1 Mapping Reductions

Let $ALL_{TM}$ be defined as follows:

$$ALL_{TM} = \left\{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^* \right\}$$

Provide a mapping reduction to show that $ALL_{TM}$ is undecidable.

**Solution:** We will provide a reduction from $A_{TM}$ i.e. $A_{TM} \leq_m ALL_{TM}$ through a Turing-computable function $f$ below. We define $f(\langle M, w \rangle) = \langle R \rangle$ where $R$ is the following TM:

$$R = \text{“On input } x:$$

1. Simulate $M$ on $w$.
2. If $M$ ever accepts, accept $x$.

In other words $f$ uses its input $\langle M, w \rangle$ to write the description of the above TM $R$. Note that $f$ itself is not simulating $M$, that is the job of $R$. $f$ just outputs the encoding of our definition of $R$ so it is computable.

Finally note that when $M$ accepts $w$ then $L(R) = \Sigma^*$. Otherwise $L(R) = \emptyset$. Therefore $f$ maps $\langle M, w \rangle \in A_{TM}$ to $R \in ALL_{TM}$ and $\langle M, w \rangle \not\in \overline{A_{TM}}$ to $R \not\in \overline{ALL_{TM}}$. We thus can conclude $A_{TM} \leq_m ALL_{TM}$ so $ALL_{TM}$ is undecidable.

2 Computation Histories

- **Configurations:** At each step of a TM’s computation three things can change: the state, the content of the tape, and the location of the head in the tape. A configuration is a “snapshot” of these items encoded in a string. Suppose a TM is in state $q$ and the content of the tape is $uv$ where $u$ and $v$ are strings and the TM’s head is on the first symbol of $v$. Then we can encode this configuration as $uqv$.

- **Computation history:** An accepting computation history for TM $M$ on input $w$ is a sequence of configurations, $C_0, C_1, \ldots, C_n$ such that $C_0$ is the starting configurations i.e. $C_0 = q_0w$, $C_n$ is an accepting configuration i.e. $q \in C_n$ is an accept state, and for every $0 \leq i \leq n-1, C_{i+1}$ correctly represents $M$’s configuration one step after $C_i$. We write these sequences as a string $\#C_0\#C_1\# \ldots \#C_n\#$. 

• How do we check a given string represents an accepting computation history of $M$ on $w$?

1. Check that $C_0$ is a the starting configuration.
2. Check that $C_n$ is an accepting configuration.
3. For every $0 \leq i \leq k - 1$, check that $C_{i+1}$ follows from $C_i$.

• Note that the above checks are all local. Therefore a Linear Bounded Automaton (LBA) can perform all the checks without using extra memory. So for an instance of $\langle M, w \rangle$ we can construct an LBA $A_{M,w}$ such that $A_{M,w}$ only accepts strings which are computation histories of $M$ on $w$. To reiterate $A_{M,w}$ performs the above three steps and accepts if and only if all steps succeed. Note that when $M$ accepts $w$ then a valid computation history exists so $L(A_{M,w}) \neq \emptyset$. Otherwise when $M$ does not accept $w$ one of the above steps will always fail in any input to $A_{M,w}$ so $L(A_{M,w}) = \emptyset$. Hence $A_{TM} \leq_m E_{LBA}$ so $E_{LBA}$ is undecidable.

Define $ALL_{PDA}$ as follows:

$$ALL_{PDA} = \{ \langle A \rangle | A \text{ is a PDA and } L(A) = \Sigma^* \}$$

3 Example - $ALL_{PDA}$

Show that $ALL_{PDA}$ is undecidable.

Solution: The proof is similar to showing that $E_{LBA}$ is undecidable. Assume towards a contradiction that $ALL_{PDA}$ is decidable by a decider $R$. We construct a decider $S$ for $A_{TM}$.

In proving that $E_{LBA}$ is undecidable, we constructed an LBA $A_{M,w}$ that accepts a computation history of $M$ on $w$. Here we will construct a PDA $B_{M,w}$ that will accept everything except the accepting computation history of $M$ on $w$.

What makes a string NOT be an accepting computation history? Instead of verifying that all steps succeed we just need to find one step that fails. In other words either

1. $C_0$ is not the starting configuration
2. $C_n$ is not the accepting configuration
3. For some $0 \leq i \leq n - 1$, $C_{i+1}$ does not follow from $C_i$.

So $B_{M,w}$ needs to check that one of the above conditions hold. The first thing that $B_{M,w}$ is going to do is to nondeterministically guess which of the conditions to check. If it is the first one, $B_{M,w}$ accepts if $C_0$ is not the starting configuration. If it is the second one $B_{M,w}$ accepts if $C_n$ is not an accepting configuration.

How does a PDA check the third condition? It will nondeterministically guess the value of $i$ for which it needs to check that $C_{i+1}$ does not follow from $C_i$. Now, it will push $C_i$ onto the stack. Then, it will pop off the stack to compare with $C_{i+1}$. 


But at this point $B_{M,w}$ faces a problem — the order of $C_i$ is reverse to that of $C_{i+1}$. To solve this issue, we write the computation history differently — every other configuration appears in reverse order:

$#C_0#C_1^R# \ldots #C_{n-1}^R#C_n#$

or

$#C_0#C_1^R# \ldots #C_{n-1}^R#C_n#$

depending on the parity of $n$.

Now, when $B_{M,w}$ pops the stack, the configuration it receives is in the same order as the one it compares with. Finally, we construct the decider $S$ for $A_{TM}$.

$S = \text{"On input } \langle M, w \rangle:"

1. Build the PDA $B_{M,w}$.
2. Run the decider for $ALL_{PDA}$, $R$, on $\langle B_{M,w} \rangle$.
3. If $R$ accepts, we reject.
4. Otherwise, accept.

Note that $L(S) = \Sigma^*$ when $M$ does not accept $w$. Otherwise $L(S) = \Sigma^* - \{C\}$ where $C$ is the accepting computation history of $\langle M, w \rangle$.

4 Example - $EQ_{CFG}$

$EQ_{CFG} = \{ <G_1, G_2> | G_1 \text{ and } G_2 \text{ are CFGs and } L(G_1) = L(G_2) \}$

By way of contradiction, we assume that $EQ_{CFG}$ is decidable and has a decider $R$. We use this decider to decide $ALL_{CFG}$ is decidable, which we proved was undecidable in lecture. So we are showing $ALL_{CFG} \leq EQ_{CFG}$ - in other words, we are reducing $ALL_{CFG}$ to $EQ_{CFG}$.

First we define a grammar $G_2$ that generates $\Sigma^*$. We construct a TM $M$ that decides $ALL_{CFG}$ as follows:

$M = \text{"on input } <G> \text{ where } G \text{ a CFG:"

1. Run $R$ on $<G, G_2>$
2. if $R$ accepts, accept. if $R$ rejects, reject.

We see that $M$ decides $ALL_{CFG}$. If the input CFG, $G$ does not generate $\Sigma^*$, then $R$ rejects because $L(G)(G_2)$ since $L(G_2) = \Sigma^*$. If $G$ does generate $\Sigma^*$, then $R$ accepts because $L(G) = L(G_2)$. So given a decider for $EQ_{CFG}$, we have constructed a decider for $ALL_{CFG}$. But since we have proven $ALL_{CFG}$ is undecidable, a decider for $EQ_{CFG}$ can not exist, thus $EQ_{CFG}$ is undecidable.
5 Example - $E_{2WAY-CA}$

See question/solution 9 on practice midterm