CS 39
Fall 2006
Theory of Computation

Homework 7 Solutions

Prepared by Khanh Do Ba

Prof. Amit Chakrabarti
Computer Science Department
Dartmouth College

1. **(Infinite recognizable languages contain infinite decidable ones)**

Proof. Recall that a language is recognizable if and only if there is an enumerator for it, and a language is decidable if and only if there is a enumerator for it which enumerates all strings in canonical order. We use these two facts to prove our claim.

Let L be any infinite recognizable language, and let E be an enumerator of L. We construct an enumerator E' which enumerates an infinite subset of L in canonical order, as follows.

Run E.
On first string w_0 output by E, OUTPUT w_0.
for each string w output by E **do**
    if $w >$ last string output (by E') **then** OUTPUT w.

Clearly, E' enumerates a subset L' of L in canonical order, so it remains to show that L' is in fact infinite. Suppose, to get a contradiction, that it is finite. Then let w_{max} be the greatest string in L', and hence the last one output by it. But L is infinite, and there are only finitely many strings $< w_{max}$, so E must eventually output a string $> w_{max}$, at which point E' would have output it as well. This contradicts the definition of w_{max}, completing our proof. □

2. **(Detectability of useless states)**

2.1. *Proof.* The following is a decider for $\mathsf{U}_{\mathsf{DFA}}$.

On input DFA $M = (Q, \Sigma, \delta, q_0, F)\ldots$
Mark q_0.
while there is an unmarked state $q \in Q$ that is reachable from a marked state **do**
    Mark q.
if there exists an unmarked state **then** ACCEPT **else** REJECT.

This is simply a search (either breadth-first or depth-first will do) of the underlying digraph of the DFA, at the end of which every state reachable from q_0 is marked. Hence, a simple traversal through all the states suffices to check if any state remaines unmarked. □

2.2. *Proof.* We know that $\mathsf{E}_{\mathsf{PDA}} = \{\langle M \rangle : M \text{ is a PDA and } \mathcal{L}(M) = \varnothing\}$ is decidable, so let D be a decider for it. We will use D as a subroutine to decide $\mathsf{U}_{\mathsf{PDA}}$ as follows.

On input PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)\ldots$
for each $q \in Q$ **do**
    Run D on the PDA $(Q, \Sigma, \Gamma, \delta, q_0, \{q\})$.
    if D accepts **then** ACCEPT.
REJECT.

Since PDAs have stacks, it is not enough to only consider the underlying digraph as in the case of DFAs, since although there may be a path in this graph, that path might not be realizable because of requirements on the stack values. Our strategy, however, guarantees that we detect all unreachable states, since by making each state the only final state, the language captured is nonempty if and only if that final state is reachable. □

2.3. *Proof.* Suppose, to get a contradiction, that U_{TM} is decidable, with decider D. We then use it as a subroutine to *nondeterministically* decide $\overline{\mathsf{E}_{\mathsf{TM}}}$, which we know to be undecidable, giving us the desired contradiction.

On input TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\ldots$

if $q_0 = q_{\text{accept}}$ **then** ACCEPT.
Guess a subset S of $Q - \{q_0, q_{\text{accept}}, q_{\text{reject}}\}$, possibly empty.
Let δ' be the restriction of δ to $Q - S$.
Run D on the TM $(Q - S, \Sigma, \Gamma, \delta', q_0, q_{\text{accept}}, q_{\text{reject}})$.
if D rejects **then** ACCEPT **else** REJECT.

Note that wherever δ maps to a state not in $Q - S$, δ' simply maps to q_{reject}. Now, to see that this is a correct decider for $\overline{\mathsf{E}_{\mathsf{TM}}}$, first observe that if $q_0 = q_{\text{accept}}$, then M would certainly accept ε, so our decider correctly accepts. If this is not the case, it remains to check whether or not q_{accept} is a useless state. The key observation here is that if it *is* a useless state, then no matter how many other states we trim away, it will still be useless. On the other hand, if it is *not* useless, than some other states might be useless, but some incarnation of our decider will guess exactly this set of useless states to be S. As a result, the restricted TM would have no useless states, so that D would reject and our decider accept. □

3. **(Recognizability vs. decidability)**

3.1. $A = \{\langle M \rangle : M \text{ is a TM and } M \text{ accepts at least two strings}\}$ is recognizable, but not decidable.

Proof. Below is a simple recognizer for A.

On input TM $M\ldots$

Convert M into an equivalent enumerator E.
Run E.
if E ever outputs two strings **then** ACCEPT.

Now, suppose, to get a contradiction, that A is decidable. Let D be a decider for A, which we will use to decide A_{TM}, giving us the desired contradiction.

First, let us defined a TM $N_{M,w}$, which depends on another TM M and a string w, as follows.

On input u

Run M on w.
if M ever accepts **then** ACCEPT.
if M ever rejects **then** REJECT.

Observe that

$$\mathcal{L}(N_{M,w}) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \varnothing & \text{otherwise} \end{cases}$$

We can then decide A_{TM} as follows.

On input TM M and string w...

1 Run D on $\langle N_{M,w} \rangle$.
2 **if** D accepts **then** ACCEPT **else** REJECT.

Now, by our construction of $N_{M,w}$,

$$\begin{aligned} M \text{ accepts } w &\iff \mathcal{L}(N_{M,w}) = \Sigma^* \\ &\iff N_{M,w} \text{ accepts at least two strings} \\ &\iff D \text{ accepts } N_{M,w} \\ &\iff \text{the above TM accepts } \langle M, w \rangle \end{aligned}$$

completing our proof. □

3.2. $A = \{\langle M \rangle : M \text{ is a TM and } M \text{ accepts exactly two strings}\}$ is unrecognizable.

Proof. We use a trick similar to the one above to reduce $\overline{\mathsf{A_{TM}}}$ to A. First, suppose, to get a contradiction, that R is a TM that recognizes A. We define a TM $N(M, w)$ as follows,

On input u...

1 **if** $u = 0$ or 00 **then** ACCEPT.
2 Run M on w.
3 **if** M ever accepts **then** ACCEPT.
4 **if** M ever rejects **then** REJECT.

We then have

$$\mathcal{L}(N_{M,w}) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \{0, 00\} & \text{otherwise} \end{cases}$$

We can then recognize $\overline{\mathsf{A_{TM}}}$ by the following TM.

On input TM M and string w...

1 Run R on $N_{M,w}$.
2 **if** R accepts **then** ACCEPT.

To see that this TM would correctly recognize $\overline{\mathsf{A_{TM}}}$, observe that

$$\begin{aligned} M \text{ does not accept } w &\iff \mathcal{L}(N_{M,w}) = \{0, 00\} \\ &\iff N_{M,w} \text{ accepts exactly two strings} \\ &\iff R \text{ accepts } N_{M,w} \\ &\iff \text{the above TM accepts } \langle M, w \rangle \end{aligned}$$

which completes our proof. □

3.3. $A = \{\langle M \rangle : M \text{ is a TM and } M \text{ halts on input } \varepsilon\}$ is recognizable but not decidable.

Proof. Below is a simple recognizer for A.

On input TM M...
1 Run M on ε.
2 **if** M ever accepts **then** ACCEPT.
3 **if** M ever rejects **then** ACCEPT.

To prove undecidability, we present a reduction from $\mathsf{A_{TM}}$. For concision, we will omit the standard procedure to use the reduction as a subroutine to decide $\mathsf{A_{TM}}$.
Define a TM $N_{M,w}$, which depends on another TM M and a string w, as follows.

On input u...
1 Run M on w.
2 **if** M ever accepts **then** ACCEPT.
3 **if** M ever rejects **then** keep running in an infinite loop.

Note that $\mathcal{L}(N_{M,w})$ is either Σ^* or $\varnothing$. Our reduction is then as follows.

On input $\langle M, w\rangle$...
1 Output $N_{M,w}$.

This is a correct reduction since

$$\begin{aligned} M \text{ accepts } w &\implies \mathcal{L}(N_{M,w}) = \Sigma^* \\ &\implies N_{M,w} \text{ accepts (and halts on) } \varepsilon \end{aligned}$$

and

$$\begin{aligned} M \text{ does not accept } w &\implies N_{M,w} \text{ loops on all inputs} \\ &\implies N_{M,w} \text{ loops on } \varepsilon \end{aligned}$$

completing our proof. □

3.4. $A = \{\langle M_1, M_2\rangle : M_1, M_2 \text{ are TMs over } \{0,1\} \text{ and } \mathcal{L}(M_1) = \overline{\mathcal{L}(M_2)}\}$ is unrecognizable.

Proof. We reduce $\overline{\mathsf{A_{TM}}}$ to A as follows. First, define $N_{M,w}$ as follows.

On input u...
1 Run M on w.
2 **if** M accepts **then** ACCEPT.
3 **if** M rejects **then** REJECT.

Note that

$$\mathcal{L}(N_{M,w}) = \begin{cases} \varnothing & \text{if } \langle M, w\rangle \in \overline{\mathsf{A_{TM}}} \\ \Sigma^* & \text{otherwise} \end{cases}$$

Our reduction is then as follows.

On input $\langle M, w \rangle \ldots$

- Construct a TM N such that $\mathcal{L}(N) = \Sigma^*$.
- Output $\langle N, N_{M,w} \rangle$.

Observe that $\langle M, w \rangle \in \overline{\mathsf{A_{TM}}} \iff \mathcal{L}(N_{M,w}) = \varnothing = \overline{\mathcal{L}(N)}$, thus this is a correct reduction. $\square$

4. **($\mathsf{EQ_{CFG}}$ is co-recognizable)**

Proof. Recall that $\mathsf{A_{CFG}}$ is decidable, so let D be a decider for it. We will use D to decide $\overline{\mathsf{EQ_{CFG}}}$ as follows.

On input $\langle G_1, G_2 \rangle \ldots$

for each string $w \in \Sigma^*$ in canonical order **do**
  Run D on $\langle G_1, w \rangle$.
  Run D on $\langle G_2, w \rangle$.
  if D accepts one and rejects the other **then** ACCEPT.

If $\mathcal{L}(G_1) \neq \mathcal{L}(G_2)$, then there must be a string w at which they differ, so that our TM will detect this after finite time and accept. On the other hand, if $\mathcal{L}(G_1) = \mathcal{L}(G_2)$, then our TM will loop forever. Hence, it correctly recognizes $\overline{\mathsf{EQ_{CFG}}}$. $\square$