Exercise 0F-2

Let A be the set of functions from X to $\mathcal{P}(Y)$ and let $B = \mathcal{P}(X \times Y)$. We will demonstrate a correspondence between A and B by exhibiting a function $f : B \to A$ and showing that it is both injective and surjective.

We define the function as follows. For any $S \in B = \mathcal{P}(X \times Y)$, define the function

$$\begin{aligned} g : X &\to \mathcal{P}(Y) \\ x &\mapsto \{y : (x, y) \in S\}. \end{aligned}$$

(Note that this function is well-defined because even if there is some $x \in X$ that never appears in S, that is, $\nexists\, y \in Y$ such that $(x, y) \in S$, $g(x)$ is simply vacuously defined as the empty set.) Put $f(S) = g$.

First we claim that f is injective. Let $S, T \in \mathcal{P}(X \times Y)$ be such that $f(S) = f(T)$, that is, $\forall\, x \in X$, $f(S)(x) = f(T)(x)$. It suffices to show that $S = T$. We will show that $S \subseteq T$, then, by symmetry we will also have $T \subseteq S$. Let $(x, y) \in S$. By injectivity, we have $y \in f(S)(x) = f(T)(x)$ and, by definition of $f(T)(x)$ this means that $y \in \{y : (x, y) \in T\}$, which implies that $(x, y) \in T$. We have shown that an arbitrary element of S is also contained in T, hence $S \subseteq T$. As stated above, an identical argument where the symbols S and T are swapped shows that $T \subseteq S$ as well, so we have $S = T$. We conclude that f is injective because S and T were chosen arbitrarily as collisions of f.

Next we claim that f is surjective. Let $g \in A$ be arbitrary, that is, g is some function $X \to \mathcal{P}(Y)$. It suffices to show that there is some $S \in B = \mathcal{P}(X \times Y)$ such that $f(S) = g$. Let $S = \{(x, y) : x \in X, y \in g(x)\}$. With S defined this way, for any $x \in X$ we have

$$\begin{aligned} f(S)(x) &= \{y : (x, y) \in S\} && \text{Expand definition of } f \\ &= \{y : (x, y) \in \{(x', y') : x' \in X, y' \in g(x')\}\} && \text{Substitute definition of } S \\ &= \{y : y \in g(x)\} && \text{Simplify redundant variables} \\ &= g(x). \end{aligned}$$

Thus we have that $f(S) = g$ and we conclude that every element in $B = \mathcal{P}(X \times Y)$ has a preimage under f.

We have shown that f is both surjective and injective. This implies that f is bijective as desired.

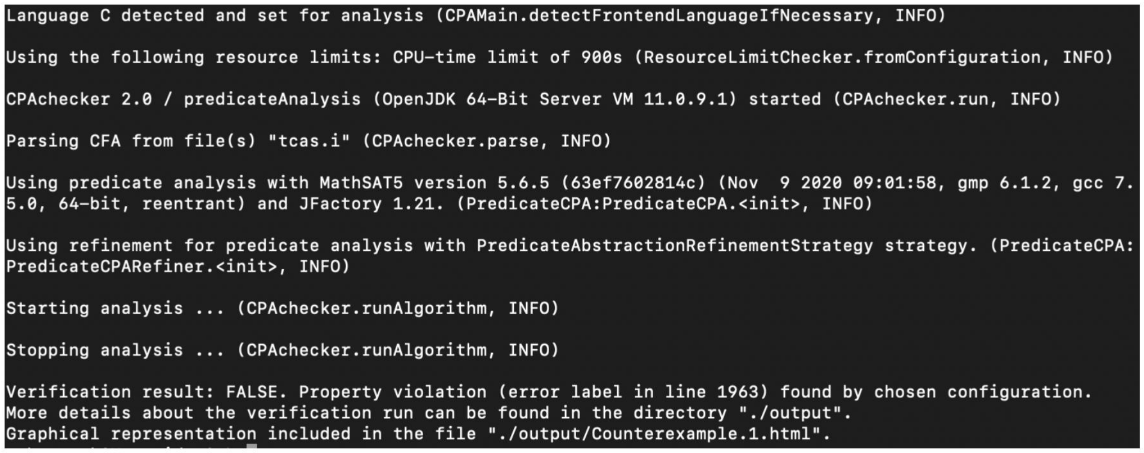
Peer Review ID: 62072167 — enter this when you fill out your peer evaluation via gradescope

Exercise 0F-3

When you run CPAChecker using the commands listed in the homework, it performs counterexample guided abstraction refinement to attempt to determine if the label specified in the `.spc` file that was passed in as an argument is reachable in program. While I'm not sure what `Property1a` corresponds to at a deep level, at a surface level, `Property1a` seems to check whether the variables `Up_Separation` and `Down_Separation` are below and above a certain threshold, and throws an error if they aren't. When run on `Property1a`, CPAChecker outputs that the property is false, which indicates that there is an execution path that makes it possible for this condition to not be met. CPAChecker then gives an example of such a path in the `output/` directory.

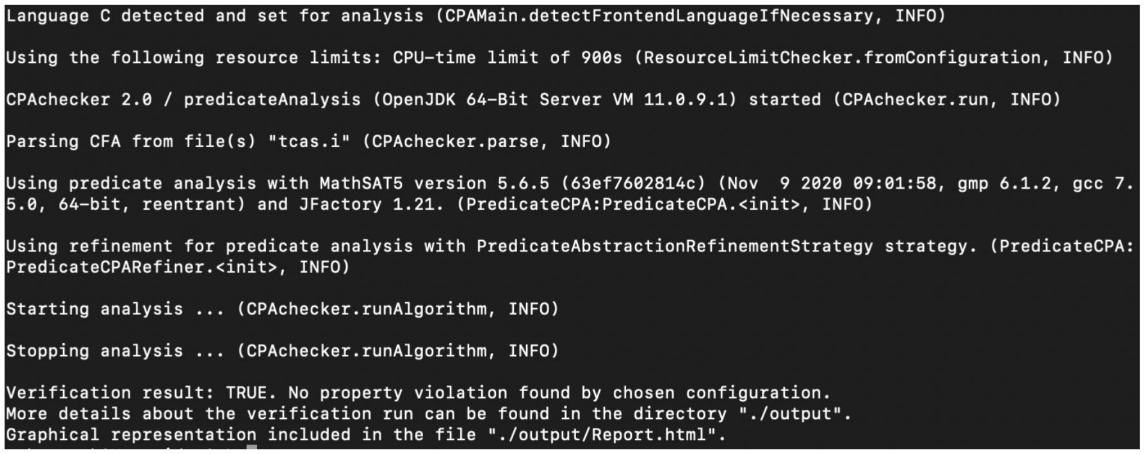
There would be a number of threats to validity if one were to use `tcas.i` as the only test case for CPAChecker. While it is inherently problematic to use only a single test suite to validate a program such as CPAChecker, there are other factors that make `tcas.i` a poor choice as a sole test suite, in particular. First, `tcas.i` is both too large for the CPAChecker results to be validated by hand (e.g. in the case where CPAChecker claims that the property is correct) and too small to be considered a comprehensive test on real-world applications. Another problem is that `tcas.i` has no loops so it wouldn't be useful for testing the tool's ability to identify loop invariants. Finally, `tcas.i` makes very little use of pointers and pointer arithmetic, so it also cannot be used to gauge the validity of the logical memory assumption. However, I do think that `tcas.i` is a useful test suite despite its shortcomings as a *singular* test suite.

I found the tool surprisingly easy to use. The result of a given validation check is presented plainly and is easy to read, and the graphical reporting tools it generates give the user an easy way to do in-depth analysis without being entirely overwhelming. I have no doubt that if I had any familiarity with the `tcas` program that I would find it remarkably easy to browse the report and make meaningful use of the presented counterexamples.



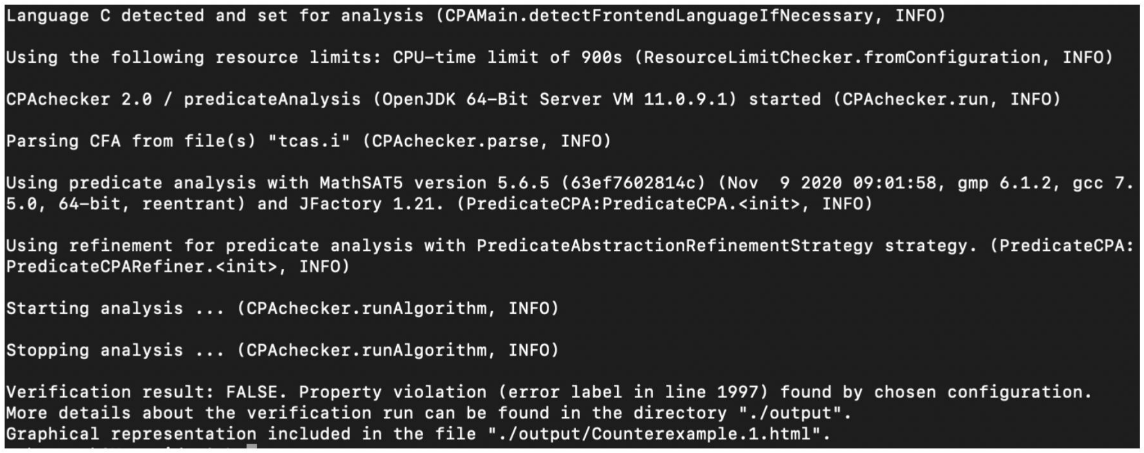
```
Language C detected and set for analysis (CPAMain.detectFrontendLanguageIfNecessary, INFO)

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".
```

Figure 1: Output of running CPAChecker on Property1a



```
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".
```

Figure 2: Output of running CPAChecker on Property1b



```
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".
```

Figure 3: Output of running CPAChecker on Property2b

1 HW0

- 0 pts Correct