CS 181 Homework 4

Sriram Balachandran

TOTAL POINTS

79 / 80

QUESTION 1

1 Question 1 - Extra Credit **10 / 10**

✓ **- 0 pts** Correct

- 5 pts Partially correct answer

- 5 pts Did not explain how “Move Left One Space” transition is implemented with only “Move Right One Space” transitions and “Move All the Way Left” transitions

- 10 pts Incorrect/No answer

QUESTION 2

2 Question 2 **20 / 20**

✓ **- 0 pts** Correct

- 1 pts typo

- 3 pts unclear detail

- 4 pts minor mistake

- 12 pts significant mistake/so unclear

- 8 pts very unclear/major mistake

- 3 pts Z to N function not bijective: 0 and -1 both map to 1

QUESTION 3

Question 3 50 pts

3.1 3a **10 / 10**

✓ **- 0 pts** Correct

- 1 pts Small Mistake

- 2 pts Imperfect Proof

- 8 pts "I don't know"

- 10 pts Incorrect

3.2 3b **15 / 15**

✓ **- 0 pts** Correct

- 1 pts Small Mistake

- 3 pts Imperfect Proof

- 12 pts "I don't know"

- 15 pts Incorrect

3.3 3c **15 / 15**

✓ **- 0 pts** Correct

- 1 pts Small Mistake

- 3 pts Imperfect Proof

- 12 pts "I don't know"

- 15 pts Incorrect

3.4 3d **9 / 10**

- 0 pts Correct

✓ **- 1 pts** Small Mistake

- 2 pts Imperfect Proof

- 8 pts "I don't know"

- 10 pts Incorrect

Note: This homework set starts with an extra credit problem. The extra credit problem is optional, but is worth extra credit points. Problems 2 and 3 are *not* extra credit problems.

1. **(Optional: 10 Extra Credit Points aka 5 Tokens)** Let us define Fast-Rewind Turing Machines (FRTM). They are similar to Turing Machines, except that the head of a FRTM is not allowed to move left one cell at a time. Instead, the head of the FRTM *must* move left only all the way to the left-hand end of the tape (i.e. the first cell). The head of the FRTM can move right one step at a time, just like the usual Turing Machines. The transition function for the FRTM is of the form $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, FIRST\}$.

   Explain how to convert any Turing Machine into an FRTM.

   A full proof is not needed, you only need to give an explanation of the intuition behind why your transformation works.

   Note that this requires you to show how to implement a "Move Left One Space" transition in a FRTM by using only "Move Right One Space" transitions and "Move All the Way Left" transitions.

   Answer:

   All properties (”move one space right”, infinite tape) of an FRTM are identical to a standard Turing Machine, except for the ”Move left one space” transition. In order to convert a ”Move left one space” transition to a sequence of transitions in an FRTM we do the following:

   1. Mark the current cell

   2. Perform a ”Move All the Way Left” transition

   3. Cut and paste each cell one over to the right using ”Move right one space” transitions. When you are attempting to paste into the marked cell, paste a marked version of what is on your clipboard, and copy an unmarked version of the previous symbol onto your clipboard. I.e. If you had b in your clipboard and a^* in the cell, you would paste b^* into the cell and copy a to your clipboard.

   4. Perform another ”Move All the Way Left” transition

   5. Perform ”Move right one space” transitions until you reach the marked cell.

   6. Replace the marked symbol with an unmarked version.

   This will result in the head being moved over one space to the left.

2. **(20 points)** Show that the set of three-dimensional coordinates $\{(x, y, z) | x, y, z \in \mathbb{Z}\}$ has size equal to $\mathbb{N}$.

   Answer:

   Let F be a bijective function mapping $\mathbb{Z} \to \mathbb{N}$. One such function is the following. If $i >= 0$, then $F(i) = 2i$. If $i < 0$, then $F(i) = 2|i| - 1$.

   Let P be a bijective function mapping $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. An example of such a function can be derived from the following logic. There exists a unique prime factorization for all natural numbers. Also consider the fact that all prime numbers except for 2 are odd, and that the product of odd numbers is always odd. From these facts we can derive the following:

$$n = 2^a(2b - 1) \tag{1}$$

1 Question 1 - Extra Credit **10 / 10**

✓ **- 0 pts** **Correct**

- 5 pts Partially correct answer

- 5 pts Did not explain how “Move Left One Space” transition is implemented with only “Move Right One Space” transitions and “Move All the Way Left” transitions

- 10 pts Incorrect/No answer

Note: This homework set starts with an extra credit problem. The extra credit problem is optional, but is worth extra credit points. Problems 2 and 3 are *not* extra credit problems.

1. **(Optional: 10 Extra Credit Points aka 5 Tokens)** Let us define Fast-Rewind Turing Machines (FRTM). They are similar to Turing Machines, except that the head of a FRTM is not allowed to move left one cell at a time. Instead, the head of the FRTM *must* move left only all the way to the left-hand end of the tape (i.e. the first cell). The head of the FRTM can move right one step at a time, just like the usual Turing Machines. The transition function for the FRTM is of the form $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, FIRST\}$.

   Explain how to convert any Turing Machine into an FRTM.

   A full proof is not needed, you only need to give an explanation of the intuition behind why your transformation works.

   Note that this requires you to show how to implement a "Move Left One Space" transition in a FRTM by using only "Move Right One Space" transitions and "Move All the Way Left" transitions.

   Answer:

   All properties ("move one space right", infinite tape) of an FRTM are identical to a standard Turing Machine, except for the "Move left one space" transition. In order to convert a "Move left one space" transition to a sequence of transitions in an FRTM we do the following:

   1. Mark the current cell

   2. Perform a "Move All the Way Left" transition

   3. Cut and paste each cell one over to the right using "Move right one space" transitions. When you are attempting to paste into the marked cell, paste a marked version of what is on your clipboard, and copy an unmarked version of the previous symbol onto your clipboard. I.e. If you had b in your clipboard and a^* in the cell, you would paste b^* into the cell and copy a to your clipboard.

   4. Perform another "Move All the Way Left" transition

   5. Perform "Move right one space" transitions until you reach the marked cell.

   6. Replace the marked symbol with an unmarked version.

   This will result in the head being moved over one space to the left.

2. **(20 points)** Show that the set of three-dimensional coordinates $\{(x, y, z)|x, y, z \in \mathbb{Z}\}$ has size equal to $\mathbb{N}$.

   Answer:

   Let F be a bijective function mapping $\mathbb{Z} \to \mathbb{N}$. One such function is the following. If $i >= 0$, then $F(i) = 2i$. If $i < 0$, then $F(i) = 2|i| - 1$.

   Let P be a bijective function mapping $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. An example of such a function can be derived from the following logic. There exists a unique prime factorization for all natural numbers. Also consider the fact that all prime numbers except for 2 are odd, and that the product of odd numbers is always odd. From these facts we can derive the following:

$$n = 2^a(2b - 1) \tag{1}$$

All natural numbers can be uniquely expressed as the product of a power of 2 and an odd number. Thus our function P to uniquely map a tuple of natural number to a single natural number is:

$$P(a, b) = 2^{a-1}(2b - 1) \tag{2}$$

We can now construct a bijective function to uniquely map a set of three-dimensional coordinates to a natural number as follows:

$$M(x, y, z) = P(P(F(x), F(y)), F(z)) \tag{3}$$

Since it is possible to construct a bijective mapping of the set of three-dimensional coordinates to the set of natural numbers, these sets must be equal in size.

3. **(50 points).** In class, we showed the existence of two kinds of infinities. Let $\Sigma = \{0, 1\}$ and $\mathcal{L} = \{L \mid L \subseteq \Sigma^*\}$. We showed that $|\Sigma^*|$ is not the same as $|\mathcal{L}|$.

   We briefly describe the proof below and establish some notation. The proof proceeds by contradiction. Let $(\epsilon, 0, 1, 00, 01, 10, 11, \ldots)$ be a given enumeration of strings in Σ^*. We assume for the sake of contradiction that there exists some enumeration $(L_1, L_2, L_3, \ldots)$ of languages in $\mathcal{L}$. Then we proceed by constructing a language $L^{DIAG} \in \mathcal{L}$ such that $\forall i, L^{DIAG} \neq L_i$ via Cantor's Diagonalization to establish a contradiction.

   (a) **(10 points)** Call $L_1^{DIAG} = L^{DIAG}$. Construct another language $L_2^{DIAG} \neq L_1^{DIAG}$ via diagonalization which is also not present in the enumeration $(L_1, L_2, L_3, \ldots)$. Thus L_2^{DIAG} would have also proved to us that $|\Sigma^*| \neq |\mathcal{L}|$.
   Answer:

   $$L_2^{DIAG} = \{w_i | i <= 1, w_i \notin L_1^{DIAG}\} \cup \{w_i | i > 1, w_i \notin L_{i+1}\} \tag{4}$$

   The first set guarantees that $L_2^{DIAG} \neq L_1^{DIAG}$. The second set continues the diagonalization to ensure $L_2^{DIAG} \notin \mathcal{L}$.

   (b) **(15 points)** Construct an infinite set of languages $\mathcal{L}^{DIAG} = \{L_1^{DIAG}, L_2^{DIAG}, \ldots, L_i^{DIAG}, \ldots\}$ via diagonalization by providing a description of L_i^{DIAG} such that

   - $\forall j, j \neq i, L_i^{DIAG} \neq L_j^{DIAG}$.
   - $\forall j, L_i^{DIAG} \neq L_j$.

   Formally prove your construction by induction.
   Answer:

   $$L_1^{DIAG} = L^{DIAG} \tag{5}$$
   $$L_j^{DIAG} = \{w_i | i <= j - 1, w_i \notin L_i^{DIAG}\} \cup \{w_i | i > j - 1, w_i \notin L_{i-j+1}\} \tag{6}$$

   We will prove the correctness of this construction via induction.
   Let us first consider $j = 1$. By the construction of L^{DIAG} through Cantor's diagonalization, we know that $L_1^{DIAG} \notin \mathcal{L}$.
   Now let us consider L_{j+1}^{DIAG}. It is trivial to see that the first set in the L_j^{DIAG} rule ensures that L_{j+1}^{DIAG} is not identical to any of the first j L^{DIAG}, by ensuring at least a single word difference between L_j^{DIAG} and each of the first j L^{DIAG}. The second set follows the rule of Cantor's diagonalization, ensuring $L_{j+1}^{DIAG} \notin \mathcal{L}$.

2 Question 2 **20 / 20**

- ✓ **- 0 pts** **Correct**
- **- 1 pts** typo
- **- 3 pts** unclear detail
- **- 4 pts** minor mistake
- **- 12 pts** significant mistake/so unclear
- **- 8 pts** very unclear/major mistake
- **- 3 pts** Z to N function not bijective: 0 and -1 both map to 1

All natural numbers can be uniquely expressed as the product of a power of 2 and an odd number. Thus our function P to uniquely map a tuple of natural number to a single natural number is:

$$P(a, b) = 2^{a-1}(2b - 1) \tag{2}$$

We can now construct a bijective function to uniquely map a set of three-dimensional coordinates to a natural number as follows:

$$M(x, y, z) = P(P(F(x), F(y)), F(z)) \tag{3}$$

Since it is possible to construct a bijective mapping of the set of three-dimensional coordinates to the set of natural numbers, these sets must be equal in size.

3. **(50 points).** In class, we showed the existence of two kinds of infinities. Let $\Sigma = \{0, 1\}$ and $\mathcal{L} = \{L \mid L \subseteq \Sigma^*\}$. We showed that $|\Sigma^*|$ is not the same as $|\mathcal{L}|$.

   We briefly describe the proof below and establish some notation. The proof proceeds by contradiction. Let $(\epsilon, 0, 1, 00, 01, 10, 11, \ldots)$ be a given enumeration of strings in Σ^*. We assume for the sake of contradiction that there exists some enumeration $(L_1, L_2, L_3, \ldots)$ of languages in $\mathcal{L}$. Then we proceed by constructing a language $L^{DIAG} \in \mathcal{L}$ such that $\forall i, L^{DIAG} \neq L_i$ via Cantor's Diagonalization to establish a contradiction.

   (a) **(10 points)** Call $L_1^{DIAG} = L^{DIAG}$. Construct another language $L_2^{DIAG} \neq L_1^{DIAG}$ via diagonalization which is also not present in the enumeration $(L_1, L_2, L_3, \ldots)$. Thus L_2^{DIAG} would have also proved to us that $|\Sigma^*| \neq |\mathcal{L}|$.
   Answer:

$$L_2^{DIAG} = \{w_i | i <= 1, w_i \notin L_1^{DIAG}\} \cup \{w_i | i > 1, w_i \notin L_{i+1}\} \tag{4}$$

   The first set guarantees that $L_2^{DIAG} \neq L_1^{DIAG}$. The second set continues the diagonalization to ensure $L_2^{DIAG} \notin \mathcal{L}$.

   (b) **(15 points)** Construct an infinite set of languages $\mathcal{L}^{DIAG} = \{L_1^{DIAG}, L_2^{DIAG}, \ldots, L_i^{DIAG}, \ldots\}$ via diagonalization by providing a description of L_i^{DIAG} such that
   - $\forall j, j \neq i, L_i^{DIAG} \neq L_j^{DIAG}$.
   - $\forall j, L_i^{DIAG} \neq L_j$.

   Formally prove your construction by induction.
   Answer:

$$L_1^{DIAG} = L^{DIAG} \tag{5}$$
$$L_j^{DIAG} = \{w_i | i <= j - 1, w_i \notin L_i^{DIAG}\} \cup \{w_i | i > j - 1, w_i \notin L_{i-j+1}\} \tag{6}$$

   We will prove the correctness of this construction via induction.
   Let us first consider $j = 1$. By the construction of L^{DIAG} through Cantor's diagonalization, we know that $L_1^{DIAG} \notin \mathcal{L}$.
   Now let us consider L_{j+1}^{DIAG}. It is trivial to see that the first set in the L_j^{DIAG} rule ensures that L_{j+1}^{DIAG} is not identical to any of the first j L^{DIAG}, by ensuring at least a single word difference between L_j^{DIAG} and each of the first j L^{DIAG}. The second set follows the rule of Cantor's diagonalization, ensuring $L_{j+1}^{DIAG} \notin \mathcal{L}$.

3.1 3a **10 / 10**

- ✓ **- 0 pts** Correct
- **- 1 pts** Small Mistake
- **- 2 pts** Imperfect Proof
- **- 8 pts** "I don't know"
- **- 10 pts** Incorrect

All natural numbers can be uniquely expressed as the product of a power of 2 and an odd number. Thus our function P to uniquely map a tuple of natural number to a single natural number is:

$$P(a,b) = 2^{a-1}(2b-1) \tag{2}$$

We can now construct a bijective function to uniquely map a set of three-dimensional coordinates to a natural number as follows:

$$M(x,y,z) = P(P(F(x),F(y)),F(z)) \tag{3}$$

Since it is possible to construct a bijective mapping of the set of three-dimensional coordinates to the set of natural numbers, these sets must be equal in size.

3. **(50 points).** In class, we showed the existence of two kinds of infinities. Let $\Sigma = \{0,1\}$ and $\mathcal{L} = \{L \mid L \subseteq \Sigma^*\}$. We showed that $|\Sigma^*|$ is not the same as $|\mathcal{L}|$.

We briefly describe the proof below and establish some notation. The proof proceeds by contradiction. Let $(\epsilon, 0, 1, 00, 01, 10, 11, \ldots)$ be a given enumeration of strings in Σ^*. We assume for the sake of contradiction that there exists some enumeration $(L_1, L_2, L_3, \ldots)$ of languages in $\mathcal{L}$. Then we proceed by constructing a language $L^{DIAG} \in \mathcal{L}$ such that $\forall i, L^{DIAG} \neq L_i$ via Cantor's Diagonalization to establish a contradiction.

(a) **(10 points)** Call $L_1^{DIAG} = L^{DIAG}$. Construct another language $L_2^{DIAG} \neq L_1^{DIAG}$ via diagonalization which is also not present in the enumeration $(L_1, L_2, L_3, \ldots)$. Thus L_2^{DIAG} would have also proved to us that $|\Sigma^*| \neq |\mathcal{L}|$.

Answer:

$$L_2^{DIAG} = \{w_i | i <= 1, w_i \notin L_1^{DIAG}\} \cup \{w_i | i > 1, w_i \notin L_{i+1}\} \tag{4}$$

The first set guarantees that $L_2^{DIAG} \neq L_1^{DIAG}$. The second set continues the diagonalization to ensure $L_2^{DIAG} \notin \mathcal{L}$.

(b) **(15 points)** Construct an infinite set of languages $\mathcal{L}^{DIAG} = \{L_1^{DIAG}, L_2^{DIAG}, \ldots, L_i^{DIAG}, \ldots\}$ via diagonalization by providing a description of L_i^{DIAG} such that

- $\forall j, j \neq i, L_i^{DIAG} \neq L_j^{DIAG}$.
- $\forall j, L_i^{DIAG} \neq L_j$.

Formally prove your construction by induction.

Answer:

$$L_1^{DIAG} = L^{DIAG} \tag{5}$$

$$L_j^{DIAG} = \{w_i | i <= j-1, w_i \notin L_i^{DIAG}\} \cup \{w_i | i > j-1, w_i \notin L_{i-j+1}\} \tag{6}$$

We will prove the correctness of this construction via induction.

Let us first consider $j = 1$. By the construction of L^{DIAG} through Cantor's diagonalization, we know that $L_1^{DIAG} \notin \mathcal{L}$.

Now let us consider L_{j+1}^{DIAG}. It is trivial to see that the first set in the L_j^{DIAG} rule ensures that L_{j+1}^{DIAG} is not identical to any of the first j L^{DIAG}, by ensuring at least a single word difference between L_j^{DIAG} and each of the first j L^{DIAG}. The second set follows the rule of Cantor's diagonalization, ensuring $L_{j+1}^{DIAG} \notin \mathcal{L}$.

3.2 3b **15 / 15**

[✓ **- 0 pts** Correct]

- 1 pts Small Mistake

- 3 pts Imperfect Proof

- 12 pts "I don't know"

- 15 pts Incorrect

(c) **(15 points)** Construct one more language $L^{SUPERDIAG}$ such that

- $\forall j, L^{SUPERDIAG} \neq L_j^{DIAG}$ and
- $\forall j, L^{SUPERDIAG} \neq L_j$

Briefly explain why your language satisfies the above properties.

Answer:

$$L^{SUPERDIAG} = \{w_{2i} | w_{2i} \notin L_i^{DIAG}\} \cup \{w_{2i-1} | w_{2i-1} \notin L_i\} \tag{7}$$

Since there are infinite number of L^{DIAG}, we can't use the exact construction we used for L_j^{DIAG}, since that only ensures that the resulting language is different from the first $j-1$ L^{DIAG}. We need a more generic method that doesn't rely on a bounding value. The method we use here is odds and evens. Let w_i be some word in Σ^* with index i in the enumeration. If i is even, we check that the word has not been included in $L_{(i+1)/2}^{DIAG}$. If i is odd, we check that the word has not been included in $L_{(i+1)/2}$. Since this even odd scheme can be continued infinitely, we can guarantee that $L^{SUPERDIAG} \notin \mathcal{L}$ and $L^{SUPERDIAG} \notin \mathcal{L}^{DIAG}$.

(d) **(10 points)** Construct yet another language $L_2^{SUPERDIAG}$ that is different from $L^{SUPERDIAG}$ satisfying the same properties as part (c). Briefly explain your answer.

Answer:

We can utilize an analogous construction to part (a).

$$L_2^{SUPERDIAG} = \{w_i | i <= 1, w_i \notin L^{SUPERDIAG}\} \cup \{w_{2i} | w_{2i} \notin L_i^{DIAG}\} \cup \{w_{2i-1} | w_{2i-1} \notin L_i\} \tag{8}$$

The first set of the construction guarantees that $L_2^{SUPERDIAG} \neq L^{SUPERDIAG}$. The last 2 sets ensure $L_2^{SUPERDIAG} \notin \mathcal{L}$ and $L_2^{SUPERDIAG} \notin \mathcal{L}^{DIAG}$ via the same principle explained in part (c).

3.3 3c **15 / 15**

✓ **- 0 pts** Correct

- 1 pts Small Mistake

- 3 pts Imperfect Proof

- 12 pts "I don't know"

- 15 pts Incorrect

(c) **(15 points)** Construct one more language $L^{SUPERDIAG}$ such that

- $\forall j, L^{SUPERDIAG} \neq L_j^{DIAG}$ and
- $\forall j, L^{SUPERDIAG} \neq L_j$

Briefly explain why your language satisfies the above properties.

Answer:

$$L^{SUPERDIAG} = \{w_{2i} | w_{2i} \notin L_i^{DIAG}\} \cup \{w_{2i-1} | w_{2i-1} \notin L_i\} \tag{7}$$

Since there are infinite number of L^{DIAG}, we can't use the exact construction we used for L_j^{DIAG}, since that only ensures that the resulting language is different from the first $j-1$ L^{DIAG}. We need a more generic method that doesn't rely on a bounding value. The method we use here is odds and evens. Let w_i be some word in Σ^* with index i in the enumeration. If i is even, we check that the word has not been included in $L_{(i+1)/2}^{DIAG}$. If i is odd, we check that the word has not been included in $L_{(i+1)/2}$. Since this even odd scheme can be continued infinitely, we can guarantee that $L^{SUPERDIAG} \notin \mathcal{L}$ and $L^{SUPERDIAG} \notin \mathcal{L}^{DIAG}$.

(d) **(10 points)** Construct yet another language $L_2^{SUPERDIAG}$ that is different from $L^{SUPERDIAG}$ satisfying the same properties as part (c). Briefly explain your answer.

Answer:

We can utilize an analogous construction to part (a).

$$L_2^{SUPERDIAG} = \{w_i | i <= 1, w_i \notin L^{SUPERDIAG}\} \cup \{w_{2i} | w_{2i} \notin L_i^{DIAG}\} \cup \{w_{2i-1} | w_{2i-1} \notin L_i\} \tag{8}$$

The first set of the construction guarantees that $L_2^{SUPERDIAG} \neq L^{SUPERDIAG}$. The last 2 sets ensure $L_2^{SUPERDIAG} \notin \mathcal{L}$ and $L_2^{SUPERDIAG} \notin \mathcal{L}^{DIAG}$ via the same principle explained in part (c).

3.4 3d **9 / 10**

- **- 0 pts** Correct
- ✓ **- 1 pts** **Small Mistake**
- **- 2 pts** Imperfect Proof
- **- 8 pts** "I don't know"
- **- 10 pts** Incorrect