Exercise 0F-2. Set Theory [5 points]. This answer should appear after the first page of your submission and may be shared during class peer review.

This exercise is meant to help you refresh your knowledge of set theory and functions. Let X and Y be sets. Let $\mathcal{P}(X)$ denote the powerset of X (the set of all subsets of X). There is a 1-1 correspondence (i.e., a bijection) bewteen the sets A and B, where $A = X \to \mathcal{P}(Y)$ and $B = \mathcal{P}(X \times Y)$. Note that A is a set of functions and B is a (or can be viewed as a) set of relations. This correspondence will allow us to use functional notation for certain sets in class. This is Exercise 1.4 from page 8 of the Winskel textbook.

Demonstrate the correspondence between A and B by presenting an appropriate function and proving that it is a bijection. For example, you might construct a function $f : B \to A$ and prove that f is an injection and a surjection.

Proof: To show there is a one-to-one correspondence between A and B, we will construct two functions $f : A \to B$ and $g : B \to A$ and show that they are inverses of each other.

We denote the function $f(a) = \{(x, y)|y \in a\}$ for $a \in A$, and the function $g(b) = \{y|(x, y) \in b\}$ for $b \in B$.

To show that f and g are inverses of each other, we can show that $g(f(a)) = a$ for $a \in A$. We have $g(f(a)) = g(\{(x, y)|y \in a\}) = \{y|(x, y) \in \{(x, y)|y \in a\}\}$, applying the definition of the function g. We observe that the right hand side is essentially the set $a(x)$ itself. Therefore, we have $g(f(a)) = a$. Thus, f and g are inverses of each other, which proves that the function $g : B \to A$ is invertible. So, there is a one-to-one correspondence between A and B. QED.

Exercise 0F-3. Model Checking [10 points]. This answer should appear after the first page of your submission and may be shared during class peer review.

Download the CPAChecker software model-checking tool using the instructions on the homework webpage. Read through enough of the manual to run the tool on the `tcas.i` testcase provided on the homework webpage. Check the two properties given. For each command, copy or screenshot the last ten non-empty lines of output from CPAChecker and include them as part of your answer to this question.

It is your responsibility to find a machine on which CPAChecker works properly (but feel free to check the class forum if you are getting stuck).

Hint: if your output when checking `Property1a` does not indicate something like "No property violation found by chosen configuration" then you have not set things up correctly.

What is going on when you run CPAChecker using the commands listed? In at most three paragraphs, summarize your experience with the CPAChecker tool. What does `Property1a` mean? Is `tcas.i` a reasonable test suite? What has been proved? Did you find CPAChecker to be a usable tool? You may find the graphical reporting option of CPAChecker to be helpful here. For full credit, do not restate my lecture on counter-example guided abstraction refinement; instead, discuss your thoughts and experience using this tool. Focus on threats to validity (e.g., imagine that you were writing a paper and using this as an experiment) over usability.

Both your ideas and also the clarity with which they are expressed (i.e., your English prose) matter. A reader should be able to identify your main claim, the arguments you are making, and your conclusion.

1. Property1a 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 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 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. (Predicate-
CPA: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 cho-
sen configuration. More details about the verification run can be found in the directory
”./output”. Graphical representation included in the file ”./output/Counterexample.1.html”.
```

When executing the commands listed in the homework, CPAchecker takes in the file specified by -spec as specification, which searches the source file, tcas.i, for certain labels and assertions identified in the spec file, as well as any paths leading to the error state. In this case, `Property1a`, `Property1b`, and `Property2b` are the specification file, which are used to search for certain labels and assertions in the source code of the input source file. More specifically, `Property1a` is checking if, in the source program, there is a case where the variable Up_Separation is greater than or equal to thresh, and Down_Separation is less than thresh. If such a case exists, it leads to error state and thus an error would be identified, whose location would also be shown.

`tcas.i` seems like a reasonable test suite, because it is fairly long and complex. Moreover, it is a simplified implementation of a traffic collision avoidance system, indicating that it has real-life meaning and significance. Furthermore, from all the different properties in `tcas.i`, it seems like the test suite covers many different possibilities with Up_Separation, Down_Separation, and thresh variables.

It was proven that the source C program `tcas.i` is correct given the specificaiton file `Property1b`, but is incorrect and leads to error state given specification files `Property1a` and `Property2b`. CPAChecker indeed seems like a very useful tool to check for correctness for C program files, because, given the specificaiton files, it is able to identify and locate the errors in the source file, if there is any. Moreover, given the reasonable test suite `tcas.i`, CPAChecker did perform well and identifies error states as specified. Therefore, I found CPAChecker to be a usable tool.

Submission. Turn in your assignment as a single PDF document via the `gradescope` website. Your name and Michigan email address must appear on the first page of your PDF submission but may not appear anywhere else.

Peer Review ID: 62061750 — enter this when you fill out your peer evaluation via gradescope

1 HW0

- **- 0 pts** Correct

Peer Review ID: 62061750 — enter this when you fill out your peer evaluation via gradescope