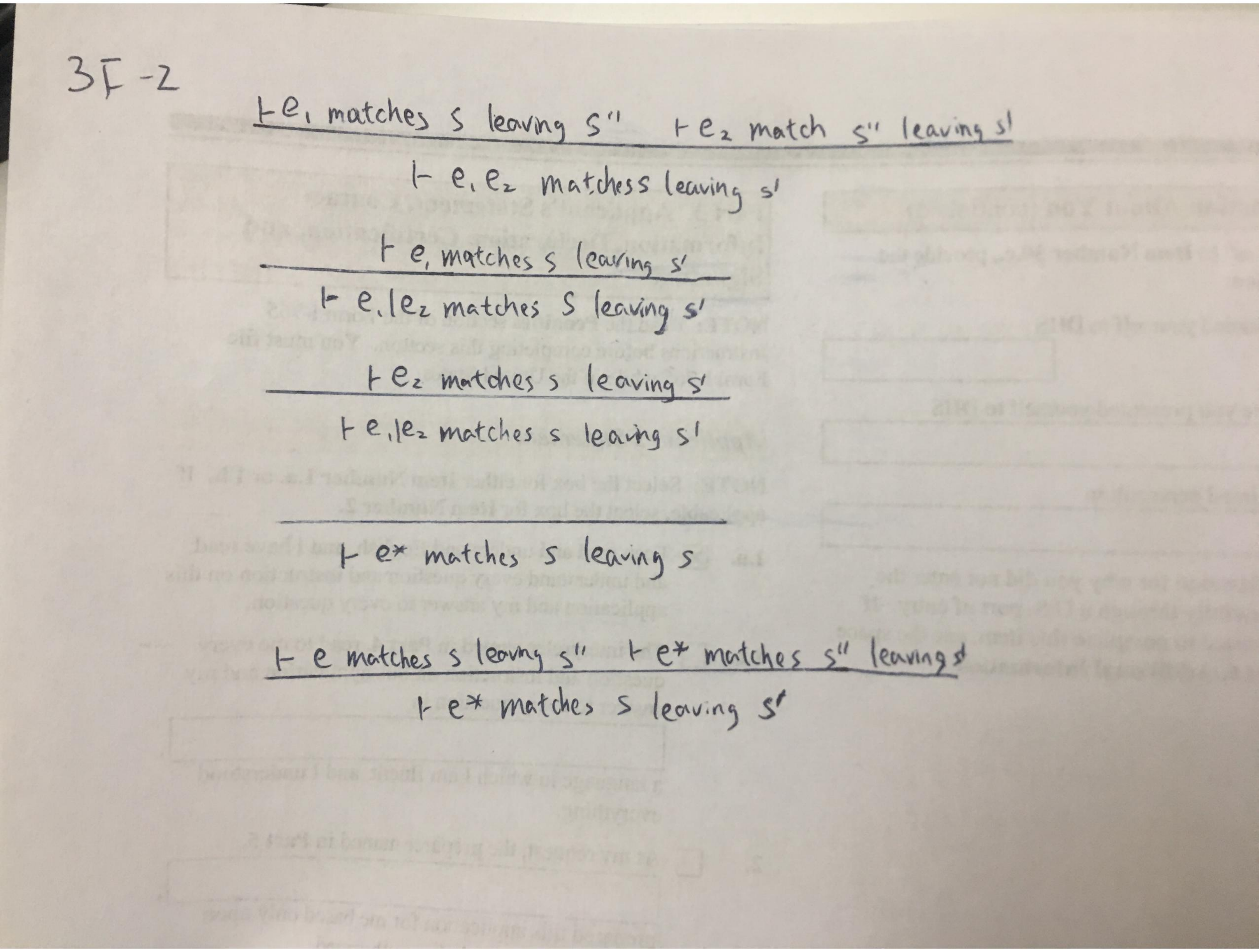
1 3F-1 Bookkeeping

- **0 pts** Correct

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope



3F-2

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } s'' \qquad \vdash e_2 \text{ match } s'' \text{ leaving } s'}{\vdash e_1 e_2 \text{ matchess leaving } s'}$$

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

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

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

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

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope

2 3F-2 Regular Expressions, Large Step

- **0 pts** Correct

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope

3F-3

It cannot be done correctly in the given framework. Because there is no finite and fixed set of hypotheses for some inference rules.

For example, I tried to figure out the following rule for $e*$.

$$\frac{\vdash e \text{ matches } s \text{ leaving } S \quad \forall x \in S \vdash e* \text{ matches } x \text{ leaving } S'_x}{\vdash e* \text{ matches } s \text{ leaving } \{s\} \cup_{x \in S} S'_x} \tag{1}$$

In this rule of inference, S might be infinite. Therefore, we cannot get each $x \in S$ and compute S'_x. Even if the S here is finite and fixed, S'_x is not a finite and fixed set. The reason is that e can match s any abtrary times. There are infinite number of possible chains for the rule and we cannot get the final answer.

I also atttempted the following rule for $e_1 e_2$.

$$\frac{\vdash e_1 \text{ matches } s \text{ leaving } S \quad \forall x \in S \vdash e_2 \text{ matches } x \text{ leaving } S'_x}{\vdash e_1 e_2 \text{ matches } s \text{ leaving } \cup_{x \in S} S'_x} \tag{2}$$

For the same reason, S might be infinite. We cannot write down an inference rule that has a countably or uncountably infinite number of hypotheses.

3 3F-3 Regular Expressions and Sets

- 0 pts Correct

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope

3F-4

To decide wheather $e_1 \sim e_2$, we need a decision procedure that takes two regular expressions and returns a boolean answer. The problem is undecideable because there does not exist any possible correct terminating algorithm A that solves it.

Assume we can solve the problem by the algorithm `ISEQUAL(e_1, e_2)`. I do not know how to reduce it to a halting problem.

3F-5

The last two test cases involve a lot of bounded integer search. For each integer, the algorithm searches 256 possible values. If multiple integers exist, the time complexity is exponential time.

To improve the performance, I would rewrite `cnf` to prune the useless search. For example, after the input converted to cnf, we can directly get some variable assignments. To get the whole snf satisfied, each of the clause must be true. The clause `(z=10)` could immediately applied to the model. Then clauses including `z` like `y > z` could be updated. So that a new cnf is used as the input of dpll.

4 3F-4 Equivalence

- 0 pts Correct

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope

3F-4

To decide wheather $e_1 \sim e_2$, we need a decision procedure that takes two regular expressions and returns a boolean answer. The problem is undecideable because there does not exist any possible correct terminating algorithm A that solves it.

Assume we can solve the problem by the algorithm `ISEQUAL(e_1, e_2)`. I do not know how to reduce it to a halting problem.

3F-5

The last two test cases involve a lot of bounded integer search. For each integer, the algorithm searches 256 possible values. If multiple integers exist, the time complexity is exponential time.

To improve the performance, I would rewrite `cnf` to prune the useless search. For example, after the input converted to cnf, we can directly get some variable assignments. To get the whole snf satisfied, each of the clause must be true. The clause `(z=10)` could immediately applied to the model. Then clauses including `z` like `y > z` could be updated. So that a new cnf is used as the input of dpll.

5 3F-5 SAT Solving

- **0 pts** Correct

Peer Review ID: 68558945 — enter this when you fill out your peer evaluation via gradescope