Exercise 2

Proof: To show that there is a bijection relationship between the sets A and B, we need to prove there exists a function f: A->B which is both an injection and surjection function.

By definition, f: A -> B expands to f: X -> P(Y) → P(X × Y). We construct f such that f(g) = {(x, y) | y ∈ g(x)}, where g represents the set of function A.

To prove that function f is an injection, we must prove for $\forall g_1 \in A$ and $g_2 \in A$, if f(g_1) = f(g_2), then $g_1 = g_2$. Let arbitrary $g_1 \in A$, $g_2 \in A$, and we assume f(g_1) = f(g_2). Then by definition of f, we have {(x, y) | y ∈ g_1 (x)} = {(x, y) | y ∈ g_2 (x)}, meaning for every (x, y), y ∈ g_1 (x) in f(g_1) , there is also one exact same relation (x, y), y ∈ g_2 (x) in f(g_2). Thus, g_1 and g_2 contain the same set of P(Y) no matter what input x is. Hence, we prove that $g_1(x) = g_2(x)$ and function f is an injective function.

To prove that function f is a surjection, we must prove for every element b in the range of function f, there exists one g in the domain of function f such that f(g) = b. Let arbitrary b ∈ B. Then by definition, b = {(x, y) | x ∈ X, y ∈ Y}. We can construct a function g in a way that f(g) = b, or f ∘ g = b. We already know that f(g) = {(x, y) | y ∈ g(x)}. Thus, we construct g(x) = {y | for all y in the set of relation b}. Then by the composition of functions, we have f(g(x)) = {(x, y) | y ∈ b} and by implication, x ∈ X. Thus, the sets of relation in f(g(x)) and b are exactly the same. Hence, f(g(x)) = b and function f is a surjective function.

Since we already demonstrated that function f is both injective and surjective, we can say that function f is a bijection function and thus, there is a bijection relation between the sets A and B. QED.

Exercise 3

Property 1A

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

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

Parsing CFA from file(s) "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 JFactor
y 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (PredicateCPA: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 configuration.
More details about the verification run can be found in the directory "./output".
Graphical representation included in the file "./output/Counterexample.1.html".
```

Property 1B

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

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

Parsing CFA from file(s) "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 JFactor
y 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (PredicateCPA: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".
```

Property 2B

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

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

Parsing CFA from file(s) "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 JFactor
y 1.21. (PredicateCPA:PredicateCPA.<init>, INFO)

Using refinement for predicate analysis with PredicateAbstractionRefinementStrategy strategy. (PredicateCPA: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 configuration.
More details about the verification run can be found in the directory "./output".
Graphical representation included in the file "./output/Counterexample.1.html".
```

What is going on when you run CPAChecker using the commands listed?

When we run the commands, the CPAChecker checks whether the source program tcas.i violates the spec properties that we specified(ex. property 1a). More specifically, the CPAChecker first detected the programming language that the source file is using (c in this case). Then it used the default setting of resources to parse tcas.i into a CFA (control flow automata) structure. Finally, CPAChecker used its internal algorithms to determine whether the CFA structure violates the properties given. If not, CPAChecker generated reports for the test. If violated, CPAChecker also gave a counterexample where the properties were violated.

In at most three paragraphs, summarize your experience with the CPAChecker tool. What does Property1a mean? Is tcas.i a reasonable test suite? What has been proved? Did you find CPAChecker to be a usable tool?

I had a pretty good experience with the CPAChecker tool. I was actually impressed by the UI and the documentation of CPAChecker. I had no previous experience in model checking or programming languages in general. However, I was able to figure out how to use and understand the report generated by CPAChecker in a relatively short period of time. The UI is not perfect, but I would say that it provides a ton of data that could be useful for analysis. After reading the documentation, I also felt that CPAChecker is more flexible on user customizations than I thought a verification software would be.

Property 1a is one of the test cases for the code. If the condition doesn't check out, the code violates property 1a and thus fails to accomplish its purposes. If the condition holds, then we prove that the code obeys this rule. For tcas.i, I believe that it is a reasonable test case, as it proves the CPAChecker works in specific cases. However, it is very far from being a reasonable test suite because it has relatively simple logic and only tests on very basic cases. E.x. tcas.i is a one thread program and the values of its variables that are later checked by the properties don't change much after initialization. Thus, running Property 1a, 1b, 1c only proves the internal validity of the CPAChecker. In general, CPAChecker could be a useful tool, but I don't think a programmer should completely trust the verification result that it gives.

1 HW0

- **0 pts** Correct