CONTENTS

1 Solutions to Practice Exam 3, 2009
2 Solutions to additional exercises: Decidability and Reductions, 2015

NOTES AND ERRATA

• In practice exam problem 2, \( H_{TM} \) is the Halting Problem, which Sipser calls \( HALT_{TM} \).

• In the solution to practice exam problem 2(e), \( \{ \langle M \rangle \mid L(M) \neq \emptyset \} \) should be circled. This language is \( \overline{E_{TM}} \), which is Turing-recognizable. Sipser gives a Turing machine to recognize it in the solution to problem 4.5 (p. 213). It is not recursive (i.e., decidable) since then \( E_{TM} \) would also be decidable, and it is proven (Theorem 5.2, p. 241) that \( E_{TM} \) is undecidable.

• The solution to practice exam problem 5 is confusing; I recommend studying the reductions in the previous handout and decidability problem 2 instead.

• In decidability problem 2, steps (d) and (e) should be numbered 2. and 3. instead. Note that when \( M \) accepts \( w \), \( L(M_1) = \{01\} \), which is not in \( L \), since it doesn’t include 10. When \( M \) rejects \( w \), \( L(M_1) = \emptyset \), which is in \( L \).
1. (10) Provided below is a Turing machine for the language $a^n b^n$. Note that “B” is a blank and the input alphabet is $\{a, b\}$.

   ![Turing Machine Diagram]

   Modify the above to create a TM for the language $a^n b^{2n}$, and draw a diagram for the resulting TM.

   The solution adds a new state, $q_5$, to handle erasing an extra $b$ for each $a$. The transition from $Q_2$ to $q_3$ is deleted and two new transitions are added: $\delta(q_2, b) = (q_5, B, L)$ and $\delta(q_5, b) = (q_3, B, L)$.

2. (10) Each of the “proofs” below claims to prove a conjecture. The conjectures might be true or false, but both of the proofs are bad. For each proof, explain clearly why the provided proof does not satisfactorily prove the given conjecture. Your answer should identify a fundamental flaw in the proof (for example, by explaining what it actually proves instead of the conjecture), not a minor issue like lack of sufficient detail.

   a. **Conjecture**: The “BusyBee” language, defined below, is undecidable:

   $$L_{BusyBee} = \{(M, w, k)\} \text{ where } M \text{ describes a Turing machine, and } k \text{ is the number of different states } M \text{ enters before halting on input } w. \text{ (Note that } q_{accept} \text{ and } q_{reject} \text{ are counted as states for the number of different states count.)}$$

   **Proof by Reduction**: We prove that $L_{BusyBee}$ is undecidable by reducing $H_{TM}$ to it. Construct $M_{BusyBee}$ given a machine $M_{HALT}$ that decides $H_{TM}$:

   $$M_{BusyBee} =$$
Simulate $M_{HALT}$ on input $\langle M, w \rangle$. If $M_{HALT}$ accepts, simulate $M$ on $w$. On a second tape, list the states of $M$, and add a mark on each state that is entered. Count the number of marked states on the second tape. If the number matches $k$, accept. Otherwise, reject (without simulating $M$ on $w$ since it does not halt).

Since building $M_{BusyBee}$ requires $M_{HALT}$, which we know does not exist, this proves that $L_{BusyBee}$ is undecidable.

**Answer:** The proof does the reduction in the wrong direction. It shows that you can decide $L_{BusyBee}$ if you can decide $H_{TM}$ (that is, that $L_{BusyBee}$ is not harder than $H_{TM}$), but it does not show the conjecture which is that $L_{BusyBee}$ is undecidable. To show that, we would need a reduction that shows how to solve $H_{TM}$ if you were given a machine that can solve $L_{BusyBee}$.
b. **Conjecture:** The set of languages that can be recognized by a deterministic pushdown automaton is equivalent to the set of languages that can be recognized by a Turing machine.

**Proof by Simulation.** We prove the conjecture by showing how to simulate any DPDA with a TM. . . . (unnecessary details elided) . . . Thus, we can simulate a DPDA with a TM. This proves a DPDA is equivalent to a TM.

**Answer:** The proof only covers one direction: it shows a DPDA can be simulated by a TM. To show the sets of languages they recognize are equivalent, we need to show both directions (in this case, this would be impossible, since it is not possible for a DPDA to simulate a TM). Hence, what is actually proven here is that a DPDA is no more powerful than a TM (which is not surprising, since the set of languages that can be recognized by a DPDA is a subset of the Turing-recognizable languages).
3. (15) Consider the Turing Machine

\[ M = (\{q_0, q_1, q_f\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \]

where \( \delta \) consists of the following rules:

- \( \delta(q_0, 0) = (q_0, \sqcup, R) \)
- \( \delta(q_0, 1) = (q_1, \sqcup, R) \)
- \( \delta(q_1, 0) = (q_1, \sqcup, R) \)
- \( \delta(q_1, \sqcup) = (q_{\text{accept}}, \sqcup, R) \)
- \( \delta((q_1, 0) = (q_{\text{reject}}, \sqcup, R) \)

a. Informally but clearly describe the language \( L(M) \).

**Answer:** \( L(M) = \{0^*1^+\} \), that is, the language consisting of 0 or more 0’s followed by at least one 1.

b. Give a string in this language that is of length at least 4, and show the sequence of moves of this TM when this string is the input to \( M \).

**Answer:** Consider the string \( s = 0111 \). The following sequence of moves recognizes \( s \):

\[
\begin{align*}
q_0 & \quad 0111 \\
\sqcup q_0 & \quad 111 \\
\sqcup \sqcup q_1 & \quad 11 \\
\sqcup \sqcup \sqcup q_1 & \\
\sqcup \sqcup \sqcup \sqcup q_1 & \\
\sqcup \sqcup \sqcup \sqcup q_f &
\end{align*}
\]
4. (30) Answer each of the following as indicated.

a. Indicate whether the following is true or false and fully justify your answer: If language $A$ is a subset of language $B$, and $A$ is recursively enumerable, then $B$ is recursively enumerable.

**Answer:** False. Let $A = \emptyset$, the empty set. Then $A$ is recursively enumerable (it is accepted by a TM that rejects everything) but $A \subseteq B$ for any language $B$.

But we know that there are languages that are not recursively enumerable.

b. Indicate whether the following is true or false and fully justify your answer: A non-deterministic TM accepts a string $w$ if every possible computation on the input leads to an accepting state.

**Answer:** False. A non-deterministic TM accepts $w$ if at least one computation on $w$ leads to an accepting state.

c. Indicate whether the following is true or false and fully justify your answer: The computational path of a TM on an input $w$ always either halts accepting, halts rejecting, or enters a never halting loop.

**Answer:** True. The only two ways for a TM to halt are by accepting and rejecting. If the TM does not reject or accept, it never halts, i.e., enters a never halting loop.

d. Circle the number to the left of each of the languages below that is recursive but not context-free.

(1) $\{wcw|w \in \{a, b, \}^*\}$

(2) $\{a^{2n}b^n c^{2n}|n \geq 0\}$

e. Circle the number to the left of each of the languages below that is recursively enumerable (Turing-recognizable) but not recursive.

(1) $\{\langle M, w \rangle | \text{TM } M \text{ accepts input } w\}$

f. Circle the number to the left of each of the languages below that is not recursively enumerable (non-Turing-recognizable).

(1) $\{\langle M_1, M_2 \rangle | L(M_1) \cap L(M_2) = \emptyset\}$
5. Consider the following languages:

\[ X = \{ \langle M \rangle \mid M \text{ is a TM that always accepts} \} \]

\[ Y = \{ \langle M, w \rangle \mid M \text{ is a TM that does not accept } w \} \]

Let \( R_X \) be a Turing machine recognizing \( X \) (if it exists). Consider TM \( R_Y \) defined as follows:

On input \( \langle M, w \rangle \):

- Compute the description \( \langle M, w \rangle \) of the Turing machine \( M_w \) that acts as follows: on input binary integer \( x \), \( M_w \) runs \( M \) on input \( w \) for \( x \) steps. If \( M \) has reached its accept state within the \( x \) steps, then reject; otherwise, accept.
- Run \( R_X \) on input \( \langle M_w \rangle \), and accept if it accepts, reject if it rejects.

a. (15 points) Suppose \( R_X \) exists. What language does \( R_Y \) recognize? Justify your answer.

**Answer:** \( R_X \) recognizes \( Y \). Suppose \( \langle M, w \rangle \in Y \). Then \( M \) does not accept \( w \), no matter how many steps it runs for. Then for all inputs \( x \), \( M_w \) will accept: it will run \( M \) on \( w \) for \( x \) steps and will see that \( M \) has not accepted. Then \( \langle M_w \rangle \in X \), so \( R_X \) will accept. Then \( R_Y \) accepts.

Suppose \( \langle M, w \rangle \notin Y \). Then \( M \) accepts \( w \) after some \( t \) steps. Then for \( x \geq t \), \( M_w \) rejects on input \( x \). Then \( \langle M_w \rangle \notin X \). Therefore, \( R_X \) will not accept. The \( R_Y \) will not accept.

b. (5 points) Using part (a), show that if \( Y \) is not Turing recognizable, then neither is \( X \).

**Answer:** Suppose \( X \) is Turing recognizable. Then let \( R_X \) be a TM recognizing \( X \). Then the TM \( R_Y \) described in (a) recognizes \( Y \). This is a contradiction if \( Y \) is not Turing recognizable.

c. (5 points) Using part (a), show that if \( Y \) is not decidable, then neither is \( X \).

**Answer:** If \( R_X \) is a decider, then \( R_Y \) is also a decider because the behavior of \( R_Y \) is determined by the behavior of \( R_X \). So if \( X \) is decidable, so is \( Y \). Then if \( Y \) is undecidable, so is \( X \).

d. (10 points) Using the fact that \( A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \} \) is Turing-recognizable and undecidable, or any other material covered, prove that \( Y \) is not Turing-recognizable.

**Answer:** Note that \( Y = \overline{A_{TM}} \), and in class we saw that \( \overline{A_{TM}} \) is undecidable but Turing recognizable. We also saw that the complement of a language that is both undecidable and Turing recognizable cannot be Turing recognizable.
3 Solutions

1. Determine whether the following problems are decidable or not:

   a. Is a number $m$ prime?

   This problem is decidable because there is an algorithm which, after a finite number of steps, will halt and give us a definitive answer to the question. Here is the algorithm:
   - On input $m$:
     * Divide the number $m$ by all the numbers between 2 and $\sqrt{m}$, starting from 2.
     * If any of these numbers produces a remainder zero, reject.
     * Otherwise, accept.
   The algorithm is guaranteed to halt with a definitive yes or no answer because the set of numbers between 2 and $\sqrt{m}$ is finite and therefore bounded.

   b. Given a regular language $L$ and string $w$, is $w \in L$?

   Construct a DFA that accepts $L$ and check if $w$ is accepted. This algorithm is guaranteed to terminate in a finite number of steps, namely, $|w|$ steps, and give a yes or no answer depending on whether it is in a final state at the end of the computation. So this problem is decidable.

2. Prove that the language $L = \{ (M) | M$ is a TM, and $M$ accepts string $w$ whenever it accepts string $w^R \}$ is undecidable.

   Solution: reduction from $A_{TM}$.

   Basic idea: given $M$ and $w$ as an input, construct another machine $\langle M_1 \rangle$ such that
   - if $M$ accepts $w$, then $L(M_1)$ is not empty, and
   - if $M$ doesn’t accept $w$, then $L(M_1)$ is empty.

   The code of the machine $M_1$ (created by the “code converter”) was passed as an input to the (hypothetical) machine for $E_{TM}$. If $E_{TM}$ accepts $\langle M_1 \rangle$, this means that $L(M_1)$ is empty, and so, by the above, $M$ doesn’t accept $w$. Conversely, if $E_{TM}$ rejects $\langle M_1 \rangle$, this means that $L(M_1)$ is not empty, and so, by the above, $M$ accepts $w$.

   Here we’re going to do exactly the same thing, except the machine $M_1$, the code of which we will construct, will behave slightly differently, so that we can use the decider for $L$, instead of the decider for $E_{TM}$.

   Back to the solution of our problem: As usual, we assume that $L$ is decidable, and let $M_L$ be the Turing machine that decides $L$. We’re going to construct a TM that decides $A_{TM}$. Since $A_{TM}$ is known to be undecidable, this leads to contradiction, proving that $L$ is undecidable.

   The decider for $A_{TM}$ can be built as follows:
   $M_{ATM} = “on input \langle M, w \rangle$:

   1. Construct the code for the following Turing machine $M_1$:

      $M_1 = “on input x,$
(a) Simulate M on w.
(b) If M rejects w, reject.
(c) If M accepts w, accept if x = 01, reject otherwise.
(d) Simulate ML on input \langle M_1 \rangle.
(e) Accept if ML rejects, reject otherwise.

The key point is this: what is the language of \( M_1 \)? It depends on whether M accepts w or not.

3. Show that it is undecidable whether a TM ever writes a particular symbol on the tape.

Assume we have TM M and string w. Construct a new machine \( M_w \). The TM \( M_w \) is programmed to erase its input, write w on the tape, and pass this over to M. If M accepts, then \( M_w \) writes a special symbol, say $ on the tape. Thus if one could answer the question whether \( M_w \) writes $ or not, one would be able to decide \( \text{A}_{TM} \), which is undecidable.

4. Let B be the set of all infinite sequences over \{0,1\}. Show that B is not recursively enumerable (i.e., it is uncountable), using a proof by diagonalization.

Each element in B is an infinite sequence \((b_1, b_2, b_3, \ldots)\), where each \( b_i \in \{0,1\} \). Suppose B is countable. Then we can define a correspondence between \( \mathbb{N} = \{1, 2, 3, \ldots\} \) and B. Specifically, for \( n \in \mathbb{N} \), let each element \( n = (b_{n1}, b_{n2}, b_{n3}, \ldots) \), where \( b_{ni} \) is the ith bit in the nth sequence.

This can be represented in the following grid:

<table>
<thead>
<tr>
<th>n</th>
<th>Bit Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, \ldots))</td>
</tr>
<tr>
<td>2</td>
<td>((b_{21}, b_{22}, b_{23}, b_{24}, b_{25}, \ldots))</td>
</tr>
<tr>
<td>3</td>
<td>((b_{31}, b_{32}, b_{33}, b_{34}, b_{35}, \ldots))</td>
</tr>
<tr>
<td>4</td>
<td>((b_{41}, b_{42}, b_{43}, b_{44}, b_{45}, \ldots))</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Now take the major diagonal of this grid, and complement the result. For example, if

<table>
<thead>
<tr>
<th>n</th>
<th>Bit Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((0, 1, 1, 0, 0, \ldots))</td>
</tr>
<tr>
<td>2</td>
<td>((1, 0, 1, 0, 1, \ldots))</td>
</tr>
<tr>
<td>3</td>
<td>((1, 1, 1, 1, 1, \ldots))</td>
</tr>
<tr>
<td>4</td>
<td>((1, 0, 0, 1, 0, \ldots))</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

then the complement would be \((1, 1, 0, 0, \ldots)\). Thus, this set differs from each sequence in B by at least one bit, and so correspond to any n, which is a contradiction. Hence, B is not recursively enumerable.