CS/ECE 374 P30

Abhay Varmaraja, Jiawei Tang, Pengxu Zheng

TOTAL POINTS

85 / 100

QUESTION 1

1 30.A. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Blank or says the language is decidable

- 20 pts Says the language is undecidable, but reduction is missing or completely incorrect (for example, reduces in the wrong direction)

- 15 pts Says the language is undecidable; reduction is on the right track but misses some key step

- 10 pts Says the language is undecidable, reduction is correct, but proof is missing or incorrect

- 5 pts Says the language is decidable, reduction is correct, but one direction of the proof is missing or incorrect

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions

QUESTION 2

2 30.B. **10 / 25**

- 0 pts Correct

- 18.75 pts IDK

- 25 pts Blank or says the language is undecidable

- 20 pts Says the language is decidable, but algorithm is missing or completely incorrect

✓ **- 15 pts Says the language is decidable; algorithm is on the right track but misses some key step**

- 10 pts Says the language is decidable, algorithm is correct, but brief justification of algorithm is missing or incorrect.

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions

After you construct the DFS from the NFS, you have to find cycle in the DFS (the strongly connected components) first, because there might be cycles occurring among non-accepting states, leading the result to be incorrectly infinite.

QUESTION 3

3 30.C. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Says the language is undecidable.

- 20 pts Says the language is decidable, but algorithm is missing or completely incorrect.

- 15 pts Says the language is decidable; algorithm is on the right track but misses some key step

- 10 pts Says the language is decidable, algorithm is correct, but brief justification of algorithm is missing or incorrect

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions

QUESTION 4

4 30.D. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Blank or says the language is decidable

- 20 pts Says the language is undecidable, but reduction is missing or completely incorrect (for example, reduces in the wrong direction)

- 15 pts Says the language is undecidable; reduction is on the right track but misses some key step

- 10 pts Says the language is undecidable, reduction is correct, but proof is missing or incorrect

- 5 pts Says the language is undecidable, reduction is correct, but one direction of the proof is missing or incorrect

- **- 1 pts** One typo or minor omission
- **- 3 pts** Two typos or minor omissions

HW10 Solution

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.0**

Submitted by:

- **≪Pengxu Zheng≫**: **≪pzheng5≫**
- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Abhay Varmaraja≫**: **≪abhaymv2≫**

30

Solution:

30.A. (Proof template borrowed from Discussion 12b. solution)

This language is not decidable. For the sake of the contradiction, we assume there exists an algorithm $DecideL1(< M, N >)$ that correctly decides the language L_1. Then we try to solve the halting problem as follows:

DecideHalt(<M, N, w>):
    Encode the following Turing Machine M_1:
        M_1(x):
            run M on any input w
            return TRUE
    Encode the following NFA N_1:
        N_1(x):
            Let N have only one starting state with no accepting states and no transitions
            return TRUE
    if $DecideL1(< M_1, N_1 >)$:
        return TRUE
    return FALSE

With no accepting states, L(N_1) is essentially the empty set.

We prove the above reduction correct as follows:

Suppose M halts on input w:
Then M_1 accepts any input strings.
Thus, L(M_1) = Σ^* = $\overline{L(N_1)}$.
Therefore, $DecideL1$ accepts <M_1, N_1>.
Therefore, $DecideHalt$ correctly accepts <M, w>.

Suppose M doesn't halt on input w:
Then M_1 diverges on any input strings.
Thus, L(M_1) = $\emptyset$ = L(N_1) $\neq \overline{L(N_1)}$.
Therefore, $DecideL1$ rejects <M_1, N_1>.
Therefore, $DecideHalt$ correctly rejects <M, w>.

In both cases, $DecideHalt$ is correctly decided. However, this contradicts with Turing's theory that the halting problem is never decidable. Therefore, we conclude that the language L_1 is not decidable.

30.B.

This language is decidable. We propose the algorithm to decide L_2 as follows:

We use the powerset construction method introduced earlier to convert NFA N to a DFA named G. G can be viewd as a directed graph with its states being vertices and transitions being edges. After that, we apply DFS to conduct cycle detection on G. If a cycle was found, then we can conclude that there exist an infinite number of transitions (a "self-loop" on a state, or some states that point toward their preceding states, for example) in N, which indicates that the language L(N) is infinite. We conclude the language L(N) is finite otherwise.

30.C. (Discussed with Hengzhi Yuan's Group)

This language is decidable. We propose the algorithm to decide L_3 as follows:

We first construct 2 DFAs for the given input R and N and name the new DFAs G_r and G_n, accordingly. We let the newly created DFAs to accept whatever the input R and N accept, respectively. We then creat a new DFA named G to be the product construction of G_r and G_n, which its vertices and edges are the combinations of states and transitions of G_r and G_n, respectively. G now becomes a directed graph. We now apply BFS on the starting state of G to all reachable vertices. For every vertex traversed, we examine if a specific state's composition, say, q_r and q_n, both belong to (or both don't belong to) the set of corresponding accepting states that originate from G_r and G_n, respectively. If that condition is true, then we conclude that L(G_r) = L(G_n), which further implies L(R) = L(N), and vice versa if there exists any combination of q_r and q_n such that either one of the composition fails to belong to the same set of states (in G_r and G_n) with the other.

30.D. (Proof template borrowed from Discussion 12b. solution)

This language is not decidable. For the sake of the contradiction, we assume there exists an algorithm $DecideL4(< M >)$ that correctly decides the language L_4. Then we try to solve the halting problem as follows:

```
DecideHalt(<M, w>):
    Encode the following Turing Machine M1:
        M1(x):
            run M on any input w
            return TRUE
    if DecideL4(< M1 >):
        return TRUE
    return FALSE
```

We prove the above reduction correct as follows:

Suppose M halts on input w:
Then M_1 accepts any input strings.
Thus, L(M_1) = Σ^* that must include some words of even length.
Therefore, $DecideL4$ accepts <M_1>.
Therefore, $DecideHalt$ correctly accepts <M, w>.

1 30.A. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Blank or says the language is decidable

- 20 pts Says the language is undecidable, but reduction is missing or completely incorrect (for example, reduces in the wrong direction)

- 15 pts Says the language is undecidable; reduction is on the right track but misses some key step

- 10 pts Says the language is undecidable, reduction is correct, but proof is missing or incorrect

- 5 pts Says the language is decidable, reduction is correct, but one direction of the proof is missing or incorrect

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions

In both cases, $DecideHalt$ is correctly decided. However, this contradicts with Turing's theory that the halting problem is never decidable. Therefore, we conclude that the language L_1 is not decidable.

30.B.

This language is decidable. We propose the algorithm to decide L_2 as follows:

We use the powerset construction method introduced earlier to convert NFA N to a DFA named G. G can be viewd as a directed graph with its states being vertices and transitions being edges. After that, we apply DFS to conduct cycle detection on G. If a cycle was found, then we can conclude that there exist an infinite number of transitions (a "self-loop" on a state, or some states that point toward their preceding states, for example) in N, which indicates that the language L(N) is infinite. We conclude the language L(N) is finite otherwise.

30.C. (Discussed with Hengzhi Yuan's Group)

This language is decidable. We propose the algorithm to decide L_3 as follows:

We first construct 2 DFAs for the given input R and N and name the new DFAs G_r and G_n, accordingly. We let the newly created DFAs to accept whatever the input R and N accept, respectively. We then creat a new DFA named G to be the product construction of G_r and G_n, which its vertices and edges are the combinations of states and transitions of G_r and G_n, respectively. G now becomes a directed graph. We now apply BFS on the starting state of G to all reachable vertices. For every vertex traversed, we examine if a specific state's composition, say, q_r and q_n, both belong to (or both don't belong to) the set of corresponding accepting states that originate from G_r and G_n, respectively. If that condition is true, then we conclude that L(G_r) = L(G_n), which further implies L(R) = L(N), and vice versa if there exists any combination of q_r and q_n such that either one of the composition fails to belong to the same set of states (in G_r and G_n) with the other.

30.D. (Proof template borrowed from Discussion 12b. solution)

This language is not decidable. For the sake of the contradiction, we assume there exists an algorithm $DecideL4(< M >)$ that correctly decides the language L_4. Then we try to solve the halting problem as follows:

```
DecideHalt(<M, w>):
    Encode the following Turing Machine M1:
        M1(x):
            run M on any input w
            return TRUE
    if DecideL4(< M1 >):
        return TRUE
    return FALSE
```

We prove the above reduction correct as follows:

Suppose M halts on input w:
Then M_1 accepts any input strings.
Thus, L(M_1) = Σ^* that must include some words of even length.
Therefore, $DecideL4$ accepts <M_1>.
Therefore, $DecideHalt$ correctly accepts <M, w>.

2 30.B. **10 / 25**

- **- 0 pts** Correct
- **- 18.75 pts** IDK
- **- 25 pts** Blank or says the language is undecidable
- **- 20 pts** Says the language is decidable, but algorithm is missing or completely incorrect
- ✓ **- 15 pts Says the language is decidable; algorithm is on the right track but misses some key step**
- **- 10 pts** Says the language is decidable, algorithm is correct, but brief justification of algorithm is missing or incorrect.
- **- 1 pts** One typo or minor omission
- **- 3 pts** Two typos or minor omissions

After you construct the DFS from the NFS, you have to find cycle in the DFS (the strongly connected components) first, because there might be cycles occurring among non-accepting states, leading the result to be incorrectly infinite.

In both cases, $DecideHalt$ is correctly decided. However, this contradicts with Turing's theory that the halting problem is never decidable. Therefore, we conclude that the language L_1 is not decidable.

30.B.

This language is decidable. We propose the algorithm to decide L_2 as follows:

We use the powerset construction method introduced earlier to convert NFA N to a DFA named G. G can be viewd as a directed graph with its states being vertices and transitions being edges. After that, we apply DFS to conduct cycle detection on G. If a cycle was found, then we can conclude that there exist an infinite number of transitions (a "self-loop" on a state, or some states that point toward their preceding states, for example) in N, which indicates that the language L(N) is infinite. We conclude the language L(N) is finite otherwise.

30.C. (Discussed with Hengzhi Yuan's Group)

This language is decidable. We propose the algorithm to decide L_3 as follows:

We first construct 2 DFAs for the given input R and N and name the new DFAs G_r and G_n, accordingly. We let the newly created DFAs to accept whatever the input R and N accept, respectively. We then creat a new DFA named G to be the product construction of G_r and G_n, which its vertices and edges are the combinations of states and transitions of G_r and G_n, respectively. G now becomes a directed graph. We now apply BFS on the starting state of G to all reachable vertices. For every vertex traversed, we examine if a specific state's composition, say, q_r and q_n, both belong to (or both don't belong to) the set of corresponding accepting states that originate from G_r and G_n, respectively. If that condition is true, then we conclude that L(G_r) = L(G_n), which further implies L(R) = L(N), and vice versa if there exists any combination of q_r and q_n such that either one of the composition fails to belong to the same set of states (in G_r and G_n) with the other.

30.D. (Proof template borrowed from Discussion 12b. solution)

This language is not decidable. For the sake of the contradiction, we assume there exists an algorithm $DecideL4(< M >)$ that correctly decides the language L_4. Then we try to solve the halting problem as follows:

```
DecideHalt(<M, w>):
    Encode the following Turing Machine M1:
        M1(x):
            run M on any input w
            return TRUE
    if DecideL4(< M1 >):
        return TRUE
    return FALSE
```

We prove the above reduction correct as follows:

Suppose M halts on input w:
Then M_1 accepts any input strings.
Thus, L(M_1) = Σ^* that must include some words of even length.
Therefore, $DecideL4$ accepts <M_1>.
Therefore, $DecideHalt$ correctly accepts <M, w>.

3 30.C. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Says the language is undecidable.

- 20 pts Says the language is decidable, but algorithm is missing or completely incorrect.

- 15 pts Says the language is decidable; algorithm is on the right track but misses some key step

- 10 pts Says the language is decidable, algorithm is correct, but brief justification of algorithm is missing or incorrect

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions

In both cases, $DecideHalt$ is correctly decided. However, this contradicts with Turing's theory that the halting problem is never decidable. Therefore, we conclude that the language L_1 is not decidable.

30.B.

This language is decidable. We propose the algorithm to decide L_2 as follows:

We use the powerset construction method introduced earlier to convert NFA N to a DFA named G. G can be viewd as a directed graph with its states being vertices and transitions being edges. After that, we apply DFS to conduct cycle detection on G. If a cycle was found, then we can conclude that there exist an infinite number of transitions (a "self-loop" on a state, or some states that point toward their preceding states, for example) in N, which indicates that the language L(N) is infinite. We conclude the language L(N) is finite otherwise.

30.C. (Discussed with Hengzhi Yuan's Group)

This language is decidable. We propose the algorithm to decide L_3 as follows:

We first construct 2 DFAs for the given input R and N and name the new DFAs G_r and G_n, accordingly. We let the newly created DFAs to accept whatever the input R and N accept, respectively. We then creat a new DFA named G to be the product construction of G_r and G_n, which its vertices and edges are the combinations of states and transitions of G_r and G_n, respectively. G now becomes a directed graph. We now apply BFS on the starting state of G to all reachable vertices. For every vertex traversed, we examine if a specific state's composition, say, q_r and q_n, both belong to (or both don't belong to) the set of corresponding accepting states that originate from G_r and G_n, respectively. If that condition is true, then we conclude that L(G_r) = L(G_n), which further implies L(R) = L(N), and vice versa if there exists any combination of q_r and q_n such that either one of the composition fails to belong to the same set of states (in G_r and G_n) with the other.

30.D. (Proof template borrowed from Discussion 12b. solution)

This language is not decidable. For the sake of the contradiction, we assume there exists an algorithm $DecideL4(< M >)$ that correctly decides the language L_4. Then we try to solve the halting problem as follows:

```
DecideHalt(<M, w>):
    Encode the following Turing Machine M1:
        M1(x):
            run M on any input w
            return TRUE
    if DecideL4(< M1 >):
        return TRUE
    return FALSE
```

We prove the above reduction correct as follows:

Suppose M halts on input w:
Then M_1 accepts any input strings.
Thus, L(M_1) = Σ^* that must include some words of even length.
Therefore, $DecideL4$ accepts <M_1>.
Therefore, $DecideHalt$ correctly accepts <M, w>.

Suppose M doesn't halt on input w:
Then M_1 diverges on any input strings.
Thus, L(M_1) $= \emptyset$, which further implies that L(M_1) must not include any words of even length.
Therefore, $DecideL4$ rejects $<M_1>$.
Therefore, $DecideHalt$ correctly rejects $<$M, w$>$.

In both cases, $DecideHalt$ is correctly decided. However, this contradicts with Turing's theory that the halting problem is never decidable. Therefore, we conclude that the language L_4 is not decidable.

4 30.D. **25 / 25**

✓ **- 0 pts** Correct

- 18.75 pts IDK

- 25 pts Blank or says the language is decidable

- 20 pts Says the language is undecidable, but reduction is missing or completely incorrect (for example, reduces in the wrong direction)

- 15 pts Says the language is undecidable; reduction is on the right track but misses some key step

- 10 pts Says the language is undecidable, reduction is correct, but proof is missing or incorrect

- 5 pts Says the language is undecidable, reduction is correct, but one direction of the proof is missing or incorrect

- 1 pts One typo or minor omission

- 3 pts Two typos or minor omissions