Exercise 0F-2. Set Theory [5 points]. This answer should appear after the first page of your submission and may be shared during class peer review.

Solution: Let us define a function $f : A \to B$. We will prove that f is an injection and a surjection. First let us define $f : \forall a \in A,\ f(a) = \{(x, y)|x \in X, y \in a(x)\}$.

Next we prove our function is surjective. Let $b \in B$ be arbitrary. Next, let us define a function $a : X \to \mathcal{P}(Y)$, where $\forall x \in X, a(x) = \{y|(x, y) \in b\}$. Then $f(a) = b$, so f is surjective.

Finally we prove our function is injective. Let $a_1, a_2 \in A, a_1 \neq a_2$, and assume that $f(a_1) = f(a_2)$. Then, $\{(x, y)|x \in X, y \in a_1(x)\} = \{(x, y)|x \in X, y \in a_2(x)\}$. So, $\forall x \in X,\ a_1(x) = a_2(x)$, therefore $a_1 = a_2$.

Exercise 0F-3. Model Checking [10 points]. This answer should appear after the first page of your submission and may be shared during class peer review.

Solution:

When we run CPAChecker with the commands listed, CPAChecker is running a predicate analysis, similar to the discussion of SLAM in lecture. In our case, we are running the analysis on `tcas.i` and we specify what to check for as an error with the variable property.spc files. For example, when we run our command with Property1a, we check if we reach the function "property1a." We deduce this from the regex expression mapping "property1a" to an error state. `tcas.i` is complicated but a reasonable test suite as it was designed for traffic collision avoidance. Property1a and Property2a are both defined as error states (within `tcas.i` and the .spc files); the the CPAChecker found that both labels could be reached within `tcas.i` and provided a concrete example of reaching them. So, we proved that the software used to avoid aviation collisions can reach error states, aka lead to collisions. Proof of reaching these error states was provided in the output of the CPAChecker, discussed further below.

As for running CPAChecker, the inputs to the analysis were straightforward: the program to be verified, and a file specifying what should be identified as an error state. The error states were each defined with a regular expression that matched a label found in the program being analyzed. The output of CPAChecker is guaranteed to be either be 1) confirmation that an error state cannot be reached or 2) an explicit example of how to reach an error state. In the case that the analysis finds that the program can reach an error state, the graphical output contains a reachability tree, specifying the exact initial state and transitions that would need to occur to violate the specified properties and reach an error state. With our given inputs, the graphical output provides a concrete example of when the traffic collision avoidance system algorithm could reach an error state.

Question assigned to the following page: 3

Property 1a:

```
Language C detected and set for analysis (CPAMain.detectFrontendLanguageIfNecessary, INFO)

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, reentra
nt) and JFactory 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:

```
Language C detected and set for analysis (CPAMain.detectFrontendLanguageIfNecessary, INFO)

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, reentra
nt) and JFactory 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 2a:

```
Language C detected and set for analysis (CPAMain.detectFrontendLanguageIfNecessary, INFO)

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, reentra
nt) and JFactory 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".
```