Exercise 0F-2

I will demonstrate a bijection $f : B \to A$. Let f be defined by mapping $S \in B$ to $g_S \in A$ where

$$g_S(x) = \{y \,:\, (x, y) \in S\}.$$

I claim that f is a bijection. I will first show that f is injective. Suppose there exist $S_1, S_2 \in B$ such that $f(S_1) = f(S_2)$. I aim to show $S_1 \subseteq S_2$. Consider any $(x, y) \in S_1$. By definition, we have that $y \in g_{S_1}(x)$ and by equality of $f(S_1)$ and $f(S_2)$, we have that $g_{S_1}(x) = g_{S_2}(x)$, so $y \in g_{S_2}(x)$. This implies by definition of g_{S_2} that $(x, y) \in S_2$. Therefore $S_1 \subseteq S_2$. By symmetry, we also have that $S_2 \subseteq S_1$ and so $S_1 = S_2$ whenever $f(S_1) = f(S_2)$. That is, f is injective.

Now I will show that f is surjective. Consider any function $g \in A$. Let

$$S_g := \{(x, y) \,:\, x \in X, y \in g(x)\}.$$

Then it is clear by definition of f that $f(S_g) = g$, and so f is surjective. We have seen that f is both surjective and injective, and is thus a bijection, as wanted. $\square$

Question assigned to the following page: 3

Exercise 0F-3

Output of property1a:

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

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13) started (CPAchecker.run, INFO)

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May 31 2023 12:38:06, gmp 6.2.0, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:Predi
ateCPA.<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".
```

Output of property1b:

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

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13) started (CPAchecker.run, INFO)

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May 31 2023 12:38:06, gmp 6.2.0, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:Predic
ateCPA.<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".
```

Output of property2b:

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

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13) started (CPAchecker.run, INFO)

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May 31 2023 12:38:06, gmp 6.2.0, gcc 7.5.0, 64-bit, reentrant) and JFactory 1.21. (PredicateCPA:Predic
ateCPA.<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".
```

When CPAChecker is run as prescribed, say with property1a, it aims to prove that reaching the label `PROPERTY1A` is impossible. CPAChecker knows to check this label because `property1a.spc` prescribes this label as the only error condition. When CPAChecker finishes running, it either outputs a proof that there is no way to reach this label, or gives an example execution trace that leads to this label. It appears that CPAChecker could also time out, and terminate with neither a counterexample nor a claim of correctness, but I did not observe this behavior during my testing.

The supplied `tcas.i` file appears to be a basic system for determining whether an aircraft should ascend, descend, or maintain altitude based on various properties of the aircraft and one other aircraft. The property testing functions aim to describe correct behavior of the aircraft given these properties. In particular, `property1a` is checked whenever the system has determined the aircraft should *descend*. This property requires that there either be sufficient space to descend, or insufficient space to ascend, based on some threshold value. This appears like a reasonable property to enforce, as if there isn't sufficient space to descend, then we should only descend if forced. In fact, it's possible the property should be made stronger: we should only descend if we have sufficient space to descend or if we don't even have sufficient space to maintain altitude. However, CPAChecker determines that this property is in fact not maintained by the code, so there is some issue associated with the decision process for descending. Remarkably, `property1b` models the symmetric condition when the aircraft is advised to ascend, and is maintained by the code, so there is some asymmetry in the decision process that tells the aircraft to descend when it should not.

The CPAChecker tool provides several useful counterexample views when an error state is reached. I found the "ARG" tab to be particularly useful as it shows exactly which predicates must be true and the path through the

Question assigned to the following page: 3

program that reaches the error state in an easy to follow flow diagram. It is quite simple to indicate error states to CPAChecker, requiring only inserting a label in the correct spot and indicating it in the appropriate spc file. Thus, due to its ease of use and simple interface, there is relatively little room for user error to threaten the validity of CPAChecker.