1 3F-1 Bookkeeping

- 0 pts Correct

Peer Review ID: 68548129 — enter this when you fill out your peer evaluation via gradescope

Exercise 3F-2:

We denote $(s's'')$ as string concatenation of strings s' and s''.

①

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } s' \qquad \vdash e_2 \text{ matches } s \text{ leaving } s''}{\vdash e_1 e_2 \text{ matches } s \text{ leaving } (s's'')}$$

②

a).

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } s'}{\vdash e_1 | e_2 \text{ matches } s \text{ leaving } s'}$$

b).

$$\frac{\vdash e_2 \text{ matches } s \text{ leaving } s''}{\vdash e_1 | e_2 \text{ matches } s \text{ leaving } s''}$$

③

a).

$$\frac{\vdash e \text{ matches } s \text{ leaving } s' \qquad \vdash e^{*} \text{ matches } s \text{ leaving } s''}{\vdash e^{*} \text{ matches } s \text{ leaving } (s's'')}$$

b).

$$\frac{}{\vdash e^{*} \text{ matches empsty leaving empty}}$$

Exercise 3F-3:

2 3F-2 Regular Expressions, Large Step

- **0 pts** Correct

Peer Review ID: 68548129 — enter this when you fill out your peer evaluation via gradescope

I argue that it cannot be done correctly.

①. For $e_1 e_2$, my attempt is:

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } S \qquad \vdash e_2 \text{ matches } s \text{ leaving } S'}{\vdash e_1 e_2 \text{ matches } s \text{ leaving } S \cap S'}$$

This attempt is incorrect because it does not return a consecutive $e_1 e_2$ matching of s. Rather, it only returns the intersection of the results.

In order to apply the consecutive ordering, we will have to embed a derivation inside a set constructor. as shown in the following attempt:

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } S \qquad \vdash e_2 \text{ matches } S \text{ leaving } S' = \{ s' \mid x \in S \text{ s.t. } x \text{ matches } s \text{ leaving } S' \}}{\vdash e_1 e_2 \text{ matches } s \text{ leaving } S'}$$

Since we cannot embed derivation in a set, it is not possible for $e_1 e_2$ case.

②. For e^*, my attempt is:

$$\frac{}{\vdash e^* \text{ matches empty leaving } \emptyset}$$

$$\frac{\vdash e \text{ matches } s \text{ leaving } S \qquad \vdash e^* \text{ matches } S \text{ leaving } S' = \{ s' \mid x \in S \text{ s.t. } e^* \text{matches } s \text{ leaving } S' \}}{\vdash e^* \text{ matches } s \text{ leaving } s'}$$

But we may not use embedded derivation in sets. Thus, e^* case is also impossible.

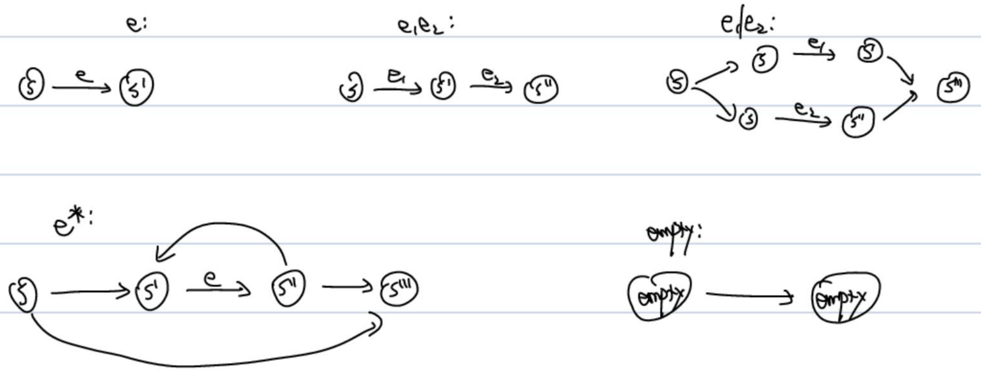
Exercise 3F-4:

3 3F-3 Regular Expressions and Sets

- **0 pts** Correct

Peer Review ID: 68548129 — enter this when you fill out your peer evaluation via gradescope

4. To show that for regular expressions, $e_1 \sim e_2$ is decidable, we transform both e_1 and e_2 to finite state machines. Therefore, the equivalence of e_1 and e_2 can be shown by that of the finite state machines, respectively. We show such transformations below:



These are examples of NFA, which together can make ups more complex NFAs. Moreover, DFAs can be constructed from such NFAs. Therefore, we have that if two regular expressions are equivalent, then their corresponding DFAs will traverse through the same states when given the same input. Since it is proven that DFA equivalence is decidable, we have shown that $e_1 \sim e_2$ is decidable. □

Exercise 3F-5:

The last two included tests take a comparatively longer time because in our implementation, when it comes to Arithmetic models, we are currently performing an exhaustive bounded search to find a possible solution. Furthermore, on each recursive call of the function bounded_search, we would start a new exhaustive search from lower bound all the way up to upper bound. Thus, the algorithm needs to explore a much greater search space for the last two test cases. Therefore, the time it takes to finish the bounded_search to find an answer takes a while when it has to repeatedly search for answers exhaustively.

To improve the module, instead of using an exhaustive bounded search, a binary search makes more sense in this case to find the result we need. Overall, I would improve the Arith module so that its runtime complexity would improve from exponential to maybe polynomial, since this module is heavily used.

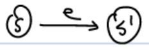
4 3F-4 Equivalence

- 0 pts Correct

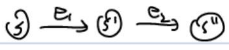
Peer Review ID: 68548129 — enter this when you fill out your peer evaluation via gradescope

4. To show that for regular expressions, $e_1 \sim e_2$ is decidable, we transform both e_1 and e_2 to finite state machines. Therefore, the equivalence of e_1 and e_2 can be shown by that of the finite state machines, respectively. We show such transformations below:

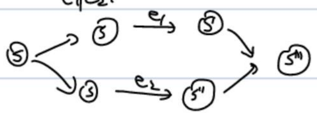
e:



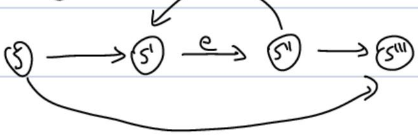
e_1e_2:



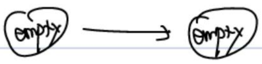
$e_1|e_2$:



e^*:



empty:



These are examples of NFA, which together can make ups more complex NFAs. Moreover, DFAs can be constructed from such NFAs. Therefore, we have that if two regular expressions are equivalent, then their corresponding DFAs will traverse through the same states when given the same input. Since it is proven that DFA equivalence is decidable, we have shown that $e_1 \sim e_2$ is decidable. □

Exercise 3F-5:

The last two included tests take a comparatively longer time because in our implementation, when it comes to Arithmetic models, we are currently performing an exhaustive bounded search to find a possible solution. Furthermore, on each recursive call of the function bounded_search, we would start a new exhaustive search from lower bound all the way up to upper bound. Thus, the algorithm needs to explore a much greater search space for the last two test cases. Therefore, the time it takes to finish the bounded_search to find an answer takes a while when it has to repeatedly search for answers exhaustively.

To improve the module, instead of using an exhaustive bounded search, a binary search makes more sense in this case to find the result we need. Overall, I would improve the Arith module so that its runtime complexity would improve from exponential to maybe polynomial, since this module is heavily used.

5 3F-5 SAT Solving

- **0 pts** Correct