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.

This exercise is meant to help you refresh your knowledge of set theory and functions. Let X and Y be sets. Let $\mathcal{P}(X)$ denote the powerset of X (the set of all subsets of X). There is a 1-1 correspondence (i.e., a bijection) between the sets A and B, where $A = X \to \mathcal{P}(Y)$ and $B = \mathcal{P}(X \times Y)$. Note that A is a set of functions and B is a (or can be viewed as a) set of relations. This correspondence will allow us to use functional notation for certain sets in class. This is Exercise 1.4 from page 8 of the Winskel textbook.

Demonstrate the correspondence between A and B by presenting an appropriate function and proving that it is a bijection. For example, you might construct a function $f : B \to A$ and prove that f is an injection and a surjection.

Proof. Let us define a function $f : B \to A$ that converts an element of B (in this case a relation) into an element of A (a function). Based on this, we can begin by stating that $f_R : X \to P(Y)$, where $R \in B$. For every $x \in X$, we will define $f_R(x)$ as all y values that satisfy $(x, y) \in R$. In order to prove that it is a bijection, we will prove that f is both an injection and a surjection.

In order to prove injectivity, we want to show that if $f_{R_1} = f_{R_2}$, then $R_1 = R_2$. Using our definition of f_R, we can represent $f_{R_1}(x)$ as $\{y \in Y | (x, y) \in R_1\}$ and $f_{R_2}(x)$ as $\{y \in Y | (x, y) \in R_2\}$. Since we assumed that $f_{R_1} = f_{R_2}$, we know that for all $x \in X$, $\{y \in Y | (x, y) \in R_1\} = \{y \in Y | (x, y) \in R_2\}$. Based on this, we can then surmise that $(x, y) \in R_1 \iff (x, y) \in R_2$ and $R_1 = R_2$. Thus, we have proved that f is an injection.

Next, we will prove that f is a surjection. We need to show that for every function $a \in A$, there exists a $R \in B$ such that $f_R = a$. Let us then define a relation $R_1 \in B$ that represents the set of all (x, y) where y is a subset of $a(x)$. In other words, $R_1 = \{(x, y) | y \in a(x)\}$. We can see that $f_{R_1}(x) = \{y \in Y | (x, y) \in R_1\}$, which matches our definition of a. Therefore, $f_{R_1} = a$, and f is surjective.

Since f is both an injection and a surjection, we have shown that it is a bijection. Therefore, we have proved the correspondence between A and B. □

Exercise 0F-3. Model Checking [10 points].

Proof. Property1a:

```
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:

Question assigned to the following page: 3

```
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:

```
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 the CPAChecker is run with a specified property, builds an abstract representation of the program and verifies if the property holds within the abstracted control flow. Property1a checks for error locations following a "property1a" label. Based on the source code, if need_downward_RA is 1, Up_Separation is greater than or equal to the threshold, and Down_Separation is less than the threshold, Property1a.spc will identify an error. In this case, an error was identified and the location of the PROPERTY1A error on line 1963 was outputted. Property1b was found to hold, while an error state was reached for Property2b as well.

The source code tcas.i tests with non-deterministic values. This allows it to verify a multitude of possibilities but may not replace a functional test suite with real-world inputs. As such, tcas.i may not be the most reasonable test suite. If the CPAChecker verifies that no error states are reached for a property, it proves that the property holds. If it reaches an error state, it suggests that the property may be violated and that the source code should be reviewed.

I found CPAChecker to be a relatively usable tool. Inputs can be fairly easily provided to CPAChecker and verified. The graphical HTML output shows the possible paths to an error state, as well as the intended default return path. If an error state is reached, the counterexample path is highlighted in red.