

CSE105 - HOMEWORK 9 SOLUTIONS

We show the undecidability of the various languages using proofs by contradictions.

- $L = \{\langle M \rangle \mid M \text{ is a TM with a stay option that uses its stay option on some input}\}$.

Let M_L be a decider for L . Then the following is a decider for A_{TM} .

M_{ATM} : on input $\langle M, w \rangle$

1. Construct M' : on input $\langle M \rangle$
 - Simulate $M(w)$ without using the stay option.
 - if $M(w)$ accepts, use the stay option for one step.
2. Run $M_L(M')$ and accept or reject if it does.

- $L = \{\langle M \rangle \mid M \text{ enters its reject state on some input}\}$.

Let M_L be a decider for L . Then the following is a decider for A_{TM} .

M_{ATM} : on input $\langle M, w \rangle$

1. Construct M' : on input $\langle M \rangle$
 - Simulate $M(w)$ without entering the reject state (of M').
 - if $M(w)$ accepts, enter the reject state.
2. Run $M_L(M')$ and accept or reject if it does.

- $L = \{\langle M \rangle \mid M \text{ enters its accept state on some input}\}$.

Let M_L be a decider for L . Then the following is a decider for A_{TM} .

M_{ATM} : on input $\langle M, w \rangle$

1. Construct M' : on input $\langle M \rangle$
 - Simulate $M(w)$ without entering the accept state (of M').
 - if $M(w)$ accepts, enter the accept state.
2. Run $M_L(M')$ and accept or reject if it does.

- $L = \{\langle M \rangle \mid M \text{ is a 2-way TM that moves to the left of its starting position on some input}\}$. Let M_L be a decider for L . Then the following is a decider for A_{TM} .

M_{ATM} : on input $\langle M, w \rangle$

1. Construct M' : on input $\langle M \rangle$
 - Simulate $M(w)$ using only the tape cells to the right of the start position. Note that M itself is assumed to be a one-way TM, so if we simulate it verbatim, we'd need only to replace attempts at moving left past the start position with a right move followed by a left move.
 - if $M(w)$ accepts, move left past the start position.
2. Run $M_L(M')$ and accept or reject if it does.

- $L = \{\langle M_1, M_2, M_3 \rangle \mid L(M_1) = L(M_2) = L(M_3)\}$.

Let M_L be a decider for L . Then the following is a decider for EQ_{TM} .

M_{EQTM} : on input $\langle M_1, M_2 \rangle$

- Simulate $M_L(M_1, M_2, M_2)$.
- Accept or reject if it does.

- $L = \{\langle M_1, M_2 \rangle \mid L(M_1) \cap L(M_2) = \emptyset\}$. Let M_L be a decider for L . Then the following is a decider for EMPTY_{TM} .

$M_{\text{EMPTY}_{TM}}$: on input $\langle M \rangle$:

1. Let M_{ALL} be a TM that accepts on every input.
2. Run $M_L(M, M_{ALL})$.
3. Accept or reject if it does.

We show the decidability of the following two languages by constructing deciders.

- $L = \{\langle M, w \rangle \mid M(w) \text{ ever reads a blank}\}$. Suppose that M does *not* read a blank on input w . Then M looks at only $|w|$ tape cells (at most); any other tape cells will be blank when M scans to them. Now let $|Q|$ be the number of states that M has, and $|\Gamma|$ be the number of tape symbols. Since M has a finite number of states, uses only a finite number of tape symbols (as do all TMs), and accesses a bounded number of tape cells, the total number of configurations it can be in is finite. In fact, it is bounded by $2^{|\Gamma||Q||w|}$.

We use this fact to give a decider for L . The idea is that if M is to look at a blank symbol, it must do so within $2^{|\Gamma||Q||w|} + 1$ steps. Otherwise, it will either have stopped or been in the same configuration at least twice. If it has visited the same configuration twice, we can conclude that it is looping and furthermore that the loop does not contain a stop on a blank symbol. So, we have the following decider for L .

M_L : on input $\langle M, w \rangle$,

1. Simulate $M(w)$ for $2^{|\Gamma||Q||w|} + 1$ steps.
2. If it read a blank symbol during those steps, accept.
3. Otherwise, reject.

- $L = \{\langle M, w, k \rangle \mid M(w) \text{ accepts within } k \text{ steps}\}$.

M_L : on input $\langle M, w, k \rangle$,

1. Simulate $M(w)$ for k steps.
2. Accept if it does; otherwise reject.