Exercise 0F-2

let $F : B \to A$ be the function defined as :

$F(S) = f_S : X \to P(Y)$ where $f_S(x) = \{ y \mid (x,y) \in S \}$,

where $S \in B$, $x \in X$

Proving injection :

let $F(S) = F(S')$ where $S, S' \in B$

Need to show : $S = S'$

$F(S) = F(S')$

$\Rightarrow f_S = f_{S'}$ (By definition of F)

$\Rightarrow \forall x \in X, \; f_S(x) = f_{S'}(x)$

$\Rightarrow \forall x \in X, \; \{ y \mid (x,y) \in S \} = \{ y \mid (x,y) \in S' \}$ (*)

(By definition of f)

let $(x', y') \in S$ where $x' \in X$ and $y' \in Y$

$\Rightarrow y' \in \{ y \mid (x', y) \in S \}$

$\Rightarrow y' \in \{ y \mid (x', y) \in S' \}$ (By (*))

$\Rightarrow (x', y') \in S'$

Thus, $(x', y') \in S \Rightarrow (x', y') \in S'$

By symmetry $(x', y') \in S' \Rightarrow (x', y') \in S$

$\Rightarrow S = S'$

F is injective

Proving surjection :

Need to show : $\forall f \in A, \exists S \in B$ such that $F(S) = f$

let $f \in A$.

Consider the following $S \in B$:

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

$F(S) = f_S$

$\forall x \in X$, $f_S(x) = \{ y \mid (x,y) \in S \}$ (By definition)

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

(By definition of S)

$= f(x)$

$\Rightarrow \quad f_S = f$

$F(S) = f$

F is surjective.

Since F is injective and surjective, F is bijective.

Exercise 0F-3

```
vagrant@vagrant:/vagrant/hw0$ CPAchecker-2.0-unix/scripts/cpa.sh -predicateAnalysis -spec Property1a.spc tcas.i
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 11.0.5) 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 JFact
ory 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".
vagrant@vagrant:/vagrant/hw0$ CPAchecker-2.0-unix/scripts/cpa.sh -predicateAnalysis -spec Property1b.spc tcas.i
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 11.0.5) 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 JFact
ory 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".
vagrant@vagrant:/vagrant/hw0$ CPAchecker-2.0-unix/scripts/cpa.sh -predicateAnalysis -spec Property2b.spc tcas.i
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 11.0.5) 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 JFact
ory 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".
```

Peer Review ID: 61896718 — enter this when you fill out your peer evaluation via gradescope

[What is going on?] When I run CPAChecker with the listed commands, CPAChecker performs a program analysis using the rule I have provided as part of the "-spec" argument. CPAChecker converts the tcase.i source code into control-flow automata (CFA) and the performs reachability analysis using MathSAT (a satisfiability solver). The conditions for which satisfiability is checked are the ones provided in .spc files. In case of property 1a, the condition being checked is "can control reach an error location labeled 'PROPERTY1a'? "

[Is tacos.i a reasonable suit?] tcase.i is a reasonable test suite because it carefully constructs various edge cases using variables to check if the their reachability can be established. For property 1a and 2b, CPAChecker successfully finds counterexamples that can cause error statements to be reached. **[What has been proved?**] This proves that these properties can be violated. For property 1b, the tool fails to find a property violation. Since the tool fails to find a violation in the abstract state, there cannot exist an execution path that violates property 1b.

[Did you find CPA checker to be a usable tool?]CPAChecker is a fairly usable tool. I used an Ubuntu VM on Mac, and the installation and use was easy. Generating reports for each property required only a single command. The terminal output was informative. I found the html counterexample output a difficult to understand. The tooltip suggested that double-clicking an edge on a CFA would link me to the line in the source code, but when I double-clicked the edge, it took me to the start of the source file. **[Threats to validity]** Running CPAChecker successively on failing properties also seemed to override the previous counterexamples. After I run CPAChecker for property 2b, the html counterexample for property 1a got overwritten. This could have easily avoided if the developers of CPAChecker had taken care to name the counterexample files systematically.

1 HW0

- **0 pts** Correct