Exercise 0F-2. Set Theory

Let us define a function $f : B \to A$. For a given set of pairs $b \in B$, we will define $f(b)$ as the function $f(x \in X) = \{y \mid (x, y) \in b\}$. We will now show that f is injective.

1. Let us define $b_1, b_2 \in B$, such that $f(b_1) = f(b_2)$.
2. $\forall (x, y) \in b_1 . y \in f(b_1)(x)$.
3. $f(b_1) = f(b_2) \implies f(b_1)(x) = f(b_2)(x) \implies \forall (x, y) \in b_1 . y \in f(b_2)(x)$.
4. $\forall (x, y) \in b_1 . y \in f(b_2)(x) \implies (x, y) \in b_2$.
5. $\forall (x, y) \in b_1 . (x, y) \in b_2 \implies b_1 \subseteq b_2$.
6. We can trivially reverse b_1 and b_2 to show that $b_2 \subseteq b_1$. This means $b_1 = b_2$, therefore $f(b_1) = f(b_2) \implies b_1 = b_2$, therefore f is injective.

We will now show that f is surjective:

1. Let us define a function $a : X \to \mathcal{P}(Y) \in A$.
2. We construct a set $b = \bigcup_{x \in X} \{x\} \times a(x)$. In other words, a set of all pairs (x, y) such that $x \in X, y \in a(x)$.
3. For every pair $(x, y) \in b$, $y \in Y$, since it is an element of a member of a's codomain - $\mathcal{P}(Y)$.
4. For every pair $(x, y) \in b$, $x \in X \wedge y \in Y \implies (x, y) \in X \times Y$. Consequently, $b \in B$.
5. $f(b)$ yields a function $f'(x \in X) = \{y \mid (x, y) \in b\}$. As established, $\forall y \in b . y \in Y$. Therefore, the codomain of f' is a subset of $\mathcal{P}(Y)$. And by our construction of f', the domain of f' is X.
6. Therefore, $f' : X \to \mathcal{P}(Y) \implies f' \in A$. Thus, $\forall a \in A . \exists b \in B . f(b) = a$, and f is surjective.

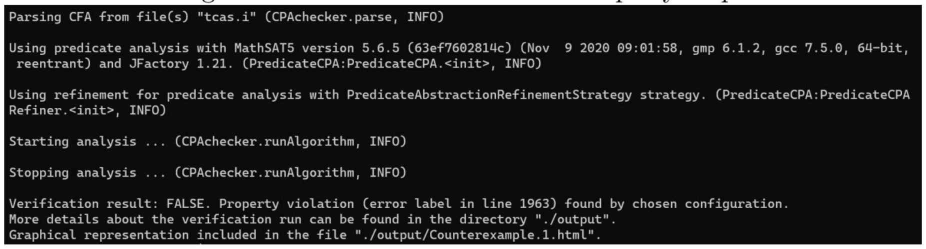
As f is both surjective and injective, it is bijective. Thus, there does exist a bijective function $f : B \to A$, which implies a correspondence between A and B.

Exercise 0F-3. Model Checking

If my understanding is correct, when CPAChecker is run on tcas.i, it interprets the program incompletely, abstracting the real values of variables involved into true or false predicates about their values. Let's examine testing property 1a as an example. In this test, we error if `Up_Separation` is greater than or equal to `thresh` and `Down_Separation` is less than or equal to `thresh` - these are our relevant predicates. In addition, Property 1a is only tested in one branch, reachable through the passage of a number of conditions, which depend on global variables such as `Own_Tracked_Alt` - each of these conditions becomes a check for an additional predicate. For CPAChecker to report a failure when testing Property 1a implies that the predicates `Up_Separation >= thresh` and `Down_Separation < thresh` could not be proven to be true simultaneously in property 1a's control flow - meaning that it was *possible* that said property does not always hold. For other properties that do not fail however, such as 1b, CPAChecker's result is stronger. We *prove*, to the extent of CPAChecker's understanding of the program, that an incorrect program state as defined by property 1b is not reachable.

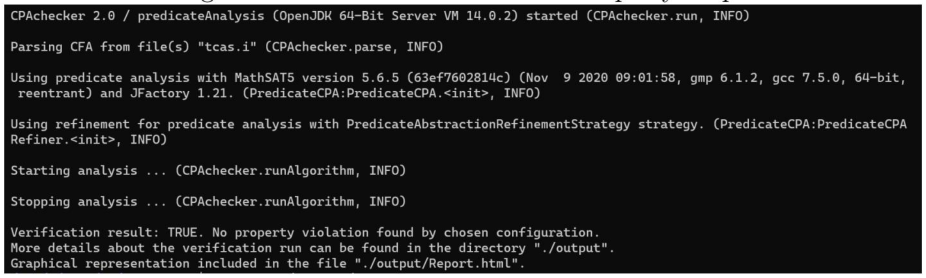
However, for a number of reasons, I do not consider this result to be terribly useful. tcas.i is a weak test case for this tool, and ultimately I don't think we've proved anything terribly interesting. As it happens, most of the properties tested are global variables that are never written to after initialization, and they are initialized to the verifier as non-deterministic values. The program has no loops, simple logic, and minimal mutation of variables. Ultimately, the results we get are more the assessment of how a few random numbers fit together than a useful exploration of complex behavior. This may be a weakness of tcas.i specifically - there are essentially no constraints on the numbers the properties are testing, as far as I can tell, and they are never mutated, so our results depend less on the actual program behavior than on how some numbers are initialized. It could also be interpreted as a weakness of verification tools like CPAChecker in general. Since we don't prove failure, only fail to prove success, a small ambiguous programs like tcas.i's failure on some property could have pretty high odds of being a false positive. Perhaps in a real system the global variables each property is testing fall into relatively clear ranges, or have certain patterns that would realistically preclude a failure from ever occurring. Or, perhaps the failures reported are real concerns. It's impossible to know, since we never showed CPAChecker the rest of the system or gave it better constraints. Ultimately, I don't think CPAChecker is being used to its full potential on tcas.i. Its strengths would probably be better-shown on a larger, more complex program, where values go through many more operations and predicate values are informed more by the program logic than random starting conditions.

Figure 1: CPAChecker results for Property1a.spc



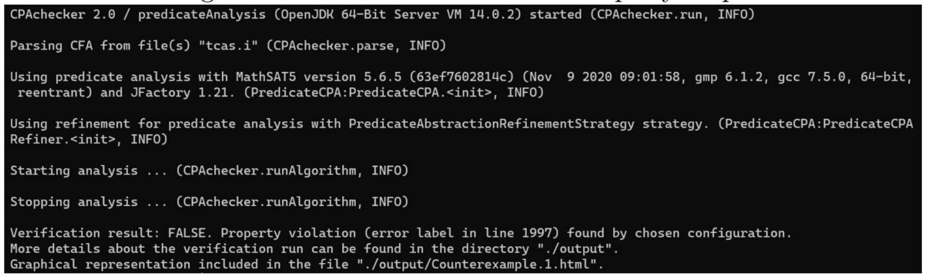
```
Parsing CFA from file(s) "tcas.i" (CPAchecker.parse, INFO)

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:PredicateCPA
Refiner.<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".
```

Figure 2: CPAChecker results for Property1b.spc



```
CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 14.0.2) started (CPAchecker.run, INFO)

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:PredicateCPA
Refiner.<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".
```

Figure 3: CPAChecker results for Property2b.spc



```
CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 14.0.2) started (CPAchecker.run, INFO)

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:PredicateCPA
Refiner.<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".
```

1 HW0

- **0 pts** Correct