Exercise 0F-2. Set Theory [5 points]. Let X and Y be sets, and let $\mathcal{P}(X)$ denote the powerset of X.

Proposition. *There is a bijective correspondence between the sets* $A = X \to \mathcal{P}(Y)$ *and* $B = \mathcal{P}(X \times Y)$.

Proof. By Schröder-Bernstein, there is a bijective correspondence between A and B if there is an injective map from A to B and an injective map from B to A.

Let $n \in \mathbb{N}$ and define $f : A \to B$ by

$$f(x \mapsto \{y_1, y_2, \dots, y_n\}) = \{(x, y_1), (x, y_2), \dots, (x, y_n)\}$$

and define $g : B \to A$ by

$$g(\{(x, y_1), (x, y_2), \dots, (x, y_n)\}) = x \mapsto \{y_1, y_2, \dots, y_n\}$$

We claim that f and g are injective and proceed by contradiction.

Suppose f is not injective. Then for some $n, n' \in \mathbb{N}$ there exist

$$a_1 = x_1 \mapsto \{y_{11}, y_{12}, \dots, y_{1n}\} \qquad a_2 = x_2 \mapsto \{y_{21}, y_{22}, \dots, y_{2n'}\}$$

such that $a_1 \neq a_2$ and $f(a_1) = f(a_2)$. If $a_1 \neq a_2$, then $x_1 \neq x_2$ or $n \neq n'$ or $y_{1k} \neq y_{2k}$ for some $k \in \{1, 2, \dots, n\}$. On the other hand, if $f(a_1) = f(a_2)$, then

$$\{(x_1, y_{11}), (x_1, y_{12}), \dots, (x_1, y_{1n})\} = \{(x_2, y_{21}), (x_2, y_{22}), \dots, (x_2, y_{2n'})\}$$

By construction, $x_1 = x_2$ and $n = n'$ and $y_{1j} = y_{2j}$ for all $j = 1, \dots, n$ so we have $a_1 = a_2$, which contradicts our assumption that $a_1 \neq a_2$. Thus, f is an injective map.

The same argument applies to g. Suppose g is not injective. Then for some $n, n' \in \mathbb{N}$ there exist

$$b_1 = \{(x_1, y_{11}), (x_1, y_{12}), \dots, (x_1, y_{1n})\} \qquad b_2 = \{(x_2, y_{21}), (x_2, y_{22}), \dots, (x_2, y_{2n'})\}$$

such that $b_1 \neq b_2$ and $g(b_1) = g(b_2)$. If $b_1 \neq b_2$, then $x_1 \neq x_2$ or $n \neq n'$ or $y_{1k} \neq y_{2k}$ for some $k \in \{1, 2, \dots, n\}$. On the other hand, if $g(b_1) = g(b_2)$, then

$$x_1 \mapsto \{y_{11}, y_{12}, \dots, y_{1n}\} = x_2 \mapsto \{y_{21}, y_{22}, \dots, y_{2n'}\}$$

By construction, $x_1 = x_2$ and $n = n'$ and $y_{1j} = y_{2j}$ for all $j = 1, \dots n$, and we have $b_1 = b_2$, which contradicts our assumption that $b_1 \neq b_2$. Thus, g is an injective map and, therefore, there exists a bijective correspondence between A and B. □

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

Property 1a

```
scripts/cpa.sh -predicateAnalysis -spec ../Property1a.spc ../tcas.i
```

```
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 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)

Running CPAchecker with default heap size (1200M). Specify a larger value with -heap if
↪  you have more RAM.
Running CPAchecker with default stack size (1024k). Specify a larger value with -stack if
↪  needed.
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

```
scripts/cpa.sh -predicateAnalysis -spec ../Property1b.spc ../tcas.i
```

```
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 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 2b

```
scripts/cpa.sh -predicateAnalysis -spec ../Property2b.spc ../tcas.i
```

```
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 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".
```

`tcas.i` is a C++ source file that has been run through a preprocessor. When I run CPAChecker, it determines whether a property holds for all possible execution paths. When a property is violated, CPAChecker prints the line of code where it found the violation and generates a comprehensive report with a counterexample and a detailed account of the execution path that led to it.

Consider property 1a, which specifies that arriving at the label `PROPERTY1A` is a violation. A predicate analysis on `tcas.i` for this property reports a violation at line 1963. According to the reported execution path, this occurs when all of the following criteria are met:

- The `alt_sep_test` routine is enabled
- TCAS is not equipped or its intent is not known
- Downward RA is needed but upward RA is not
- Up-separation has exceeded a given threshold that down-separation has not

`tcas.i` is a self-contained file that demonstrates both successful and failed properties, so it is at least marginally reasonable as a test suite. On the other hand, my understanding of what it means to violate one property versus another is impeded by a lack of knowledge of traffic control automation. So, although `tcas.i` is useful for understanding the behavior of CPAChecker in the abstract, and for demonstrating its utility at scales of realistic size and consequence, I believe smaller programs from more accessible application spaces—say, a register allocator, or a concurrency primitive implemented in a higher level language and transpiled to C—are necessary for building confidence in CPAChecker's ability to perform as intended.

1 HW0

- **0 pts** Correct

Peer Review ID: 62020635 — enter this when you fill out your peer evaluation via gradescope