Computability III

If C is a class of languages, then $co - C = \{\overline{L} | L \in C\}$, where $\overline{L} = \Sigma^* - L$ = complement of L. Note that $co - C$ is *not* the same as $\overline{C}$.

$co - P = P$
$co - NP = \{\overline{L} | L \in NP\}$ may not be $= NP$.
co-decidable = decidable
co-semi-decidable $\neq$ semi-decidable. For example $\overline{A_{TM}}$ is co-semi-decidable but not semi-decidable.

Theorem 1 L is decidable iff L is both semi-decidable and co-semi-decidable.

PROOF: $\Rightarrow$ Suppose L is decidable. Then $\overline{L}$ is decidable. Every decidable set is also semi-decidable.

$\Leftarrow$ Assume that both L and $\overline{L}$ are semi-decidable. Say $L = L(M_1)$ and $\overline{L} = L(M_2)$. Construct a decider M_3 so $L(M_3) = L$, where M_3 halts on all inputs. M_3 is given as follows:

> On input x, run M_1 and M_2 "in parallel" on input x. If M_1 accepts x, then accept.
> If M_2 accepts x, then reject.

Either M_1 or M_2 must accept x, so M_3 always halts. To run M_1 and M_2 in parallel, M_3 could use two tapes, one for each machine (a 1-tape machine can simulate a 2-tape machine). Or on input $x\#x$, simulate M_1 on the first copy of x, M_2 on the second, alternating between M_1 and M_2. $\square$

Corollary 1 $\overline{A_{TM}}$ is not semi-decidable.

PROOF: We know that A_{TM} is not decidable, and we also know that A_{TM} is semi-decidable. If $\overline{A_{TM}}$ were semi-decidable, then by the theorem A_{TM} would be decidable, but it is not. $\square$

Reducibility

Recall: if $f : \Sigma_1^* \rightarrow \Sigma_2^*$, M is a TM, we define a notion "M computes f".

M on input x computes $f(x)$ and halts. M must halt on every input.

[1]These notes were taken originally by Antonina Kolokolova in Steven Cook's csc364 lecture, Spring 99, and they have been edited for this term.

Definition 1 f is computable if some TM computes f.

Definition 2 If $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, we say $L_1 \leq_m L_2$ (L_1 is *mapping reducible*, or *many-one reducible*) to L_2 iff there is a computable function $f : \Sigma_1^* \to \Sigma_2^*$, s.t. $x \in L_1 \Leftrightarrow f(x) \in L_2$, for all $x \in \Sigma_1^*$.

Note that $L_1 \leq_p L_2 \Rightarrow L_1 \leq_m L_2$.

Recall: If $L_1 \leq_p L_2$ and $L_2 \in P$, then $L_1 \in P$.

Theorem 2 If $L_1 \leq_m L_2$ and L_2 is decidable, then L_1 is decidable.

PROOF: Suppose M decides L_2. Then M' decides L_1, where M' is given by:

> On input x, compute $f(x)$, then run M on input $f(x)$.

$\square$

Theorem 3 If $L_1 \leq_m L_2$ and L_2 is semi-decidable, then L_1 is semi-decidable.

PROOF: Same as above. $\square$

Corollary 2 If $L_1 \leq_m L_2$, and L_1 is not decidable, then L_2 is not decidable.

PROOF: This is a contrapositive of the above theorem ($A \Rightarrow B$ is equivalent to $\neg B \Rightarrow \neg A$). $\square$

Thus, to show that L is not decidable, it suffices to show $A_{TM} \leq_m L$.

Analogy: If $L_1 \leq_p L_2$ and L_1 is NP-hard, then L_2 is NP-hard.

$HALT_{TM} = \{< M, w > | M \text{ halts on input } w\}$.

Claim 1 $HALT_{TM}$ is not decidable, because $A_{TM} \leq_m HALT_{TM}$.

PROOF: Given an input $< M, w >$ to A_{TM} we construct an input $f(< M, w >) = < M', w' >$ to $HALT_{TM}$, s.t. $< M, w > \in A_{TM} \Leftrightarrow < M', w' > \in HALT_{TM}$, i.e. M accepts $w \Leftrightarrow M'$ halts on w'. Take $w' = w$. Let M' be a modified version of M, namely, M' is the same as M, except M' loops when M hits a reject state. To do so, replace q_{reject} by a new state q', which loops. Therefore, $HALT_{TM}$ is undecidable (but semi-decidable). $\square$

Definition 3 (Emptiness) $E_{TM} = \{M | L(M) = \emptyset\}$

Claim 2 E_{TM} is not semi-decidable.

Proof: We will show $\overline{A_{TM}} \leq_m E_{TM}$. Given an input $< M, w >$ to $\overline{A_{TM}}$ construct and input $< M' >$ to E_{TM}, such that M does not accept w iff $L(M') = \emptyset$. I.e., M accepts w iff $L(M') \neq \emptyset$. M' is given as follows:

On input x, erase x, print w on input tape and run M.

If M accepts w, then $L(M') = \Sigma^*$. If M does not accept w, $L(M') = \emptyset$. □

On the other hand E_{TM} is co-semi-decidable, i.e. $\overline{E_{TM}}$ is semi-decidable. To show this we construct a TM M such that $\overline{E_{TM}} = L(M)$, i.e. M accepts $\overline{E_{TM}}$. M is given as follows:

Let s_0, s_1, s_2, ... be an enumeration of all strings in Σ^*.

```
On input ⟨M⟩
for i : 1..∞,
    for j : 1..i
        Simulate M on input s_j for i steps.
        if M accepts s_j, then ACCEPT.
    end for
end for
```

Definition 4 $EQ_{TM} = \{< M_1, M_2 > | L(M_1) = L(M_2)\}$

Theorem 4 EQ_{TM} is not semi-decidable, and neither is $\overline{EQ_{TM}}$.

Proof: We'll show that $E_{TM} \leq_m EQ_{TM}$ (so EQ_{TM} is not semi-decidable), and that $E_{TM} \leq_m \overline{EQ_{TM}}$ (so EQ_{TM} is not co-semi-decidable).

Given an input $< M >$ to E_{TM}, construct an input $< M_1, M_2 >$ to EQ_{TM} s.t. $L(M) = \emptyset$ iff $L(M_1) = L(M_2)$. Let $M_1 = M$, and M_2 some fixed machine s.t. $L(M_2) = \emptyset$. Then $L(M) = \emptyset \Leftrightarrow L(M) = L(M_2) = \emptyset$.

For the second statement, $L(M) = \emptyset$ iff $L(M_1) \neq L(M_2)$. Take $L(M_2) = \Sigma^*$. Design M_1 so that it accepts everything iff M accepts something. □

Theorem 5 All semi-decidable sets are many-one reducible to A_{TM}.

In other words, if L is semi-decidable, then $L \leq_m A_{TM}$. Analog in the theory of NP-completeness: If L is in NP, then $L \leq_m SAT$.

Proof of the theorem: Let L be semi-decidable, and suppose that M_0 semi-decides (accepts) L. In order to show that $L \leq_m A_{TM}$ we must, for each $w \in \Sigma^*$, construct $\langle M, w' \rangle$ such that $w \in L$ iff $\langle M, w' \rangle \in A_{TM}$ (i.e., iff M accepts w'). Take $w' = w$, $M = M_0$. This finishes the proof.