1 3F-1 Bookkeeping

- **0 pts** Correct

2 Exercise 3F-2. Regular Expression, Large-Step [10 points].

$$\frac{\vdash e_1 \ matches \ s_1 \ leaving \ s_2 \quad \vdash e_2 \ matches \ s_2 \ leaving \ s_3}{\vdash e_1 e_2 \ matches \ s_1 \ leaving \ s_3}$$

$$\frac{\vdash e_1 \ matches \ s_1 \ leaving \ s_2}{\vdash e_1 | e_2 \ matches \ s_1 \ leaving \ s_2}$$

$$\frac{\vdash e_2 \ matches \ s_1 \ leaving \ s_2}{\vdash e_1 | e_2 \ matches \ s_1 \ leaving \ s_2}$$

$$\frac{}{\vdash e* \ matches \ s_1 \ leaving \ s_1}$$

$$\frac{\vdash e \ matches \ s_1 \ leaving \ s_2 \quad \vdash e* \ matches \ s_2 \ leaving \ s_3}{\vdash e* \ matches \ s_1 \ leaving \ s_3}$$

3 Exercise 3F-3. Regular Expression and Sets [5 points].

I will claim that operational semantics rules of inference for e and e_1e_2 cannot be done correctly in the given framework.

For e_1e_2, I attempted the following semantics rule. It turns out that matching e_1 will always leave a set of string $\bigcup_i S_{1\_i}$ for e_2 to match. The size of such set is finite but not fixed. Therefore, the subsequent e_2 matching can not be expressed clearly. In the given framework, this semantics rule is incomplete.

$$\frac{\vdash e_1 \ matches \ s \ leaving \ \bigcup_i S_{1\_i} \quad \vdash e_2 \ matches \ S_{1\_1} \ leaving \ S_{2\_1} \quad \vdash e_2 \ matches \ S_{1\_2} \ leaving \ S_{2\_2} \ ....}{\vdash e_1 e_2 \ matches \ s \ leaving \ \bigcup_j S_{2\_j}}$$

For e*, I attempted the following semantics rule. Here, I tried to consider all the combination of e* starting from empty, e, ee, eee all the way to the concatenation of infinite number of e. The number of hypotheses is therefore infinite. My semantics rule is incomplete.

$$\frac{\vdash e \ matches \ s \ leaving \ S_1 \quad \vdash ee \ matches \ s \ leaving \ S_2 \quad \vdash eee \ matches \ s \ leaving \ S_3 \ ....}{\vdash e* \ matches \ s \ leaving \ \bigcup_i S_i}$$

Additionally, if we try to reuse the semantics for e* in 3F-2, we will once again run into the problem of having unfixed number of S_{1_i} in the process of matching e. Additionally, this circular definition can lead to an infinite loop on expressions such as r = empty*. Therefore, the semantics is unsound.

In general, operational semantics rules of inference for e and e_1e_2 can not be done in the given framework.

2 3F-2 Regular Expressions, Large Step

- **0 pts** Correct

2 Exercise 3F-2. Regular Expression, Large-Step [10 points].

$$\frac{\vdash e_1 \ matches \ s_1 \ leaving \ s_2 \quad \vdash e_2 \ matches \ s_2 \ leaving \ s_3}{\vdash e_1 e_2 \ matches \ s_1 \ leaving \ s_3}$$

$$\frac{\vdash e_1 \ matches \ s_1 \ leaving \ s_2}{\vdash e_1 | e_2 \ matches \ s_1 \ leaving \ s_2}$$

$$\frac{\vdash e_2 \ matches \ s_1 \ leaving \ s_2}{\vdash e_1 | e_2 \ matches \ s_1 \ leaving \ s_2}$$

$$\frac{}{\vdash e* \ matches \ s_1 \ leaving \ s_1}$$

$$\frac{\vdash e \ matches \ s_1 \ leaving \ s_2 \quad \vdash e* \ matches \ s_2 \ leaving \ s_3}{\vdash e* \ matches \ s_1 \ leaving \ s_3}$$

3 Exercise 3F-3. Regular Expression and Sets [5 points].

I will claim that operational semantics rules of inference for e and e_1e_2 cannot be done correctly in the given framework.

For e_1e_2, I attempted the following semantics rule. It turns out that matching e_1 will always leave a set of string $\bigcup_i S_{1\_i}$ for e_2 to match. The size of such set is finite but not fixed. Therefore, the subsequent e_2 matching can not be expressed clearly. In the given framework, this semantics rule is incomplete.

$$\frac{\vdash e_1 \ matches \ s \ leaving \ \bigcup_i S_{1\_i} \quad \vdash e_2 \ matches \ S_{1\_1} \ leaving \ S_{2\_1} \quad \vdash e_2 \ matches \ S_{1\_2} \ leaving \ S_{2\_2} \ ....}{\vdash e_1 e_2 \ matches \ s \ leaving \ \bigcup_j S_{2\_j}}$$

For e*, I attempted the following semantics rule. Here, I tried to consider all the combination of e* starting from empty, e, ee, eee all the way to the concatenation of infinite number of e. The number of hypotheses is therefore infinite. My semantics rule is incomplete.

$$\frac{\vdash e \ matches \ s \ leaving \ S_1 \quad \vdash ee \ matches \ s \ leaving \ S_2 \quad \vdash eee \ matches \ s \ leaving \ S_3 \ ....}{\vdash e* \ matches \ s \ leaving \ \bigcup_i S_i}$$

Additionally, if we try to reuse the semantics for e* in 3F-2, we will once again run into the problem of having unfixed number of S_{1_i} in the process of matching e. Additionally, this circular definition can lead to an infinite loop on expressions such as r = empty*. Therefore, the semantics is unsound.

In general, operational semantics rules of inference for e and e_1e_2 can not be done in the given framework.

3 3F-3 Regular Expressions and Sets

- **0 pts** Correct

4 Exercise 3F-4. Equivalence [7 points].

I will claim that the equivalence relation for regular expressions can be computed. The general idea is to find the corresponding finite state machines for e_1 and e_2. Check the equivalence for finite state machines.

Since the reader is familiar with the relevant literature. I will skip the concept of DFA (Deterministic Finite Automata). In general, one can always convert a regular expression to DFA as shown below.

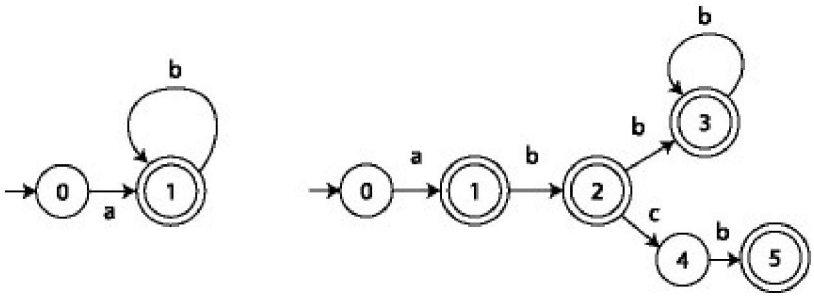

Figure 1: DFA for ab* and a(b*|bcb)

New that we can find the DFA M_1 and M_2 for regular expression e_1 and e_2, we can check equivalence by running BFS on M_1 and M_2. After compressing the paths of M_1 and M_2 using Hopcroft/Ullman algorithm, we perform BFS on M_1 and M_2 in parallel. Every time we first hit a node in the BFS, we record the correspondence for such node between M_1 and M_2. When we hit a node for the second time, we can check equivalence by confirming if its correspondence established before is still preserved. Therefore, this process is decidable.

Additionally, proving DFA equivalence is decidable can be considered as a graph isomorphism problem, which is decidable.

5 Exercise 3F-5. SAT Solving [6 points].

The last two tests take a comparatively long time since we are performing the backtracking based search in our implementation. Since the last two test cases have a relatively large search space, the time to finish last two test cases can be long.

In particular, for Arithmetic model, the current implementation will perform exhaustive bounded search in the function **bounded_search**. Since **bounded_search** is called recursively, exhaustive bounded search between lower bound to upper bound is performed recursively. As a result, the time complexity can go up to exponential. As the search space expands, the running time can be extremely long. To improve the current implementation, we can simply replace this search algorithm with binary search. The time complexity will then be reduced to polynomial, which is more reasonable.

4 3F-4 Equivalence

- **0 pts** Correct

4 Exercise 3F-4. Equivalence [7 points].

I will claim that the equivalence relation for regular expressions can be computed. The general idea is to find the corresponding finite state machines for e_1 and e_2. Check the equivalence for finite state machines.

Since the reader is familiar with the relevant literature. I will skip the concept of DFA (Deterministic Finite Automata). In general, one can always convert a regular expression to DFA as shown below.

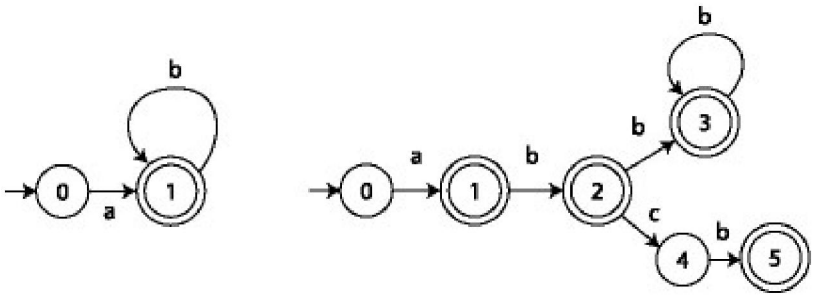

Figure 1: DFA for ab* and a(b*|bcb)

New that we can find the DFA M_1 and M_2 for regular expression e_1 and e_2, we can check equivalence by running BFS on M_1 and M_2. After compressing the paths of M_1 and M_2 using Hopcroft/Ullman algorithm, we perform BFS on M_1 and M_2 in parallel. Every time we first hit a node in the BFS, we record the correspondence for such node between M_1 and M_2. When we hit a node for the second time, we can check equivalence by confirming if its correspondence established before is still preserved. Therefore, this process is decidable.

Additionally, proving DFA equivalence is decidable can be considered as a graph isomorphism problem, which is decidable.

5 Exercise 3F-5. SAT Solving [6 points].

The last two tests take a comparatively long time since we are performing the backtracking based search in our implementation. Since the last two test cases have a relatively large search space, the time to finish last two test cases can be long.

In particular, for Arithmetic model, the current implementation will perform exhaustive bounded search in the function **bounded_search**. Since **bounded_search** is called recursively, exhaustive bounded search between lower bound to upper bound is performed recursively. As a result, the time complexity can go up to exponential. As the search space expands, the running time can be extremely long. To improve the current implementation, we can simply replace this search algorithm with binary search. The time complexity will then be reduced to polynomial, which is more reasonable.

5 3F-5 SAT Solving

- **0 pts** Correct