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.

Answer

To show that there is a 1-1 correspondence between sets A and B, we define a function $f : B \to A$ and show that f is an injection and a surjection, thereby showing that f is a bijection. Let us define a function f that takes in input $b \in B$ and produces output $a \in A$. Input b is a relation, represented by a set of (i, j) pairs where $i \in X$ and $j \in Y$. Output a is a function, represented as an ordered set $Z = \{z_1, z_2, ..., z_n\}$, where $n = |X|$ and every $z_i \in \mathcal{P}(Y)$.

Based on the definition of set B as described in the problem, we know that every $b \in B$ is a set of zero or more (i, j) pairs, where $i \in X$ and $j \in Y$. In the following paragraph, let the variable *input* be used to represent b, or a set of zero or more (i, j) pairs.

Let us define function f as follows: $f(input) = \{g_1(input), g_2(input), g_3(input), ..., g_n(input)\}$ where $n = |X|$. Additionally, let us define function g as follows: $g_k(input) = \{j | i = x_k\}$, where x_k means the kth element of X. For example, the set $g_3(input)$ is equal to the set of all j values such that the pair (x_k, j) exists in *input*.

First, we show that f is an injection. Let $b \in B$ and $b' \in B$, and suppose that $f(b) = f(b')$. We will show that $b = b'$. We know that $f(b)$ and $f(b')$ are defined as an ordered set of sets, and that in both b and b', there are $n = |A|$ sets within the single outer set. For each element e within the kth inner set, there is exactly one (i, j) pair that could have resulted in that element. If the element e were different, than the j within the pair would be different. If the element e were not at the kth inner set, the i within the pair would be different. Because each particular element in the inner set can be generated from one (i, j) pair, only one unique input b can map to a single output a. This means that b and b' must be identical, which means that f is an injection.

Second, we show that f is a surjection. To do so, we will show that for every $a \in A$ there is some $b \in B$ such that $f(b) = a$. We know that a is an ordered set of sets of length

$|A|$, and that each of the inner sets is an element of $P(Y)$. Because of this, we can translate a into the format of b by the following method. For the ith inner set in a, generate a pair (x, y_k) for every x in that inner set, where y_k is the kth element of Y. We know that the resulting set of pairs is an element of B, because by the definition of this method, the first item in each pair is an element of X and the second item in each pair is an element of Y. This means that the resulting set is an element of $P(X \times Y)$. Therefore, f is a surjection.

Because f is both injective and surjective, it is also bijective. In other words, there is a 1-1 correspondence between the sets 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.

Answer

Output for Property1a - Violated

```
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) 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:Predica
teCPARefiner.<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".
```

Output for Property1b - Not Violated

```
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) 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:Predica
teCPARefiner.<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".
```

Output for Property2b - Violated

```
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) 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:Predica
teCPARefiner.<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".
```

In my experience, the CPAChecker has been an easy-to-use tool for simple property verification. As specified in the commands used to run the checker, we perform predicate analysis. Additionally, the checker performs reachability analysis to see if the abstracted program is able to reach an error state. If so, the CPAChecker will then generate a counterexample report to demonstrate to the user how the checker came to generate a violation of the property. However, it is important to remember that this violation was found on the abstracted version of the program rather than the original program itself.

Based on the analysis done by the CPAChecker, we know that there could possibly be a violation of `Property1b`. We can better understand how the program described by `tcas.i` was found to be in violation of `Property1b` by examining the counterexample provided within the `output/` directory. The counterexample demonstrates a violation of `Property1b` on the program's abstraction, which does not necessarily entail that the property is violated by the original program. This is due to the fact that the analysis performed by the CPAChecker is done on the original program's abstraction, which in this case is `tcas.i`. The CPAChecker did not find a violation of either `Property1a` and `Property 2b`, which provides a guarantee that a violation of either property is not possible in the original program, in addition to the abstracted version. In short, the CPAChecker proves that there is not a violation of `Property1a` or `Property2b`, but we cannot say with absolute certainty that `Property1b` is never violated.

To provide further context, a violation of `Property1a` means the program was able to reach some state where `UpSeparation` was greater than or equal to some threshold and `DownSeparation` was less than that threshold. The other two properties also compared these two program variables (`UpSeparation` and `DownSeparation`) to each other and/or some threshold. For this reason, it might be the case that simply applying the CPAChecker on `tcas.i` and the properties described towards the end of `tcas.i` might not be enough to

demonstrate or, perhaps more importantly, test the full capabilities of the checker. Though `tcas.i` appears to make use of malloc operations, threads, and locking/unlocking, the properties defined towards the end of the file appear to focus strictly on doing simple checks on the results of simple arithmetic, further suggesting that `tcas.i` might not rigorously validate the CPAChecker. To summarize, it appears that `tcas.i` might not be the most reasonable test suite for the purposes of examining the full capabilities of the CPAChecker because of the simplicity of `tcas.i` as a benchmark and the lack of variety in the properties that are evaluated.

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.

1 HW0

- **0 pts** Correct