CS 373: Theory of Computation
Sariel Har-Peled and Madhusudan Parthasarathy

Discussion 13: Reductions

29 April 2009

Questions on homework ?

Any questions? Complaints, etc?

1 Undecidability and Reduction

(Q1) Prove that the language $\text{ODD}_{TM} = \left\{ M \;\middle|\; L(M) \text{ has odd length strings} \right\}$ is undecidable.

Solution:

Let $N(x)$ be the output of the procedure below:

if $x = ab$, accept x; if M accepts w (by simulation), accept x, else reject x;

Thus $L(N) = \Sigma^*$ (which includes odd length strings) if M accepts w , and $L(N) = \{ab\}$ (a set of even length strings) otherwise.

We reduce A_{TM} to ODD_{TM}.

Suppose IsOdd(M) is a decider for ODD_{TM}.

$DeciderATM(M, w)$:
$N \leftarrow$ generate the code for procedure $N(x)$ above;
return IsOdd(N)

(Q2) Prove that the language $\text{SUBSET}_{TM} = \left\{ \langle M, N \rangle \;\middle|\; L(M) \subseteq L(N) \right\}$ is undecidable.
Hint: Reduce $EQ_{TM} = \left\{ \langle M, N \rangle \;\middle|\; L(M) = L(N) \right\}$ to SUBSET_{TM} for this purpose.

Solution:

Suppose IsSubset(M, N) is a decider for SUBSET$_{TM}$.

DeciderEQTM(M, N) :
if IsSubset(M, N) and IsSubset(N, M) then return "Yes", else return "No"

(Q3) Assume L_1 and L_2 are recognizable languages where $L_1 \cup L_2 = \Sigma^*$. Reduce L_1 to $L_1 \oplus L_2$

Solution:

Let $ORAC_{xor}$ be a decider for $L_1 \oplus L_2$ and let M_1 be a machine that recognizes L_1 and define M_2 similarly for L_2.

$Decider_1(x)$:

Simulate $ORAC_{xor}$ on x

If $ORAC_{xor}$ rejects, then accept

Let x_1 be a simulation of M_1 on x and x_2 be a simulation of M_2 on x

While true:

Advance x_1 and x_2 by 1 step

If x_1 accepts, accept

If x_2 accepts, reject

Since all strings are in either L_1, L_2 or both, if a string isn't in $L_1 \oplus L_2$, it means that it is in L_1. Otherwise the string is in either one of L_1 or L_2. Since both M_1 and M_2 must halt on strings they accept, one of the machines will eventually halt.

(Q4) Let EQ$_{TM} = \{\langle M, N\rangle \mid L(M) = L(N)\}$. Reduce A_{TM} to EQ$_{TM}$ as another way to prove that EQ$_{TM}$ is undecidable

Solution:

For a given (M, w), consider M_w defined as follows:

$M_w(y)$:
If $y \neq w$
Reject
else
Simulate M on w and accept only if it accepts.

And, consider N_w defined as follows:

$N_w(y)$:
If $y = w$
Accept
else
Reject

Let $ORAC_{EQ}$ be a decider for EQ_{TM}. We can design a decider for A_{TM} as follows using M_w and N_w:

$Decider_A(M, w)$:
Simulate $ORAC_{EQ}$ on (M_w , N_w)
Accept only if the above simulation accepts.

N_w only accepts w and M_w accepts nothing if M does not accept w, and $\{w\}$ otherwise. Therefore the two languages will be equal iff M accepts w.

(Q5) Prove that L is not recursively enumerable (is not recognizable):

$$L = \{\langle M \rangle : L(M) \text{ is infinite.}\}$$

Solution:

We reduce $\overline{A_{TM}}$ to L. Let's check the following routine first (fix M and w):

$N(x)$:
Simulate $M(w)$ for $|x|$ steps.
Accept, iff simulation above does not accept.

Observe that $L(N)$ is infinite iff $M(w) \neq$ Yes. So now we have the following reduction:

$Decider_{\overline{A_{TM}}}(M, w)$:
$N \leftarrow$ generate code for $N(x)$
Negate the result of $Decider_L(N)$ and return it.