More on decidability/undecidability.

If an enumerator E prints strings in ascending order by length, i.e., if E prints a string x and later on prints a string y, then

$$|x| \leq |y|,$$

then $L(E)$ is decidable.

Theorem: Every infinite T-rec language includes an infinite decidable subset.

Proof: Let L be infinite and T-rec. There exists an enumerator E such that $L = L(E)$ (thm proved last class).

Idea: define an enumerator E' that enumerates an infinite subset $L' \subseteq L$ in length-monotone ascending order. Thus L' is decidable by the exercise above.

E' := "On no input:

1. let $\ell := 0$
2. Run E until E prints a string w such that $|w| > \ell$.
   a) print w
   b) $\ell := |w|$

   || ℓ is the length of the longest string printed by E so far.
3. Continue running E (go to step 2). //

Note:

1. E' will only print a string if E prints it, so $L(E') \subseteq L(E) (= L)$.
2. E' prints infinitely many strings b/c E prints arbitrarily long strings, so step 2(a–b) is triggered infinitely many times, for strings in strict monotone order by length.
3. $L(E')$ is decidable, by the HW exercise mentioned above.

▨

Functions $\Sigma^* \to \Sigma^*$

Def.: Let $f : \Sigma^* \to \Sigma^*$ (Σ some alphabet)

Say that f is computable if there exists a TM M such that, for every $w \in \Sigma^*$, M on input w eventually halts, leaving $f(w)$ on its tape:

f(w)	blanks

A TM that computes a function in this way is called a transducer.

High-level description of a transducer can use the primitive statement

"output x" ← halting behavior

for a prev. defined string x

Ex: All arith ops are computable (on integers) (of course)

Ex:

f := "On input $\langle M, w, t \rangle$, where M is a TM, w a string, and t a nonnegative integer:

1. Run M on input x for up to t steps.
2. If M halts within t steps, then output $0x$, where x is the final contents of M's tape up to the first blank.
3. Otherwise, output ε. //

Note: For any transducer M computing a function g, for every input $w \in \Sigma^*$,

$0g(w) = f(\langle M, w, t \rangle)$

for all sufficiently large t.

For all other t, $f(\langle M, w, t \rangle) = \varepsilon$.

Thm: Let M be a TM that recognizes some undecidable language L (e.g., A_{TM}). Then let f be the following function:

For all $w \in \Sigma^*$, if M halts on input w, then

$f(w)$ = # of steps it took for M to halt on w.

Otherwise (if M loops on w), $f(w) = 0$.

Then f is not computable.

__Proof__: Suppose f is computable. Then the following D decides $L(M)$ [contradiction]:

D := "On input w:

1. Let $t := f(w)$
2. If $t = 0$ and M does not halt immediately (0 steps, i.e., $q_0 \in \{q_{acc}, q_{rej}\}$), then __reject__
   // (M will loop on input w)
3. Otherwise, simulate M for t steps.
   // M halts in t steps
4. If M accepts w (in t steps) then __accept__; else __reject__."

$L(D) = L(M)$ and D is a decider.
$\therefore L(M)$ is decidable ↯ //

__Reducibility__

Informally — one problem P reduces to another problem Q if you can solve P given a solution for Q.

Formally (for languages):

__Def__: Let $A, B \subseteq \Sigma^*$ be any languages, we say that A mapping-reduces (m-reduces) to B ($A \leq_m B$) if there exists a computable function $f: \Sigma^* \to \Sigma^*$ such that, given any string $w \in \Sigma^*$,

$$w \in A \iff f(w) \in B.$$

The function is called a mapping reduction from A to B (m-reduction), and we might say, "$A \leq_m B$ via f."

__Thm__: Let A, B be languages such that $A \leq_m B$. Then
1. If B is decidable, then A is decidable.
2. If B is T-rec, then A is T-rec.

__Proof__: Let B be recognized by some TM M ($B = L(M)$). Let f m-reduce A to B. Let

N := "On input w:

1. Compute $x := f(w)$
2. Run M on input x:
   a) if M accepts x, then __accept__ // N accepts w
   b) if M rejects x, then __reject__ // N rejects w
   [c) else, loop // N loops on w"

Claim: $A = L(N)$:

__Pf__: $\forall w$,

$$w \in A \iff \underbrace{x}_{f(w)} \in B \quad \text{since } f \text{ m-reduces } A \text{ to } B$$
$$\iff M \text{ accepts } x$$
$$\iff N \text{ accepts } w.$$

$\therefore$ If B is T-rec, then A is T-rec. This proves (2).

For (1), no assume that B is decidable. Choose M to be a decider for B. Then M never loops on any input x, and thus N halts on all inputs w. $\therefore N$ is a decider. Since $A = L(N)$, A is decidable. This proves (1). ▨

__Facts__: $\leq_m$ is reflexive:

$$A \leq_m A$$

and $\leq_m$ is transitive:

if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.