Reminder

We continue to study the hardness of various computational problems. We currently have the following hierarchy of classes of languages:

\[ \phi \subsetneq \text{REG} \subsetneq \text{CFL} \subsetneq \text{DEC} \subsetneq \text{RECOG} \subsetneq \Sigma^* \]

where \( \text{REG} \) denotes the set of regular languages, \( \text{CFL} \) the set of context-free languages, \( \text{DEC} \) the set of decidable languages, \( \text{RECOG} \) the set of recognizable languages and \( \Sigma^* \) is the set of all possible languages. Let \( \text{UNDEC} \) denote the set of undecidable languages and let \( \text{UNRECOG} \) denote the set of unrecognizable languages. As a reminder:

- \( \Sigma^* \in \text{REG} \)
- \( \{0^k1^k : k \geq 0\} \in \text{CFL} \setminus \text{REG} \)
- \( \{0^k1^k2^k : k \geq 0\} \in \text{DEC} \setminus \text{CFL} \)
- \( A_{TM} \in \text{RECOG} \setminus \text{DEC} \)
- \( \overline{A_{TM}} \in \Sigma^* \setminus \text{RECOG} \)

Note that \( A_{TM} \in \text{UNDEC} \) and \( \overline{A_{TM}} \in \text{UNRECOG} \).

We are currently focusing on the Turing Machine as our model of computation and the notion of mapping reducibility as a tool to compare the hardness of two problems. Recall that a language \( A \) is mapping reducible to a language \( B \), denoted by \( A \leq_m B \) if and only if there exists a computable reduction \( f : A \to B \) such that \( a \in A \iff f(a) \in B \). This notion of a reduction enjoys the following nice and intuitive properties (that one should prove):

1. If \( A \leq_m B \), then \( A \) is no harder to solve than \( B \). In fact, if there exists a solver for \( B \) that of a certain caliber, then there exists one for \( A \) of the very same caliber. This gives us the following:
   - \( B \in \text{DEC} \implies A \in \text{DEC}, \) or \( A \in \text{UNDEC} \implies B \in \text{UNDEC} \).
   - \( B \in \text{RECOG} \implies A \in \text{RECOG}, \) or \( A \in \text{UNRECOG} \implies B \in \text{UNRECOG} \).

2. It is possible to have \( A \leq_m B \) and \( B \leq_m A \) which would imply that the two are equally easy/hard.

3. Mapping reducibility is transitive, that is, if \( A \leq_m B \) and \( B \leq_m C \), then \( A \leq_m C \). Intuitively, if \( A \) is no harder than \( B \) and \( B \) is no harder than \( C \), then \( A \) is no harder than \( C \).
4. An unexpected but wonderful property: \( A \leq_m B \iff \overline{A} \leq_m \overline{B} \). In fact, if \( A \leq_m B \) by virtue of the reduction \( f \), the same reduction is a witness to the fact that \( \overline{A} \leq_m \overline{B} \).

We finally have one more interesting property that follows from dovetailing.

**Lemma 1.** For any language \( L \), \( L \in \text{DEC} \iff L, \overline{L} \in \text{RECOG} \).

The above lemma gives us the following.

**Lemma 2.** For any language \( L \) if \( L \in \text{RECOG} \setminus \text{DEC} \), then \( \overline{L} \in \text{UNRECOG} \).

For instance, we noted that \( \overline{E_{TM}} \in \text{RECOG} \setminus \text{DEC} \) as \( E_{TM} \in \text{UNDEC} \), and this implies that \( E_{TM} \in \text{UNRECOG} \). Equipped with the above properties and the closure properties for \( \text{DEC} \) and \( \text{RECOG} \) languages, we are ready to study the hardness of certain computational problems.

**Example 1 — Will a 2-tape TM used its second tape?**

Consider 2-tape TMs as TMs that have a standard input tape and an additional workspace tape. Both tapes are read-write tapes. Let

\[
L_{2\text{tape}} = \{ \langle N, w \rangle : N \text{ is a 2-tape TM that uses its second tape while running on } w \} \]

Show that \( L_{2\text{tape}} \in \text{UNDEC} \).

**Solution**

We show that \( L_{2\text{tape}} \in \text{UNDEC} \) by showing that \( A_{TM} \leq_m L_{2\text{tape}} \). We describe here the reduction \( f : A_{TM} \rightarrow L_{2\text{tape}} \). Let \( M \) be a standard 1-tape TM and let \( w \) be a string. We have \( f(\langle M, w \rangle) = \langle N_M, w \rangle \) where \( N_M \) is a 2-tape TM defined as follows:

- On input \( w \), \( N_M \) simulates \( M \) on \( w \) on the input (first) tape of \( N_M \).
- If \( M \) ever halts and accepts \( w \), then \( N_M \) writes a symbol on its second tape.

Firstly note that \( f \) is computable. Furthermore, \( \langle M, w \rangle \in A_{TM} \iff \langle N_M, w \rangle \in L_{2\text{tape}} \). This shows that \( A_{TM} \leq_m L_{2\text{tape}} \) and hence completes the proof.

**Example 2 — Will a TM try to go to the left of its left-most cell?**

Let

\[
L_{\text{left}} = \left\{ \langle N, w \rangle : N \text{ is a TM that attempts to go to the left of its left-most cell while running on } w \right\} \]

Show that \( L_{\text{left}} \in \text{UNDEC} \).
Solution

We show that $L_{\text{left}} \in \text{UNDEC}$ by showing that $A_{TM} \leq_m L_{\text{left}}$. We describe here the reduction $f : A_{TM} \rightarrow L_{\text{left}}$. Let $M$ be a TM and let $w$ be a string. We have $f(\langle M, w \rangle) = \langle N_M, w \rangle$ where $N_M$ is a TM defined as follows:

- On input $w$, $N_M$ simulates $M$ on $w$. It additionally takes care to ensure that if $M$ ever attempts to go to the left of the left-most cell while running on $w$, $N_M$ does not. This can be ensured by having a special marker symbol on the left-most cell that helps detect this pathological behavior of $M$ on $w$.

- If $M$ ever halts and accepts $w$, then $N_M$ keeps moving left and attempts to go to the left of the left-most cell.

Firstly note that $f$ is computable. Furthermore, $\langle M, w \rangle \in A_{TM} \iff \langle N_M, w \rangle \in L_{\text{left}}$. This shows that $A_{TM} \leq_m L_{\text{left}}$ and hence completes the proof.

Example 3 — Is the language of a TM regular?

Let $REG_{TM} = \{ \langle N \rangle : N$ is a TM such that $L(N) \in \text{REG} \}$

Show that $REG_{TM} \in \text{UNDEC}$.

Solution

We show that $REG_{TM} \in \text{UNDEC}$ by showing that $A_{TM} \leq_m REG_{TM}$. We describe here the reduction $f : A_{TM} \rightarrow REG_{TM}$. Let $M$ be a TM and let $w$ be a string. We have $f(\langle M, w \rangle) = \langle N_M, w \rangle$ where $N_{M,w}$ is a TM defined as follows:

- On input $w'$, $N_{M,w}$ first checks if $w' \in \{0^k1^k : k \geq 0 \}$. If so, it accepts $w'$ and halts.

- If not, $N_{M,w}$ simulates $M$ on $w$. If $M$ ever halts and accepts $w$, then $N_M$ accepts $w'$ and halts.

We motivate the above construction as follows. The correctness of the reduction stipulates that $L(N_{M,w}) \in \text{REG} \iff \langle M, w \rangle \in A_{TM}$. Let us say we begin designing the reduction with the following goals in mind:

- If $\langle M, w \rangle \in A_{TM}$, then $L(N_{M,w}) = \Sigma^* \in \text{REG}$.

- If $\langle M, w \rangle \notin A_{TM}$, then $L(N_{M,w}) = \{0^k1^k : k \geq 0 \} \notin \text{REG}$.

Notice that a reduction that respects the above conditions would be correct. We now argue that out reduction does in fact satisfy the above conditions. Firstly, $\{0^k1^k : k \geq 0 \} \subseteq L(N_{M,w})$ regardless of whether $\langle M, w \rangle \in A_{TM}$ or not. This justifies the first step in the description of $N_{M,w}$ which accepts all $w' \in \{0^k1^k : k \geq 0 \}$. Now, by definition, $N_{M,w}$ accepts other $w'$ if and only if $\langle M, w \rangle \in A_{TM}$. This justifies the second step in the description of $N_{M,w}$ which accepts other $w'$ if $M$ accepts $w$. This shows that $\langle M, w \rangle \in A_{TM} \iff \langle N_{M,w} \rangle \in REG_{TM}$.
To complete the proof, we need to show that $f$ is computable. This is the part that might seem most confusing. This is because in the second step of the description of $N_{M,w}$, we simulate $M$ on $w$. What if $M$ loops on $w$? The thing to realize is that this does not matter. We are not running $N_{M,w}$. It is possible to write down a call to a function that loops, without looping. Notice that we only have to argue that we can write down the description of $N_{M,w}$ without looping, and not that $N_{M,w}$ doesn’t loop. The fact that $N_{M,w}$ may loop should be the worry of a decider that tries to decide if $N_{M,w} \in REG_{TM}$ and not the reduction that writes down the description of $N_{M,w}$.

Thus, $f$ is computable and $\langle M, w \rangle \in A_{TM} \iff \langle N_{M,w} \rangle \in REG_{TM}$. This shows that $A_{TM} \leq_m REG_{TM}$ and hence completes the proof.

Note that the above reduction can be trivially modified to show that $CFL_{TM} \in UNDEC$ where

$$CFL_{TM} = \{ \langle N \rangle : N \text{ is a TM such that } L(N) \in CFL \}$$