on *w*} |
| Does TM *M* not halt on *w*? | $\neg$H = {<*M*, *w*> : *M* does not halt on *w*} |
| Does TM *M* halt on the empty tape? | $H_\varepsilon$ = {<*M*> : *M* halts on $\varepsilon$} |
| Is there any string on which TM *M* halts? | $H_{ANY}$ = {<*M*> : there exists at least one string on which TM *M* halts } |
| Does TM *M* accept all strings? | $A_{ALL}$ = {<*M*> : *L*(*M*) = $\Sigma^*$} |
| Do TMs $M_a$ and $M_b$ accept the same languages? | EqTMs = {<$M_a$, $M_b$> : *L*($M_a$) = *L*($M_b$)} |
| Is the language that TM *M* accepts regular? | TMreg = {<*M*>:*L*(*M*) is regular} |