Exercise 0F-2. Set Theory

Let X and Y be sets and let A be the set of functions $X \to \mathcal{P}(Y)$, while $B = \mathcal{P}(X \times Y)$. We will construct a function $f : B \to A$ and show that is is bijective, thereby demonstrating a one-to-one correspondence between the sets A and B.

Proof. First we will construct f. Let $b \in B$ be an arbitrary element. By definition of powerset, $b \subseteq X \times Y$. We will define $f(b)$ as a function $g : X \to \mathcal{P}(Y)$ defined as follows:

$$g(x) = \{y \mid (x, y) \in b\}, \quad \text{for all } x \in X$$

Next we will show that f is *bijective*.

- **Injectivity:** Let $b_1, b_2 \in B$ such that $f(b_1) = f(b_2)$. We aim to show that $b_1 = b_2$, proving that f is injective.

  By definition of f, for any $x \in X$:

  $$f(b_1)(x) = \{y \mid (x, y) \in b_1\}, \quad f(b_2)(x) = \{y \mid (x, y) \in b_2\}.$$

  Since $f(b_1) = f(b_2)$, it follows that:

  $$\{y \mid (x, y) \in b_1\} = \{y \mid (x, y) \in b_2\}, \quad \text{for all } x \in X.$$

  This equality implies that for all (x, y), $(x, y) \in b_1$ if and only if $(x, y) \in b_2$.

  Thus, $b_1 = b_2$, showing that f is injective.

- **Surjectivity:** Let $a \in A$ be an arbitrary element. By definition, a is a function $a : X \to \mathcal{P}(Y)$. We aim to construct an element $b \in B$ such that $f(b) = a$.

  Define:

  $$b = \{(x, y) \mid x \in X \text{ and } y \in a(x)\}.$$

  By construction, $b \subseteq X \times Y$, so $b \in B$.

  Now, consider $f(b)$. By definition of f, we have:

  $$f(b)(x) = \{y \mid (x, y) \in b\}, \text{ for all } x \in X.$$

  Substituting the definition of b, it follows that:

  $$f(b)(x) = \{y \mid y \in a(x)\}.$$

  Thus, $f(b)(x) = a(x)$ for all $x \in X$, which means $f(b) = a$.

  Since $a \in A$ was arbitrary, f is surjective.

We have constructed a function $f : B \to A$ and shown it is a bijection. Thus the result holds. □

Question assigned to the following page: 3

Exercise 0F-3. Model Checking

The program being analyzed seems to be some sort of aircraft control system. When running CPAChecker against the program with the specification it proves whether or not the specified label is reachable for the program. In the event that it is reachable it returns a concrete counter-example trace, control-flow automaton, and abstract reachability graph in a self-contained HTML file for analysis.

My interpretation of `property1a` is that an downward resolution advisory should only be given if the plane above is within some threshold and the plane below is at least the threshold away. The error path displayed by CPAChecker seems to occur because there is an unsigned int overrun on line 1875 where `Up_Separation + 100` becomes `4294967295` and overflows to `88` and is no longer greater than down. In the counterexample, the `Up_Separation` is significantly bigger than the threshold and `Down_Separation` is smaller than the threshold but a downward resolution is still given. The properties can probably be addressed by adding some preconditions on the unsigned integers to prove they are within some predetermined range to avoid overflow. CPAChecker proved that it is possible for this incorrect downward resolution advisory to be given.

The CPAChecker tool was very easy to install with docker and use from the command line. I think integration with an IDE or language server protocol would make the tool much easier to use than it's current incarnation. The separated visual output of the code being distinct from the editor makes it hard to navigate the codebase as you would if you were using your regular tooling. In addition going from the error path back to the CFA/ARG feels cumbersome. It's also hard to keep track of predicates that have been assumed throughout the program in the error path view. As a "step debugger" I think the representation is fine but it struggles when you're trying to quickly drive into how something happened. I also could not seem to figure out how to go from the error path to specific source lines. The logging and statistics information provided in the output were quite useful. I do think it would make more sense to have a live application where that was updated more similar to jvisualvm or other analysis tools. The output being completely distinct from the configuration of the tool also feels awkward.

Program output

```
[nix-shell:~/Projects/CSE-590]$ docker run -v $(pwd):/workdir  -u $UID:
    $GID   sosylab/cpachecker -predicateAnalysis -spec Property1a.spc
    tcas.i
Using the following resource limits: CPU-time limit of 900s (
    ResourceLimitChecker.fromConfiguration, INFO)

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13)
    started (CPAchecker.run, INFO)
```

Question assigned to the following page: 3

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May
     31 2023 12:38:06, gmp 6.2.0, 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".

[nix-shell:~/Projects/CSE-590]$ docker run -v $(pwd):/workdir  -u $UID:
    $GID   sosylab/cpachecker -predicateAnalysis -spec Property1b.spc
    tcas.i
Using the following resource limits: CPU-time limit of 900s (
    ResourceLimitChecker.fromConfiguration, INFO)

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13)
    started (CPAchecker.run, INFO)

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May
     31 2023 12:38:06, gmp 6.2.0, 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".
[nix-shell:~/Projects/CSE-590]$ docker run -v $(pwd):/workdir  -u $UID:
```

Question assigned to the following page: 3

```
    $GID   sosylab/cpachecker -predicateAnalysis -spec Property2b.spc
    tcas.i
Using the following resource limits: CPU-time limit of 900s (
    ResourceLimitChecker.fromConfiguration, INFO)

CPAchecker 4.0 / predicateAnalysis (OpenJDK 64-Bit Server VM 17.0.13)
    started (CPAchecker.run, INFO)

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

Using predicate analysis with MathSAT5 version 5.6.10 (9293adc746be) (May
     31 2023 12:38:06, gmp 6.2.0, 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".
```