1 3F-1 Bookkeeping

- **0 pts** Correct

3F-2

Concatenation:

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

Or:

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

Kleene star:

$$\frac{\vdash e \text{ matches } s \text{ leaving } s' \quad \vdash e* \text{ matches } s' \text{ leaving } s''}{\vdash e* \text{ matches } s \text{ leaving } s''}$$

$$\frac{}{\vdash e* \text{ matches } s \text{ leaving } s}$$

2 3F-2 Regular Expressions, Large Step

- **0 pts** Correct

Peer Review ID: 68533798 — enter this when you fill out your peer evaluation via gradescope

3F-3

I argue that it is not possible to give operational semantics rules of inference for evaluating regexes deterministically. Consider the following attempts at opsem rules for

$e*$:

(1) $$\frac{}{\dashv e*\ \text{matches}\ s\ \text{leaving}\ \{s\}}$$

(2) $$\frac{\dashv e*\ \text{matches}\ s\ \text{leaving}\ S = \{s' \mid s = e :: s'\} \quad \dashv e*\ \text{matches}\ s'\ \text{leaving}\ S'}{\dashv e*\ \text{matches}\ s\ \text{leaving}\ S \cup S'}$$

e_1e_2:

(3) $$\frac{\dashv e_1\ \text{matches}\ s\ \text{leaving}\ S = \{s' \mid s = e_1 :: s'\} \quad \dashv e_2\ \text{matches}\ s'\ \text{leaving}\ S' = \{s'' \mid s'' = e_2 :: s'\}}{\dashv e_1e_2\ \text{matches}\ s\ \text{leaving}\ S \cup S'}$$

On first glance, these rules look correct. Closer inspection of rules (2) and (3) reveal that the second hypotheses of the rules are dependent on the first hypotheses. The second hypotheses contain the symbol s', which represent the elements in the set produced by the first hypotheses. The second judgment must be evaluated for all the elements in the set produced by the first judgement (S). Since the size of the set S is not known statically, it is not possible to write down the rules with a fixed number of hypotheses. Thus, regexes that produce sets as answers cannot be evaluated using operational semantics with a fixed number of hypotheses.

3 3F-3 Regular Expressions and Sets

- 0 pts Correct

3F-4

I claim that regex equivalence is decidable. To compute it, we use the following algorithm:

1. Convert each regex into a Deterministic Finite Automaton. (I know this is possible because this link says so.)

2. Compare the DFAs of each regex using graph-isomorphism.
    2.1 If the graphs are isomorphic, return that the regexes are equivalent.
    2.2 Return not equivalent otherwise

Since conversion of regex to DFA terminates and graph isomorphism check also terminates, we can be sure that regex equivalence checking also terminates.

Thus, the equivalence of two regexes is decidable.

4 3F-4 Equivalence

- 0 pts Correct

3F-5

Test 35 and 36 are the only ones that have dependent arithmetic variables. The more constraining conditions such as equality are present later in the expression than less constraining conditions such as inequalities. This means that the arithmetic module produces many solutions for the first few inequalities, but each one is rejected because of a more constraining equality condition that is evaluated later. This causes a lot of waste of computation.

Since tests 35 and 36 are the only ones with dependent variables, this phenomenon only occurs in those tests.

I would improve the arith.ml module. The current arithmetic module is very inefficient. It basically searches the entire search space of 256^n (n=number of theory symbols) by brute force to find a solution. I would include the following heuristics to improve the performance:

1. Inspect the constraints to identify sets of dependent variables and then solve each set separately.
2. Within each set, solve for the most constrained variable first and then solve for variables with less constraints. This reduces the chance of an assignment made to a less constrained variable from further constraining a more constrained variable.
3. Inspect the constraints to narrow the search space. For example, if one of the constraints is x>0, don't check values <=0.
4. The search space of each variable can also be constrained by transitively applying constraints of variables the that current variable is dependent on.

5 3F-5 SAT Solving

- 0 pts Correct

Peer Review ID: 68533798 — enter this when you fill out your peer evaluation via gradescope