University of California, San Diego
Department of Computer Science and Engineering
Course: CSE 105 – Theory of Computability
2004 Summer Session I
July 29, 2004

Instructor: Adriana Palacio
TAs: Ragesh Jaiswal
Vadim Lyubashevsky
Saurabh Panjwani

Problem Set 3 Solutions

1. Prove that the following language is co-recognizable:

$$A = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = \overline{L(G_2)} \}$$

We show that A is co-recognizable by constructing a recognizer for its complement, namely,

$$\overline{A} = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) \neq \overline{L(G_2)} \}$$

Want: A TM M that recognizes $\overline{A}$, i.e., such that for all CFGs G_1, G_2:

$$\begin{array}{lcll} L(G_1) \neq \overline{L(G_2)} & \Longrightarrow & M(\langle G_1, G_2 \rangle) & \textit{accepts} \\ L(G_1) = \overline{L(G_2)} & \Longrightarrow & M(\langle G_1, G_2 \rangle) & \textit{rejects or does not halt} \end{array}$$

Construction: Theorem 4.6 says that the following language is decidable:

$$A_{\mathsf{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates } w \}$$

Let M_{CFG} be a TM that decides A_{CFG}. The following TM recognizes $\overline{A}$.

M = On input $\langle G_1, G_2 \rangle$
  Let Σ be the union of the terminal alphabets of G_1 and G_2
  For all $w \in \Sigma^*$ do
    If $M_{\mathsf{CFG}}(\langle G_1, w \rangle)$ accepts then $\mathsf{acc}_1 \leftarrow 1$ else $\mathsf{acc}_1 \leftarrow 0$ EndIf
    If $M_{\mathsf{CFG}}(\langle G_2, w \rangle)$ accepts then $\mathsf{acc}_2 \leftarrow 1$ else $\mathsf{acc}_2 \leftarrow 0$ EndIf
    If $\mathsf{acc}_1 = \mathsf{acc}_2$ then *accept* EndIf
  EndFor

Correctness:

If $L(G_1) \neq \overline{L(G_2)}$ then there is some $w \in \Sigma^*$ such that 1) $w \in L(G_1)$ and $w \notin \overline{L(G_2)}$ OR 2) $w \notin L(G_1)$ and $w \in \overline{L(G_2)}$. By the definition of the complement of a set, this means that there is some $w \in \Sigma^*$ such that 1) $w \in L(G_1)$ and $w \in L(G_2)$ OR 2) $w \notin L(G_1)$ and $w \notin L(G_2)$. M searches for a string w satisfying this property. When it finds one, it accepts.

If $L(G_1) = \overline{L(G_2)}$ then for all $w \in \Sigma^*$: 1) $w \in L(G_1)$ and $w \in \overline{L(G_2)}$ OR 2) $w \notin L(G_1)$ and $w \notin \overline{L(G_2)}$. By the definition of the complement of a set, this means that for all $w \in \Sigma^*$: 1) $w \in L(G_1)$ and $w \notin L(G_2)$ OR 2) $w \notin L(G_1)$ and $w \in L(G_2)$. M searches for a string w such that 1) $w \in L(G_1)$ and $w \in L(G_2)$ OR 2) $w \notin L(G_1)$ and $w \notin L(G_2)$. It does not find one, so it does not halt.

2. Prove that the following language is co-recognizable:

$$B = \{ \langle G, G_1, G_2 \rangle \mid G, G_1, G_2 \text{ are CFGs and } L(G) = L(G_1) \cap L(G_2) \}$$

We show that B is co-recognizable by constructing a recognizer for its complement, namely,

$$\overline{B} = \{ \langle G, G_1, G_2 \rangle \mid G, G_1, G_2 \text{ are CFGs and } L(G) \neq L(G_1) \cap L(G_2) \}$$

Want: A TM M that recognizes $\overline{B}$, i.e., such that for all CFGs G, G_1, G_2:

$$\begin{array}{lcll} L(G) \neq L(G_1) \cap L(G_2) & \Longrightarrow & M(\langle G, G_1, G_2 \rangle) & \textit{accepts} \\ L(G) = L(G_1) \cap L(G_2) & \Longrightarrow & M(\langle G, G_1, G_2 \rangle) & \textit{rejects or does not halt} \end{array}$$

Construction: Again, we use M_{CFG}. The following TM recognizes $\overline{B}$.

$M =$ On input $\langle G, G_1, G_2 \rangle$
  Let Σ be the union of the terminal alphabets of G, G_1 and G_2
  For all $w \in \Sigma^*$ do
    If $M_{\mathsf{CFG}}(\langle G, w \rangle)$ accepts then $\mathsf{acc} \leftarrow 1$ else $\mathsf{acc} \leftarrow 0$ EndIf
    If $M_{\mathsf{CFG}}(\langle G_1, w \rangle)$ accepts and $M_{\mathsf{CFG}}(\langle G_2, w \rangle)$ accepts then $\mathsf{acc}_{\text{int}} \leftarrow 1$
    else $\mathsf{acc}_{\text{int}} \leftarrow 0$ EndIf
    If $\mathsf{acc} \neq \mathsf{acc}_{\text{int}}$ then *accept* EndIf
  EndFor

Correctness: Very similar to the argument used in Problem 1.

---

3. Let L be a recognizable language that is not decidable. Prove that for any TM M that recognizes L there are infinitely many strings on which M does not halt.

Given: A recognizable language L, over some alphabet Σ, that is not decidable.

Want: To show that **for any** TM M that recognizes L **there are** infinitely many strings on which M does not halt.

Proof: By contradiction. Assume that **there exists** a TM M that recognizes L for which **there are not** infinitely many strings on which M does not halt. Then there are only a finite number of strings on which M does not halt. We will construct a TM that decides L. This contradicts the fact that L is not decidable.

Let $S = \{ w \in \Sigma^* \mid M(w) \text{ does not halt} \}$. Since this set is finite, a TM can easily test whether some string s is in S. (It just goes through all the strings in S and checks if they're equal to s.) The following TM, which has the description of S hardwired, decides L.

$M' =$ On input a string w
  If $w \in S$ then *reject*
  else
    Run $M(w)$
    If it accepts then *accept* else *reject* EndIf
  EndIf

Notice that the set S can be "hard-coded" into M' because S is finite. Also notice that M' calls M only on inputs on which M halts; thus M' always halts, i.e., M' is a decider.

---

4. Prove that the following language is undecidable:

$$A = \{ \langle G_1, G_2 \rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = \overline{L(G_2)} \}$$

Theorem 4.10 says that the following language is undecidable:

$$ALL_{\mathsf{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \}$$

We will use this fact in our proof.

Proof: By contradiction. Assume that A is decidable. Then there exists a TM M_A that decides it. We will use M_A to construct a TM that decides ALL_{CFG}. This contradicts the fact that ALL_{CFG} is undecidable.

Since M_A decides A, for all CFGs G_1, G_2:

$$\begin{array}{lcll} L(G_1) = \overline{L(G_2)} & \Longrightarrow & M_A(\langle G_1, G_2 \rangle) & \textit{accepts} \\ L(G_1) \neq \overline{L(G_2)} & \Longrightarrow & M_A(\langle G_1, G_2 \rangle) & \textit{rejects} \end{array}$$

We want to construct a TM M_{ALL} such that for all CFGs G:

$$\begin{array}{lcll} L(G) = \Sigma^* & \Longrightarrow & M_{\mathsf{ALL}}(\langle G \rangle) & \textit{accepts} \\ L(G) \neq \Sigma^* & \Longrightarrow & M_{\mathsf{ALL}}(\langle G \rangle) & \textit{rejects} \end{array}$$

Define M_{ALL} as follows:

M_{ALL} = On input $\langle G \rangle$
Construct a CFG G_2 that generates $\emptyset$
Run $M_A(\langle G, G_2 \rangle)$
If it accepts then *accept* else *reject* EndIf

If $L(G) = \Sigma^*$, then $L(G) = \overline{L(G_2)}$ (because $L(G_2) = \emptyset$ and $\overline{\emptyset} = \Sigma^*$). Thus $\langle G, G_2 \rangle \in A$ and so $M_A(\langle G, G_2 \rangle)$ accepts and thence $M_{\mathsf{ALL}}(\langle G \rangle)$ accepts.

If $L(G) \neq \Sigma^*$, then $L(G) \neq \overline{L(G_2)}$. Therefore, $\langle G, G_2 \rangle \notin A$ and so $M_A(\langle G, G_2 \rangle)$ rejects and thence $M_{\mathsf{ALL}}(\langle G \rangle)$ rejects.

Thus M_{ALL} is indeed a decider for ALL_{CFG}. Contradiction!

---

5. Prove that the following language is undecidable:

$$B = \{ \langle G, G_1, G_2 \rangle \mid G, G_1, G_2 \text{ are CFGs and } L(G) = L(G_1) \cap L(G_2) \}$$

This proof is almost identical to the one for Problem 4.

Proof: By contradiction. Assume that B is decidable. Then there exists a TM M_B that decides it. We will use M_B to construct a TM that decides ALL_{CFG}. This contradicts the fact that ALL_{CFG} is undecidable.

Since M_B decides B, for all CFGs G, G_1, G_2:

$$L(G) = L(G_1) \cap L(G_2) \implies M_B(\langle G, G_1, G_2\rangle) \quad \textit{accepts}$$
$$L(G) \neq L(G_1) \cap L(G_2) \implies M_B(\langle G, G_1, G_2\rangle) \quad \textit{rejects}$$

We want to construct a TM M_{ALL} such that for all CFGs G:

$$L(G) = \Sigma^* \implies M_{\mathsf{ALL}}(\langle G\rangle) \quad \textit{accepts}$$
$$L(G) \neq \Sigma^* \implies M_{\mathsf{ALL}}(\langle G\rangle) \quad \textit{rejects}$$

Define M_{ALL} as follows:

M_{ALL} = On input $\langle G\rangle$
Construct a CFG G_2 that generates Σ^*
Run $M_B(\langle G, G_2, G_2\rangle)$
If it accepts then *accept* else *reject* EndIf

If $L(G) = \Sigma^*$, then since $L(G_2) \cap L(G_2) = \Sigma^*$, $L(G) = L(G_1) \cap L(G_2)$. Therefore, $M_B(\langle G, G_2, G_2\rangle)$ accepts and thence $M_{\mathsf{ALL}}(\langle G\rangle)$ accepts.

If $L(G) \neq \Sigma^*$, then since $L(G_2) \cap L(G_2) = \Sigma^*$, $L(G) = L(G_1) \cap L(G_2)$. Therefore, $M_B(\langle G, G_2, G_2\rangle)$ rejects and thence $M_{\mathsf{ALL}}(\langle G\rangle)$ rejects.

Thus M_{ALL} is indeed a decider for ALL_{CFG}. Contradiction!

---

6. Prove that the following language is undecidable:

$$C = \{\, \langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } M_1(\varepsilon) \text{ halts and } M_2(\varepsilon) \text{ does not halt} \,\}$$

Proof: By contradiction. Assume that C is decidable. Then there exists a TM M_C that decides it. We will use M_C to construct a TM that decides $HALT$. This contradicts the fact that $HALT$ is undecidable.

Since M_C decides C, for all TMs M_1, M_2:

$$M_1(\varepsilon) \text{ halts and } M_2(\varepsilon) \text{ does not halt} \implies M_C(\langle M_1, M_2\rangle) \quad \textit{accepts}$$
$$M_1(\varepsilon) \text{ does not halt or } M_2(\varepsilon) \text{ halts} \implies M_C(\langle M_1, M_2\rangle) \quad \textit{rejects}$$

We want to construct a TM M_{HALT} such that for all $\langle M, w\rangle$:

$$M(w) \text{ halts} \implies M_{\mathsf{HALT}}(\langle M, w\rangle) \quad \textit{accepts}$$
$$M(w) \text{ does not halt} \implies M_{\mathsf{HALT}}(\langle M, w\rangle) \quad \textit{rejects}$$

Consider the following TMs, where M_1 has $\langle M\rangle$ and w hardwired:

M_1 = On input a string x
Run $M(w)$
Accept

M_2 = On input a string x
loop

Define M_{HALT} as follows:

M_{HALT} = On input $\langle M, w\rangle$
Construct M_1 and M_2 as defined above
Run $M_C(\langle M_1, M_2\rangle)$
If it accepts then *accept* else *reject* EndIf

If $M(w)$ halts, then by the definition of M_1, $M_1(\varepsilon)$ accepts. Therefore, $M_1(\varepsilon)$ halts. By the definition of M_2, $M_2(\varepsilon)$ does not halt. Since $M_1(\varepsilon)$ halts and $M_2(\varepsilon)$ does not halt, $M_C(\langle M_1, M_2\rangle)$ accepts and thence $M_{\mathsf{HALT}}(\langle M, w\rangle)$ accepts.

If $M(w)$ does not halt, then by the definition of M_1, $M_1(\varepsilon)$ does not halt. Therefore, $M_C(\langle M_1, M_2\rangle)$ rejects and thence $M_{\mathsf{HALT}}(\langle M, w\rangle)$ rejects.

Thus M_{HALT} is indeed a decider for $HALT$. Contradiction!

---

7. Prove that the following language is undecidable:

$$D = \{\, \langle M\rangle \mid M \text{ is a TM that accepts at most one string which ends in a } \mathtt{0} \,\}$$

Proof: By contradiction. Assume that D is decidable. Then there exists a TM M_D that decides it. We will use M_D to construct a TM that decides $HALT$. This contradicts the fact that $HALT$ is undecidable.

Since M_D decides D, for all TMs M:

$$\begin{array}{lcll} M \text{ accepts at most one string ending in a } \mathtt{0} & \Longrightarrow & M_D(\langle M\rangle) & \textit{accepts} \\ M \text{ accepts more than one string ending in a } \mathtt{0} & \Longrightarrow & M_D(\langle M\rangle) & \textit{rejects} \end{array}$$

We want to construct a TM M_{HALT} such that for all $\langle M, w\rangle$:

$$\begin{array}{lcll} M(w) \text{ halts} & \Longrightarrow & M_{\mathsf{HALT}}(\langle M, w\rangle) & \textit{accepts} \\ M(w) \text{ does not halt} & \Longrightarrow & M_{\mathsf{HALT}}(\langle M, w\rangle) & \textit{rejects} \end{array}$$

Consider the following TM, which has $\langle M\rangle$ and w hardwired:

M_1 = On input a binary string x
Run $M(w)$
Accept

Define M_{HALT} as follows:

M_{HALT} = On input $\langle M, w\rangle$
Construct M_1 as defined above
Run $M_D(\langle M_1\rangle)$
If it accepts then *reject* else *accept* EndIf

If $M(w)$ halts, then by the definition of M_1, $L(M_1) = \{\mathtt{0}, \mathtt{1}\}^*$. Thus M_1 accepts more than one string ending in a $\mathtt{0}$. Therefore, $M_D(\langle M_1\rangle)$ rejects and thence $M_{\mathsf{HALT}}(\langle M, w\rangle)$ accepts.

If $M(w)$ does not halt, then by the definition of M_1, $L(M_1) = \emptyset$. Thus M_1 does not accept any strings ending in a $\mathtt{0}$. This implies that it accepts at most one such string (at most means $\leq$ and 0 is certainly ≤ 1). Therefore, $M_D(\langle M_1\rangle)$ accepts and thence $M_{\mathsf{HALT}}(\langle M, w\rangle)$ rejects.

Thus M_{HALT} is indeed a decider for $HALT$. Contradiction!

---

8. Is the following language decidable or undecidable? Justify your answer (i.e., if E is decidable then construct a decider for it; otherwise, prove that it is undecidable).

$$E = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) \subseteq L(M_2) \}$$

Before thinking about the decidability of the language E, let's try and build a recognizer for it first (this seems like a more plausible task and if we are successful in doing this, we might get a hint about whether E is decidable or not, too). In other words, let's try to construct a TM M which when given as input a string of the form $\langle M_1, M_2 \rangle$ (M_1 and M_2 being TMs themselves) such that $L(M_1)$ is a subset of $L(M_2)$, always accepts and for any other input it either rejects or goes into a loop. In order to check if the language recognized by M_1 is a subset of the language recognized by M_2, what could the machine M possibly do ? Well, to be able to do that it needs to check if *every* string recognized by M_1 is also recognized by M_2. Can the machine M figure out all strings recognized by M_1 or M_2 just by looking at their respective encodings ? It can't. It has to run these machines on various strings to see which strings are recognized by them and which aren't. So the only way in which M could do its task would be to run through *all* possible strings in Σ^* and simulate each of the machines M_1 and M_2 on these strings. If it ever found a string on which M_1 ended up accepting whereas M_2 rejected that string, it would realize that $L(M_1)$ is surely not a subset of $L(M_2)$ and reject immediately. But M_1 and M_2 might loop on certain strings, for which M wouldn't know when to stop simulating them. Worse still, M would have to try infinitely many strings before it could be sure that there's no string in $L(M_1)$ which is not contained in $L(M_2)$. For this, it would have to execute an infinite loop before it could consider halting and returning `accept`.

The above attempt to construct a recognizer for E seems to be taking us nowhere. It appears that the language is not recognizable (or else, we need a radically different approach to be able to build a recognizer for it). And if the language were indeed unrecognizable, would it be possible for it to be decidable ? No. So our best guess at this stage would be that E is undecidable.

Indeed, this is what the case is. Below we use a reducibility argument to establish this.

Claim 0.1 E is undecidable.

Proof: By contradiction. Let us assume that E is decidable. We will show that this assumption leads us to conclude that the language A_{TM} is decidable. Since we already know that A_{TM} is undecidable, this would imply that the assumption we started with must have been false and so E must be undecidable.

Comment: *Before we go into the details of the proof, let us clarify one thing first. Why do we choose A_{TM} as the language for which contradiction is established ? Why couldn't our choice be $HALT_{TM}$ or E_{TM} instead ? Or BTH, for that matter ? We know all these languages to be undecidable so any of them could as well be chosen. Indeed, it is not always clear as to which is the best language to choose from in providing a reducibility argument but sometimes studying the nature of the strings in these languages might help. Every string in A_{TM} is the encoding of some TM and a string w such that M* **accepts** *on input w. The language E consists of strings which are encodings of pairs of TMs M_1 and M_2 such that every string that M_1* **accepts** *is also* **accepted** *by M_2. The*

stress is on the word "accepts". It appears intuitive that a tester for whether any string accepted by a TM M_1 is also accepted by M_2 might be useful in building a tester for whether M_2 accepts a particular string (w) itself. You might also think of choosing the language E_{TM} for reaching the contradiction (since E_{TM} also deals with what strings a TM "accepts") and there indeed is a way of doing the reducibility argument by choosing E_{TM} but we will not discuss it here. $HALT_{TM}$ surely is not a good choice since it deals with whether machines **halt** *or not on a given input while E deals with* **acceptance** *of strings.*

Assume: E is decidable that is there exists a decider R for it. By definition, the TM R has the property that for any TM's M_1 and M_2:

$$\begin{aligned} L(M_1) \subseteq L(M_2) &\implies R(\langle M_1, M_2\rangle)\text{accepts} \\ L(M_1) \not\subseteq L(M_2) &\implies R(\langle M_1, M_2\rangle)\text{rejects} \end{aligned}$$

Want: To show that A_{TM} is decidable. That is, there exists a TM S which decides A_{TM}. Such an S should satisfy the following two conditions for any TM M and string $w \in \Sigma^*$.

$$\begin{aligned} M(w)\text{accepts} &\implies S(\langle M, w\rangle)\text{accepts} \\ M(w)\text{rejects or loops} &\implies S(\langle M, w\rangle)\text{rejects} \end{aligned}$$

Construction: We construct S using the decider R as follows:

Machine $S(\langle M, w\rangle)$
Construct a TM M_1 such that M_1 accepts only the string w and rejects all other inputs.
Let M_2 be the same as M.
Run R on input $\langle M_1, M_2\rangle$.
If it accepts, accept. Else reject.

Notice that it is easy to write up the code for a TM M_1 which accepts only the string w (and rejects on all other inputs) and this can indeed be done by a Turning machine (in this case S) itself. M_1 simply has w hard-wired into its code; for any input x it checks if $x = w$; if so, it accepts and otherwise it rejects.

Proof of Correctness: In order to prove that our construction works as desired, we need to prove two things:

(a) If $\langle M, w\rangle \in A_{TM}$, then $S(\langle M, w\rangle)$ accepts. If $\langle M, w\rangle$ is in the language A_{TM}, it means that $w \in L(M)$. This can also be written as $\{w\} \subseteq L(M)$. By construction, $L(M_1) = \{w\}$ and $L(M_2) = L(M)$ and so in this case $L(M_1) \subseteq L(M_2)$, which means $\langle M_1, M_2\rangle \in E$ and so $R(\langle M_1, M_2\rangle)$ accepts. Thus, S also accepts on input $\langle M, w\rangle$.

(b) If $\langle M, w\rangle \notin A_{TM}$, then $S(\langle M, w\rangle)$ rejects. If $\langle M, w\rangle$ is not in the language A_{TM}, it means that $w \notin L(M)$. This can also be written as $\{w\} \not\subseteq L(M)$. Again, from our construction we know that $L(M_1) = \{w\}$ and $L(M_2) = L(M)$ and so in this case $L(M_1) \not\subseteq L(M_2)$, which means $\langle M_1, M_2\rangle \notin E$ and so $R(\langle M_1, M_2\rangle)$ rejects. Thus, S also rejects on input $\langle M, w\rangle$.

We conclude that S is a decider for the language A_{TM} and this contradicts the fact that A_{TM} is undecidable.