Exercise 0F-2. Set Theory [5 points]. A function $f : B \to A$ is

$$(f(b))(x) = g(x) = \{Q \epsilon P(Y) | (\forall q \epsilon Q((x, q) \epsilon b))\}.$$

Explained in words, this means that when you pass in a set b that is a subset of the set $B = P(X \times Y)$, the function $(f(b))(x)$ returns a function $g(x)$ where $g : X \to P(Y)$. Then, when you pass in x that is an element of X to $g(x)$, it returns a set Q that is an element of $P(Y)$ where Q contains all pairs (x, q) where q is in Q and (x, q) is in b.

We can determine if f is an injection if we prove that $(f(b_1)(x) = f(b_2)(x)) \to (b_1 = b_2)$. We have

$$f(b_1)(x) = \{Q \epsilon P(Y) | (\forall q \epsilon Q((x, q) \epsilon b_1))\}$$

and

$$f(b_2)(x) = \{Q \epsilon P(Y) | (\forall q \epsilon Q((x, q) \epsilon b_2))\}.$$

The two functions are the same except for b, so obviously, for the two functions to be the same, b_1 and b_2 have to be equal. So we know that f is an injection.

We can determine if f is a surjection if we show that $\forall g \epsilon A, \exists b \epsilon B(f(b) = g)$. We know that $g : X - > P(Y)$, so for any given $g(x)$ we know that the output is $Q \epsilon P(Y)$. We know that Q consists of all elements q such that (x, q) is in b, so we can determine what relations need to be in the set b to result in the q's in the output Q. Also, since we know that Q is in $P(Y)$, we know that the only elements we need in b are also part of $P(Y)$. Thus, we know that there is a set $b \epsilon B$ for all possible functions $g(x)$, so f is surjective.

Since f is both injective and surjective, we have proven that f is a bijection.

Exercise 0F-3. Model Checking [10 points]. When you run CPAChecker, it checks if the program can ever reach an error state for the property you pass in. For example, if Property1a is violated, then it goes to an error state. The provided file tcas.i file is a good test suite for CPAChecker because it contains two properties that are violated and one that passes, so it exercises both potential outcomes of running CPAChecker. However, with only three properties to check, it is not very thorough, so to more confidently show that CPAChecker works you would need more tests. As shown in Figures 1, 2, and 3, CPAChecker correctly identifies whether each of the three properties tested is violated or not, so we can be confident to a limited degree that CPAChecker performs its job as intended.

TCAS is a traffic collision avoidance system for aircraft. The simplified version provided in tcas.i runs an altitude separation test. In this test, Property1a is checked when a downward Resolution Advisory (RA) is selected (the software recommends that the aircraft descend). Property1a verifies that the altitude separation between the aircraft and a potential threat above it is greater than or equal to a certain threshold and that the altitude separation between the aircraft and a potential threat below it is less than a certain threshold. The former must be false for a downward RA to be needed, while the latter must be false for a descent to be safe. If both properties are true when a downward RA is issued, then there

should be an error. CPAChecker correctly identifies a violation of Property 1a. Property1b and Property2b check related properties and CPAChecker was run successfully on both of them as well.

CPAChecker was very straightforward to use and makes it easy to identify error states. It comes with a graphical reporting tool that helps visualize the execution path that program takes to reach an error state. For example, Figure 4 shows part of the control flow graph of tcas.i where Property 1a was violated.

```
Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.fromConfigurati
on, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) started (CPAchecker.run, I
NFO)

Parsing CFA from file(s) "../hw0/tcas.i" (CPAchecker.parse, INFO)

Using predicate analysis with MathSAT5 version 5.6.5 (63ef7602814c) (Nov  9 2020 09:01:58, gmp 6.
1.2, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (Pr
edicateCPA:PredicateCPARefiner.<init>, INFO)

Starting analysis ... (CPAchecker.runAlgorithm, INFO)

Stopping analysis ... (CPAchecker.runAlgorithm, INFO)

Verification result: FALSE. Property violation (error label in line 1963) found by chosen configu
ration.
More details about the verification run can be found in the directory "./output".
Graphical representation included in the file "./output/Counterexample.1.html".
```

Figure 1: CPAChecker output for tcas.i on Property1a

```
Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.fromConfigurati
on, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) started (CPAchecker.run, I
NFO)

Parsing CFA from file(s) "../hw0/tcas.i" (CPAchecker.parse, INFO)

Using predicate analysis with MathSAT5 version 5.6.5 (63ef7602814c) (Nov  9 2020 09:01:58, gmp 6.
1.2, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (Pr
edicateCPA:PredicateCPARefiner.<init>, INFO)

Starting analysis ... (CPAchecker.runAlgorithm, INFO)

Stopping analysis ... (CPAchecker.runAlgorithm, INFO)

Verification result: TRUE. No property violation found by chosen configuration.
More details about the verification run can be found in the directory "./output".
Graphical representation included in the file "./output/Report.html".
```

Figure 2: CPAChecker output for tcas.i on Property1b

```
Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.fromConfigurati
on, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) started (CPAchecker.run, I
NFO)

Parsing CFA from file(s) "../hw0/tcas.i" (CPAchecker.parse, INFO)

Using predicate analysis with MathSAT5 version 5.6.5 (63ef7602814c) (Nov  9 2020 09:01:58, gmp 6.
1.2, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (Pr
edicateCPA:PredicateCPARefiner.<init>, INFO)

Starting analysis ... (CPAchecker.runAlgorithm, INFO)

Stopping analysis ... (CPAchecker.runAlgorithm, INFO)

Verification result: FALSE. Property violation (error label in line 1997) found by chosen configu
ration.
More details about the verification run can be found in the directory "./output".
Graphical representation included in the file "./output/Counterexample.1.html".
```

Figure 3: CPAChecker output for tcas.i on Property2b

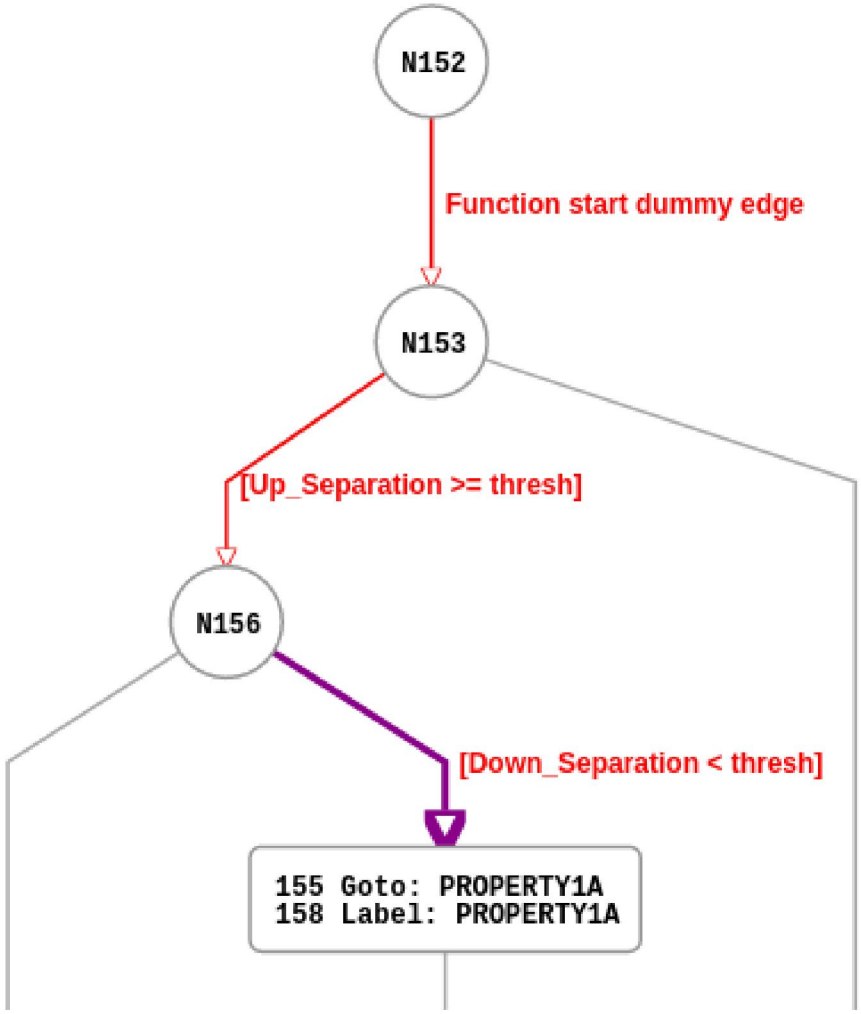

Figure 4: Graphical output of CPAChecker showing where Property1a is violated

1 HW0

- **0 pts** Correct