OF-2)

Let $f: A \to B := \lambda a \to \bigcup_{x \in X} \{(x,y) \mid y \in a(x)\}$

Let $b \in B$ and $a(x) = \{y \mid (x,y) \in b\}$

Now $a \in A$ and $f(a) = \bigcup_{x \in X} \{(x,y) \mid y \in \{y \mid (x,y) \in b\}\}$

$= b$

Thus f is surjective.

Let $a, a' \in A$ and assume $f(a) = f(a')$, so

$\bigcup_{x \in X} \{(x,y) \mid y \in a(x)\} = \bigcup_{x \in X} \{(x,y) \mid y \in a'(x)\}$

so for any x $\{(x,y) \mid y \in a(x)\} = \{(x,y) \mid y \in a'(x)\}$

so $a(x) = a'(x)$, so $a = a'$. Thus f is injective.

Question assigned to the following page: 3

Last ten lines of output for runs:

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:PredicateCPA.<init>, INFO)

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

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:PredicateCPA.<init>, INFO)

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

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:PredicateCPA.<init>, INFO)

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

Question assigned to the following page: 3

I learned from Wikipedia that TCAS is an aircraft collision avoidance system which tries to predict a potential collision and then suggest a safe flight path to the pilot. This specific file is testing a function responsible for determining what instruction (ascend or descend) the TCAS system should give, given a set of parameters about two planes. The function has been instrumented with a set of error states which may be hit if the function outputs an unsafe result. For example, property 1a is an error state reached if TCAS instructs the pilot to climb when the other plane is far above them.

In the given commands, CPAChecker is being used to find any execution paths which reach these error states, or else determine that none exist. We are providing CPAChecker with the function to be tested (in tcas.i) and a property to check (in Property__.spc). tcas.i has been preprocessed with labeled error states and Property__.spc contains an temporal logic automaton which specifies a label which CPAChecker should ensure is unreachable. The –predicateAnalysis command line algorithm CPAChecker to do this using the CEGAR procedure.

CPAChecker finds that properties 1a and 2b are not maintained by providing possible execution paths where the properties are violated, and verifies that there is no path which reaches the 1b error condition, proving property 1b is maintained. CPAChecker outputs an HTML based explorer for the counterexamples for properties 1a and 2b which makes it very easy to search for bugs. I do not understand the input function well, but by following the visualized red line through the control flow automaton and reading the provided states of all variables at each step, I was able to find a discrepancy in the code which appeared to be causing the violation of 1a. I am not confident that this discrepancy was really the bug, but the fact I got as far as I did suggests that this would be an excellent tool for someone more familiar with the code. I used the docker image to run CPAChecker, and setting it up and providing the input files and configuration was painless.