CS 181 Homework 5

Sriram Balachandran

TOTAL POINTS

96 / 100

QUESTION 1

1 Question 1 **20 / 20**

✓ **+ 4 pts Not completely incorrect**

Construction

✓ **+ 10 pts Correct**

+ 9 pts Small Mistake

+ 5 pts Flawed

+ 0 pts Incorrect

Proof of correctness

✓ **+ 6 pts Correct**

+ 5 pts Small Mistake

+ 3 pts Flawed / Incomplete

+ 0 pts Incorrect

QUESTION 2

2 Question 2 **19 / 20**

- 0 pts Correct

✓ **- 1 pts Small Mistake in construction**

- 3 pts missing some clarity

- 3 pts Flawed Proof

- 10 pts wrong strategy/partial answer

- 15 pts Error

- 15 pts no answer/ I don't know

no such thing as epsilon should be the empty set instead

QUESTION 3

Question 3 60 pts

3.1 3a **12 / 15**

- 0 pts Correct

✓ **- 3 pts Incorrect input to Turing machine**

- 6 pts Incorrect code for y

- 6 pts Incorrect/partially correct explanation of accept / reject

- 15 pts No answer

3.2 3b **15 / 15**

✓ **- 0 pts Correct**

- 6 pts Does not mention infinitely possible input x

- 12 pts "I Don't Know"

- 15 pts No answer

3.3 3c **30 / 30**

✓ **- 0 pts Correct**

- 6 pts Partially correct use of recursion theorem

- 12 pts Incorrect use of recursion theorem

- 6 pts Partially correct analysis of cases

- 15 pts No clear analysis of cases

- 24 pts "I don't know"

- 30 pts No answer

1. **(20 points).** Prove that the language

$$\mathsf{COMPL}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = \overline{L(M_2)}, \text{ where } M_1 \text{ and } M_2 \text{ are Turing machines}\}$$

is not Turing-recognizable.

Answer:

We will prove $\mathsf{COMPL}_{\mathsf{TM}}$ is not Turing-recognizable by contradiction. Assume there is a recognizer $R(x, y)$ that recognizes $\mathsf{COMPL}_{\mathsf{TM}}$.

Let $L(M_2) = \Sigma^*$, where $\Sigma^* = \{w|w = \{0,1\}^*\}$, and let $z = \langle M_1\rangle$ by the recursion theorem. For $M_1(x)$, we run $R(z, < M_2 >)$. Let us then pose contradictions that arise from every possible return state of R.

If R accepts, then we accept x. Thus $L(M_1) = \Sigma^*$. This is a contradiction since $L(M_1) \neq \overline{L(M_2)}$.

If R rejects, then we can consider $L(M_1) = \epsilon$, where ϵ is the empty set, since M_1 will reject all inputs. This poses a contradiction since $L(M_1) = \overline{L(M_2)}$, which means R should not have rejected.

If R loops forever, the recognizer does not believe $\langle M_1, M_2\rangle$ to be in $\mathsf{COMPL}_{\mathsf{TM}}$. For any input x, M_1 will loop forever, and never accept. Thus $L(M_1) = \epsilon$, which is a contradiction since $L(M_1) = \overline{L(M_2)}$, which means R should have accepted $\langle M_1, M_2\rangle$ instead of looping forever.

Since every possible outcome of a recognizer R for $\mathsf{COMPL}_{\mathsf{TM}}$ resulted in a contradiction for $\langle M_1, M_2\rangle$, $\mathsf{COMPL}_{\mathsf{TM}}$ is not Turing-recognizable.

2. **(20 points).** Define $\mathsf{SUBSET}_{\mathsf{TM}}$ to be the problem of testing whether the set of strings accepted by a Turing machine, say M_1, is also accepted by another Turing machine, say M_2. More formally,

$$\mathsf{SUBSET}_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) \subseteq L(M_2)\}.$$

Show that $\mathsf{SUBSET}_{\mathsf{TM}}$ is undecidable.

Answer:

We will prove $\mathsf{SUBSET}_{\mathsf{TM}}$ is undecidable by contradiction. Assume there exists a $D(x, y)$ that decides $\mathsf{SUBSET}_{\mathsf{TM}}$.

Let $L(M_2) = \{1000101, 110100100\}$, and let $z = \langle M_1\rangle$ by the recursion theorem. For $M_1(x)$, we run $D(z, < M_2 >)$. Let us then pose contradictions that arise from every possible return state of D.

If D accepts, then we accept x. Since x is an arbitrary input, $L(M_1) = \Sigma^*$, which is not guaranteed to be a subset of $L(M_2)$ (x could be 0, which would violate the subset condition).

If D rejects, then we would reject all x. Thus $L(M_2) = \epsilon$. However this poses a contradiction since $\epsilon \subseteq L(M_2)$, which means D shouldn't have rejected.

Since every possible outcome of a recognizer R for $\mathsf{SUBSET}_{\mathsf{TM}}$ resulted in a contradiction for $\langle M_1, M_2\rangle$, $\mathsf{SUBSET}_{\mathsf{TM}}$ is not Turing-decidable.

3. **(60 points)** We define a new notion called a "certified" language. A language L over the alphabet $\{0, 1\}$ is called "certified" if there exists a Turing machine M satisfying the following conditions:

   - For all $x \in L$, there exists a $y \in \{0, 1\}^*$ such that $M(x, y)$ accepts.

1 Question 1 **20 / 20**

✓ **+ 4 pts** **Not completely incorrect**

Construction

✓ **+ 10 pts** **Correct**

+ 9 pts Small Mistake

+ 5 pts Flawed

+ 0 pts Incorrect

Proof of correctness

✓ **+ 6 pts** **Correct**

+ 5 pts Small Mistake

+ 3 pts Flawed / Incomplete

+ 0 pts Incorrect

1. **(20 points).** Prove that the language

$$\mathsf{COMPL_{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = \overline{L(M_2)}, \text{ where } M_1 \text{ and } M_2 \text{ are Turing machines}\}$$

   is not Turing-recognizable.

   Answer:

   We will prove $\mathsf{COMPL_{TM}}$ is not Turing-recognizable by contradiction. Assume there is a recognizer $R(x, y)$ that recognizes $\mathsf{COMPL_{TM}}$.

   Let $L(M_2) = \Sigma^*$, where $\Sigma^* = \{w|w = \{0, 1\}^*\}$, and let $z = \langle M_1\rangle$ by the recursion theorem. For $M_1(x)$, we run $R(z, < M_2 >)$. Let us then pose contradictions that arise from every possible return state of R.

   If R accepts, then we accept x. Thus $L(M_1) = \Sigma^*$. This is a contradiction since $L(M_1) \neq \overline{L(M_2)}$.

   If R rejects, then we can consider $L(M_1) = \epsilon$, where ϵ is the empty set, since M_1 will reject all inputs. This poses a contradiction since $L(M_1) = \overline{L(M_2)}$, which means R should not have rejected.

   If R loops forever, the recognizer does not believe $\langle M_1, M_2\rangle$ to be in $\mathsf{COMPL_{TM}}$. For any input x, M_1 will loop forever, and never accept. Thus $L(M_1) = \epsilon$, which is a contradiction since $L(M_1) = \overline{L(M_2)}$, which means R should have accepted $\langle M_1, M_2\rangle$ instead of looping forever.

   Since every possible outcome of a recognizer R for $\mathsf{COMPL_{TM}}$ resulted in a contradiction for $\langle M_1, M_2\rangle$, $\mathsf{COMPL_{TM}}$ is not Turing-recognizable.

2. **(20 points).** Define $\mathsf{SUBSET_{TM}}$ to be the problem of testing whether the set of strings accepted by a Turing machine, say M_1, is also accepted by another Turing machine, say M_2. More formally,

$$\mathsf{SUBSET_{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) \subseteq L(M_2)\}.$$

   Show that $\mathsf{SUBSET_{TM}}$ is undecidable.

   Answer:

   We will prove $\mathsf{SUBSET_{TM}}$ is undecidable by contradiction. Assume there exists a $D(x, y)$ that decides $\mathsf{SUBSET_{TM}}$.

   Let $L(M_2) = \{1000101, 110100100\}$, and let $z = \langle M_1\rangle$ by the recursion theorem. For $M_1(x)$, we run $D(z, < M_2 >)$. Let us then pose contradictions that arise from every possible return state of D.

   If D accepts, then we accept x. Since x is an arbitrary input, $L(M_1) = \Sigma^*$, which is not guaranteed to be a subset of $L(M_2)$ (x could be 0, which would violate the subset condition).

   If D rejects, then we would reject all x. Thus $L(M_2) = \epsilon$. However this poses a contradiction since $\epsilon \subseteq L(M_2)$, which means D shouldn't have rejected.

   Since every possible outcome of a recognizer R for $\mathsf{SUBSET_{TM}}$ resulted in a contradiction for $\langle M_1, M_2\rangle$, $\mathsf{SUBSET_{TM}}$ is not Turing-decidable.

3. **(60 points)** We define a new notion called a "certified" language. A language L over the alphabet $\{0, 1\}$ is called "certified" if there exists a Turing machine M satisfying the following conditions:

   - For all $x \in L$, there exists a $y \in \{0, 1\}^*$ such that $M(x, y)$ accepts.

2 Question 2 **19 / 20**

- **- 0 pts** Correct
- ✓ **- 1 pts Small Mistake in construction**
- **- 3 pts** missing some clarity
- **- 3 pts** Flawed Proof
- **- 10 pts** wrong strategy/partial answer
- **- 15 pts** Error
- **- 15 pts** no answer/ I don't know

no such thing as epsilon should be the empty set instead

- For all $x \notin L$ and for all $y \in \{0,1\}^*$, $M(x,y)$ rejects.
  (We think of y as being the "certificate" that allows x to be in the language.)

(a) **(15 points).** Let Halt_ϵ denote the language of all the Turing machines which halt on input ϵ. More formally,

$$\text{Halt}_\epsilon = \{\langle N\rangle \mid N \text{ halts on } \epsilon\},$$

where $\langle N\rangle$ denotes the code of the machine N. Show that Halt_ϵ is a certified language.
Hint: Think about how the input y could be made to relate to whether or not a machine "halts".

Answer:

We will construct a certifier $C(x,y)$ that certifies Halt_ϵ as follows. Let x be the description of a Turing Machine we are trying to certify, and let y be some binary string. The certifier will then run the Turing machine that x describes with input $_\epsilon$ for $val(y)$ steps, where $val(y)$ is equal to the integer that the binary string y represents.

If the machine halts within $val(y)$ steps, then C accepts. Since $val(y)$ will always be some finite integer, we know that C will always finish running and accept or reject its input.

If the machine does not halt within $val(y)$ steps, then C will reject.

To prove correctness, consider the description of a turing machine $N \in \text{Halt}_\epsilon$. Since we know that it is in Halt_ϵ, we know that after some finite number of steps, the machine will eventually halt on input ϵ, meaning that all N in Halt_ϵ is guaranteed to have a certificate y such that $C(N,y)$ will accept.

Then consider $M \notin \text{Halt}_\epsilon$. Since we know M will not halt on input ϵ, absolutely no binary string y will certify M, and C will always reject M.

(b) **(15 points).** Explain why your method does not work if you try to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N\rangle \mid N \text{ halts on all inputs}\}$ is a certified language.

Answer:

The method used on the previous certifier worked because it only had to check if the provided machine halted within $val(y)$ steps on a finite number of inputs $(|\{\epsilon\}| = 1)$. However the description of Halt_{all} requires that the machine halts for all inputs. This means that if y was the certificate provided to our certifier, it would have to check that the provided input machine halts within $val(y)$ steps for all $w \in \{0,1\}^*$, which is a set of infinite size. The Certifier would never accept or reject any machine run with any input, which does not satisfy the conditions of being a certifier.

(c) **(30 points).** Use the recursion theorem to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N\rangle \mid N \text{ halts on all inputs}\}$ is not a certified language.

Answer:

We will prove that Halt_{all} is not a certified language by contradiction. Assume $C(x,y)$ certifies Halt_{all}. Let us then consider $N(x)$, where N is some Turing Machine, and $x \in \{0,1\}^*$. Let $z = \langle N\rangle$ by the recursion theorem. Then, $\forall bin(i) \in [0, |x|]$, where $bin(i)$ is the binary representation of some integer i, run $C(z,i)$.

If C accepts for any i, then we loop. This causes a contradiction since C accepting means that it should've halted on input x.

If C rejects all i, we accept the input x. If $N \notin \text{Halt}_{all}$, it is clear this will immediately cause a cause a contradiction, since we will always halt on all x, which is a contradiction.

3.1 3a 12 / 15

- **- 0 pts** Correct
- ✓ **- 3 pts Incorrect input to Turing machine**
- **- 6 pts** Incorrect code for y
- **- 6 pts** Incorrect/partially correct explanation of accept / reject
- **- 15 pts** No answer

- For all $x \notin L$ and for all $y \in \{0,1\}^*$, $M(x,y)$ rejects.
  (We think of y as being the "certificate" that allows x to be in the language.)

(a) **(15 points).** Let Halt_ϵ denote the language of all the Turing machines which halt on input ϵ. More formally,

$$\text{Halt}_\epsilon = \{\langle N \rangle \mid N \text{ halts on } \epsilon\},$$

where $\langle N \rangle$ denotes the code of the machine N. Show that Halt_ϵ is a certified language.
Hint: Think about how the input y could be made to relate to whether or not a machine "halts".

Answer:

We will construct a certifier $C(x,y)$ that certifies Halt_ϵ as follows. Let x be the description of a Turing Machine we are trying to certify, and let y be some binary string. The certifier will then run the Turing machine that x describes with input $_\epsilon$ for $val(y)$ steps, where $val(y)$ is equal to the integer that the binary string y represents.

If the machine halts within $val(y)$ steps, then C accepts. Since $val(y)$ will always be some finite integer, we know that C will always finish running and accept or reject its input.

If the machine does not halt within $val(y)$ steps, then C will reject.

To prove correctness, consider the description of a turing machine $N \in \text{Halt}_\epsilon$. Since we know that it is in Halt_ϵ, we know that after some finite number of steps, the machine will eventually halt on input ϵ, meaning that all N in Halt_ϵ is guaranteed to have a certificate y such that $C(N,y)$ will accept.

Then consider $M \notin \text{Halt}_\epsilon$. Since we know M will not halt on input ϵ, absolutely no binary string y will certify M, and C will always reject M.

(b) **(15 points).** Explain why your method does not work if you try to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N \rangle \mid N \text{ halts on all inputs}\}$ is a certified language.

Answer:

The method used on the previous certifier worked because it only had to check if the provided machine halted within $val(y)$ steps on a finite number of inputs ($|\{\epsilon\}| = 1$). However the description of Halt_{all} requires that the machine halts for all inputs. This means that if y was the certificate provided to our certifier, it would have to check that the provided input machine halts within $val(y)$ steps for all $w \in \{0,1\}^*$, which is a set of infinite size. The Certifier would never accept or reject any machine run with any input, which does not satisfy the conditions of being a certifier.

(c) **(30 points).** Use the recursion theorem to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N \rangle \mid N \text{ halts on all inputs}\}$ is not a certified language.

Answer:

We will prove that Halt_{all} is not a certified language by contradiction. Assume $C(x,y)$ certifies Halt_{all}. Let us then consider $N(x)$, where N is some Turing Machine, and $x \in \{0,1\}^*$. Let $z = \langle N \rangle$ by the recursion theorem. Then, $\forall bin(i) \in [0, |x|]$, where $bin(i)$ is the binary representation of some integer i, run $C(z,i)$.

If C accepts for any i, then we loop. This causes a contradiction since C accepting means that it should've halted on input x.

If C rejects all i, we accept the input x. If $N \notin \text{Halt}_{all}$, it is clear this will immediately cause a cause a contradiction, since we will always halt on all x, which is a contradiction.

3.2 3b **15 / 15**

- ✓ **- 0 pts** **Correct**
- **- 6 pts** Does not mention infinitely possible input x
- **- 12 pts** "I Don't Know"
- **- 15 pts** No answer

- For all $x \notin L$ and for all $y \in \{0,1\}^*$, $M(x,y)$ rejects.
(We think of y as being the "certificate" that allows x to be in the language.)

(a) **(15 points).** Let Halt_ϵ denote the language of all the Turing machines which halt on input ϵ. More formally,

$$\text{Halt}_\epsilon = \{\langle N \rangle \mid N \text{ halts on } \epsilon\},$$

where $\langle N \rangle$ denotes the code of the machine N. Show that Halt_ϵ is a certified language.
Hint: Think about how the input y could be made to relate to whether or not a machine "halts".

Answer:

We will construct a certifier $C(x,y)$ that certifies Halt_ϵ as follows. Let x be the description of a Turing Machine we are trying to certify, and let y be some binary string. The certifier will then run the Turing machine that x describes with input $_\epsilon$ for $val(y)$ steps, where $val(y)$ is equal to the integer that the binary string y represents.

If the machine halts within $val(y)$ steps, then C accepts. Since $val(y)$ will always be some finite integer, we know that C will always finish running and accept or reject its input.

If the machine does not halt within $val(y)$ steps, then C will reject.

To prove correctness, consider the description of a turing machine $N \in \text{Halt}_\epsilon$. Since we know that it is in Halt_ϵ, we know that after some finite number of steps, the machine will eventually halt on input ϵ, meaning that all N in Halt_ϵ is guaranteed to have a certificate y such that $C(N,y)$ will accept.

Then consider $M \notin \text{Halt}_\epsilon$. Since we know M will not halt on input ϵ, absolutely no binary string y will certify M, and C will always reject M.

(b) **(15 points).** Explain why your method does not work if you try to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N \rangle \mid N \text{ halts on all inputs}\}$ is a certified language.

Answer:

The method used on the previous certifier worked because it only had to check if the provided machine halted within $val(y)$ steps on a finite number of inputs ($|\{\epsilon\}| = 1$). However the description of Halt_{all} requires that the machine halts for all inputs. This means that if y was the certificate provided to our certifier, it would have to check that the provided input machine halts within $val(y)$ steps for all $w \in \{0,1\}^*$, which is a set of infinite size. The Certifier would never accept or reject any machine run with any input, which does not satisfy the conditions of being a certifier.

(c) **(30 points).** Use the recursion theorem to prove that the language $\text{Halt}_{\mathsf{all}} = \{\langle N \rangle \mid N \text{ halts on all inputs}\}$ is not a certified language.

Answer:

We will prove that Halt_{all} is not a certified language by contradiction. Assume $C(x,y)$ certifies Halt_{all}. Let us then consider $N(x)$, where N is some Turing Machine, and $x \in \{0,1\}^*$. Let $z = \langle N \rangle$ by the recursion theorem. Then, $\forall bin(i) \in [0, |x|]$, where $bin(i)$ is the binary representation of some integer i, run $C(z,i)$.

If C accepts for any i, then we loop. This causes a contradiction since C accepting means that it should've halted on input x.

If C rejects all i, we accept the input x. If $N \notin \text{Halt}_{all}$, it is clear this will immediately cause a cause a contradiction, since we will always halt on all x, which is a contradiction.

From this construction one should see that for some $N \in \text{Halt}_{all}$, some inputs x will halt. However, since we claim that C certifies Halt_{all}, it is guaranteed that for at least one input, when $|x| >= val(y)$, where y is the certificate of N in the certifier $C(x, y)$, N will not halt and cause a contradiction.

3.3 3c **30 / 30**

✓ **- 0 pts** Correct

- 6 pts Partially correct use of recursion theorem

- 12 pts Incorrect use of recursion theorem

- 6 pts Partially correct analysis of cases

- 15 pts No clear analysis of cases

- 24 pts "I don't know"

- 30 pts No answer