1 3F-1 Bookkeeping

- **0 pts** Correct

3F-3)

concatenation:

$$\frac{\vdash e1 \text{ matches } s \text{ leaving } r \qquad \vdash e2 \text{ matches } r \text{ leaving } t}{\vdash e1\ e2 \text{ matches } s \text{ leaving } t}$$

or:

$$\frac{s = e1 :: r}{\vdash e1 \mid e2 \text{ matches } s \text{ leaving } r} \qquad \frac{s = e2 :: r}{\vdash e1 \mid e2 \text{ matches } s \text{ leaving } r}$$

kleene star:

$$\frac{}{\vdash e1\ * \text{ matches } s \text{ leaving } r} \qquad \frac{s = e1 :: r \qquad \vdash e1\ * \text{ matches } r \text{ leaving } t}{\vdash e1\ * \text{ matches } s \text{ leaving } t}$$

3F-3)

2 3F-2 Regular Expressions, Large Step

- **0 pts** Correct

3F-3)

concatenation:

$$\frac{\vdash e1 \text{ matches } s \text{ leaving } r \qquad \vdash e2 \text{ matches } r \text{ leaving } t}{\vdash e1\ e2 \text{ matches } s \text{ leaving } t}$$

or:

$$\frac{s = e1 :: r}{\vdash e1 \mid e2 \text{ matches } s \text{ leaving } r} \qquad \frac{s = e2 :: r}{\vdash e1 \mid e2 \text{ matches } s \text{ leaving } r}$$

kleene star:

$$\frac{}{\vdash e1\ * \text{ matches } s \text{ leaving } r} \qquad \frac{s = e1 :: r \qquad \vdash e1\ * \text{ matches } r \text{ leaving } t}{\vdash e1\ * \text{ matches } s \text{ leaving } t}$$

3F-3)

With the given restrictions and framework this is not possible. This is because if we were to have rules for these judgements, as in the question above, there would be two rules for the Kleene star. The second rule would depend on the output of the first rule. However we don't know the size of the set and furthermore we cannot know which specific s is being referenced, and so this rule with the current framework cannot work.

kleene star wrong rule:

$$\frac{}{\vdash e1 \ * \ \textit{matches s leaving } \{s' | s = s'\}}$$

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e1 \ * \ \textit{matches s' leaving } S'}{\vdash e1 \ * \ \textit{matches s leaving } S \cup S'}$$

concatenation wrong rule:

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e2 \textit{ matches s leaving } S \ = \ \{s'' | s'' = e2 :: s'\}}{\vdash e1 \ e2 \ \textit{ matches s leaving } S \cup S'}$$

As shown in the rules above, we can't know for sure which s' is the one of importance, and if we could use constructors we could use an existential quantifier but without this, there is no way to know which s' we are talking about. The second judgement depends on the first judgement, for which we would have to evaluate this on all elements in the set but since the sets are of non-deterministic size, we cannot have a fixed number of rules and hence cannot guarantee that we will use the correct s'. Therefore, this framework cannot support these rules.

3F-4)

I believe that this is decidable. To compute this we will take advantage of the fact that there are known ways to convert regular expressions into DFAs and that checking if two graphs are isomorphic is also decidable. We first convert the two regular expressions into DFAs, next we call our graph isomorphism algorithm to check if the two DFAs are equivalent. We know that both of these algorithms will terminate and are decidable and so the whole thing is decidable; the regular expressions are equivalent if the resulting DFAs are isomorphic.

3F-5)

The last two test cases are slow because the variables in the inequalities and arithmetic statements depend on each other. In the other test cases we force conditions where for example, x < 5 and y > 3. In these cases the variables are independent. In the last two cases however, the variables in the conditions depend on each other. We are given clauses such as, x

3 3F-3 Regular Expressions and Sets

- **0 pts** Correct

With the given restrictions and framework this is not possible. This is because if we were to have rules for these judgements, as in the question above, there would be two rules for the Kleene star. The second rule would depend on the output of the first rule. However we don't know the size of the set and furthermore we cannot know which specific s is being referenced, and so this rule with the current framework cannot work.

kleene star wrong rule:

$$\frac{}{\vdash e1 \ * \ \textit{matches s leaving } \{s' | s = s'\}}$$

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e1 \ * \ \textit{matches s' leaving } S'}{\vdash e1 \ * \ \textit{matches s leaving } S \cup S'}$$

concatenation wrong rule:

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e2 \textit{ matches s leaving } S \ = \ \{s'' | s'' = e2 :: s'\}}{\vdash e1 \ e2 \ \textit{ matches s leaving } S \cup S'}$$

As shown in the rules above, we can't know for sure which s' is the one of importance, and if we could use constructors we could use an existential quantifier but without this, there is no way to know which s' we are talking about. The second judgement depends on the first judgement, for which we would have to evaluate this on all elements in the set but since the sets are of non-deterministic size, we cannot have a fixed number of rules and hence cannot guarantee that we will use the correct s'. Therefore, this framework cannot support these rules.

3F-4)

I believe that this is decidable. To compute this we will take advantage of the fact that there are known ways to convert regular expressions into DFAs and that checking if two graphs are isomorphic is also decidable. We first convert the two regular expressions into DFAs, next we call our graph isomorphism algorithm to check if the two DFAs are equivalent. We know that both of these algorithms will terminate and are decidable and so the whole thing is decidable; the regular expressions are equivalent if the resulting DFAs are isomorphic.

3F-5)

The last two test cases are slow because the variables in the inequalities and arithmetic statements depend on each other. In the other test cases we force conditions where for example, x < 5 and y > 3. In these cases the variables are independent. In the last two cases however, the variables in the conditions depend on each other. We are given clauses such as, x

4 3F-4 Equivalence

- **0 pts** Correct

With the given restrictions and framework this is not possible. This is because if we were to have rules for these judgements, as in the question above, there would be two rules for the Kleene star. The second rule would depend on the output of the first rule. However we don't know the size of the set and furthermore we cannot know which specific s is being referenced, and so this rule with the current framework cannot work.

kleene star wrong rule:

$$\frac{}{\vdash e1 \ * \ \textit{matches s leaving } \{s' | s = s'\}}$$

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e1 \ * \ \textit{matches s' leaving } S'}{\vdash e1 \ * \ \textit{matches s leaving } S \cup S'}$$

concatenation wrong rule:

$$\frac{\vdash e1 \textit{ matches s leaving } S \ = \ \{s' | s = e1 :: s'\} \qquad \vdash e2 \textit{ matches s leaving } S \ = \ \{s'' | s'' = e2 :: s'\}}{\vdash e1 \ e2 \ \textit{ matches s leaving } S \cup S'}$$

As shown in the rules above, we can't know for sure which s' is the one of importance, and if we could use constructors we could use an existential quantifier but without this, there is no way to know which s' we are talking about. The second judgement depends on the first judgement, for which we would have to evaluate this on all elements in the set but since the sets are of non-deterministic size, we cannot have a fixed number of rules and hence cannot guarantee that we will use the correct s'. Therefore, this framework cannot support these rules.

3F-4)

I believe that this is decidable. To compute this we will take advantage of the fact that there are known ways to convert regular expressions into DFAs and that checking if two graphs are isomorphic is also decidable. We first convert the two regular expressions into DFAs, next we call our graph isomorphism algorithm to check if the two DFAs are equivalent. We know that both of these algorithms will terminate and are decidable and so the whole thing is decidable; the regular expressions are equivalent if the resulting DFAs are isomorphic.

3F-5)

The last two test cases are slow because the variables in the inequalities and arithmetic statements depend on each other. In the other test cases we force conditions where for example, x < 5 and y > 3. In these cases the variables are independent. In the last two cases however, the variables in the conditions depend on each other. We are given clauses such as, x

< y and y < z. Because the variables depend on each other, we can't eliminate any of them. Without any elimination, there are no shortcuts the algorithm can take, and so it has to exhaust a lot more possibilities when the variables are dependent.

The module where I believe we can improve this is the Arithmetic Solver. As it is, it is very inefficient because the solver checks all possible values for the variables in a bounded range. To improve upon this, we could update the range of possible values using past information and using the conditions to narrow the range and hence reduce the total possible values which would reduce the time to find an answer. Another way to improve the efficiency is to try and satisfy the variables with the most constraints first. This way we solve the "more difficult" problem first and then it is more likely that satisfying the next variables will have less problems. Solving the variables with less constraints first and then realizing that it is not valid because of the highly constrained variable will waste a lot of time compared to doing it the other way around.

5 3F-5 SAT Solving

- **0 pts** Correct