Exercise 0F-2

We define $f : B \to A$ to be as follows. For some $\beta \in B, x \in X$,

$$f(\beta)(x) = \{y \mid (x, y) \in \beta\}$$

We define $g : A \to B$ to be as follows. For some $\alpha \in A$

$$g(\alpha) = \{(x, y) \mid x \in X, y \in \alpha(x)\}$$

We will show f is a bijection by showing g is the inverse of f.
For any $\beta \in B$,

$$\begin{aligned} g(f(\beta)) &= \{(x, y) \mid x \in X, y \in f(\beta)(x)\} \\ &= \{(x, y) \mid x \in X, y \in \{y' \mid (x, y') \in \beta\}\} \\ &= \{(x, y) \mid (x, y) \in \beta\} \\ &= \beta \end{aligned}$$

For any $\alpha \in A, x \in X$,

$$\begin{aligned} f(g(\alpha))(x) &= \{y \mid (x, y) \in g(\alpha)\} \\ &= \{y \mid (x, y) \in \{(x', y') \mid x' \in X, y' \in \alpha(x')\}\} \\ &= \{y \mid (x, y) \in \alpha(x)\} \\ &= \alpha(x) \end{aligned}$$

Since f is invertible, f is a bijection.

Exercise 0F-3

0.1 CPA Output

1. Property1a.spc

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

Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.
    fromConfiguration, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) 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: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".
```

2. Property1b.spc

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

Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.
    fromConfiguration, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) 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: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".
```

3. Property2b.spc

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

Using the following resource limits: CPU-time limit of 900s (ResourceLimitChecker.
    fromConfiguration, INFO)

CPAchecker 2.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 11.0.9.1) 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: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".
```

0.2 CPA discussion

When we run CPAChecker with the commands listed, we are having CPAChecker check whether the program "tcas.i" (a simplified traffic collision avoidance system test suite) satisfies the properties defined in Property1a.spec, Property1b.spec, and Property1c.spec. Each of these properties test whether the program reaches a respective PROPERTY1A, etc., states. These states can only be reached if the program violates a property relating the `Up_Separation` and `Down_Separation` variables, making sure they are above or below the `thresh` variable. Specifically, Property1a.spec, verifies that after running `Non_Crossing_Biased_Climb()` and `Non_Crossing_Biased_Descend()`, and only `Own_Above_Threat()` returns True, either `Up_Separation < thresh` or `Down_Separation >= thresh` . In using CPAChecker, we proved that Propert1a, and Property2b are violated in some configurations, but never Property1b.

Is tcas.i a reasonable test suite? If we are writing a paper on a new TCAS system, likely not. Many more traffic configurations can occur that can occur than seem to be possible to cover in the short code of tcas.i. If we want to show our TCAS system is safe, we also must show that every possible scenario is covered by the suite, which is not clear from the code or from CPAChecker output.

In my experience of using CPAChecker, I found it difficult to follow the graphical output's counter example. The counter example shown has so many edges and seemingly irrelevant variables that make it hard to figure out what is important. Perhaps for a person more familiar with CPAChecker or with the tcas.i code they could figure out what the counter example meanss.

1 HW0

- **0 pts** Correct