Exercise 0F-2. Set Theory [5 points]. 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.

Solution:
Consider the following function:
$f(b) = f_s : X \to P(Y)$ such that $b \in B$, $f_b(x) = \{y|(x, y) \in b\}$, and $x \in X$.

To prove the above function is a bijection we will show that it is one-to-one and onto.

One-to-one: We will assume that the function f is not one-to-one, which means should be able to find two distinct inputs b and b' that map to the same output. This would mean that $f(b) = f(b')$.

$f(b) = f(b')$
$f_b = f'_b$
$f_b(x) = f'_b(x)$ for all $x \in X$
$\{y|(x, y) \in b\} = \{y|(x, y) \in b'\}$

Let $x \in X$, $y \in Y$, and $(x', y') \in b$
$(x', y') \in b = y' \in \{y|(x', y) \in b\} = y' \in \{y|(x', y) \in b'\} = (x', y') \in b'$

Therefore, we can conclude that if $(x', y') \in b$ then $(x', y') \in b'$.
WLOG, if $(x', y') \in b'$ then $(x', y') \in b$.

Therefore we can show that $b = b'$. Our initial condition however was that b and b' are distinct. Since we have arrived at a contradiction, our initial assumption must be false, and so we can state that this function is one-to-one.

Onto: For some arbitrary output $a \in A$, we will provide an input $b \in B$ that maps to a.

Let $x \in X$ and $b = \{(x, y)|y \in f(x)\}$.
$f(b) = f_b(x) = \{y|(x, y) \in b\} = \{y|(x, y) \in \{(x, y)|y \in f(x)\}\} = f(x) = a$

Therefor we have shown that the function is onto. Since the function is both one-to-one and onto, it is bijective.

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 three 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.

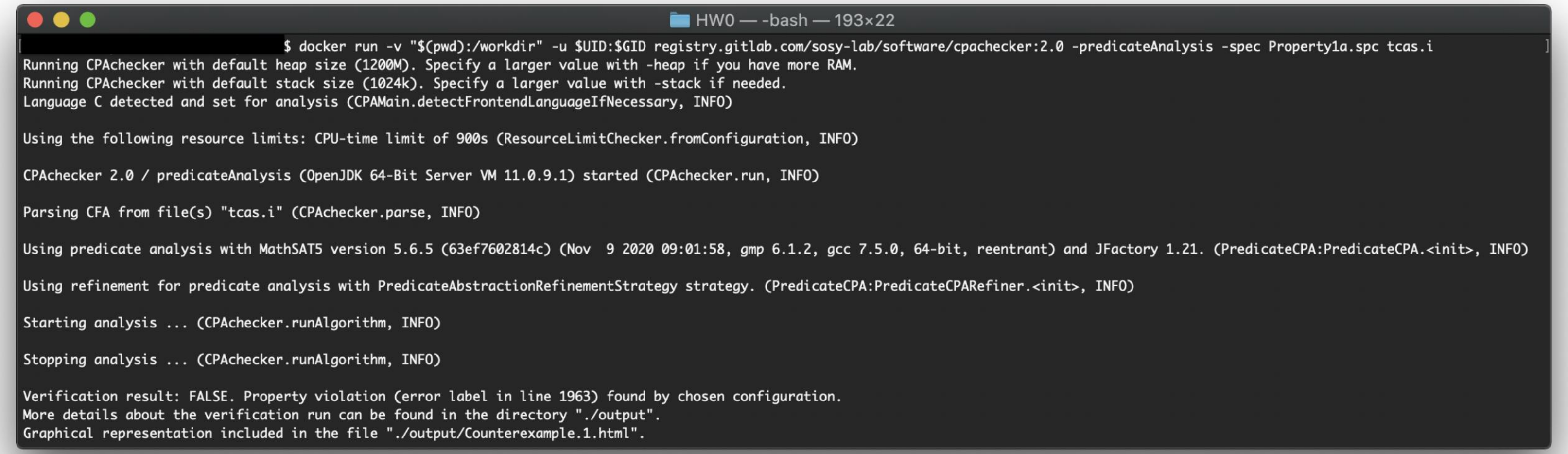
It is your responsibility to find a machine on which CPAChecker works properly (but feel free to check the class forum if you are getting stuck).

Hint: CPAChecker 2.0 should find a violation for `Property1a`, verify that `Property1b` is safe, and find a violation for `Property2b`. If your output does not match that and you are using version 2.0 then you may not 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.

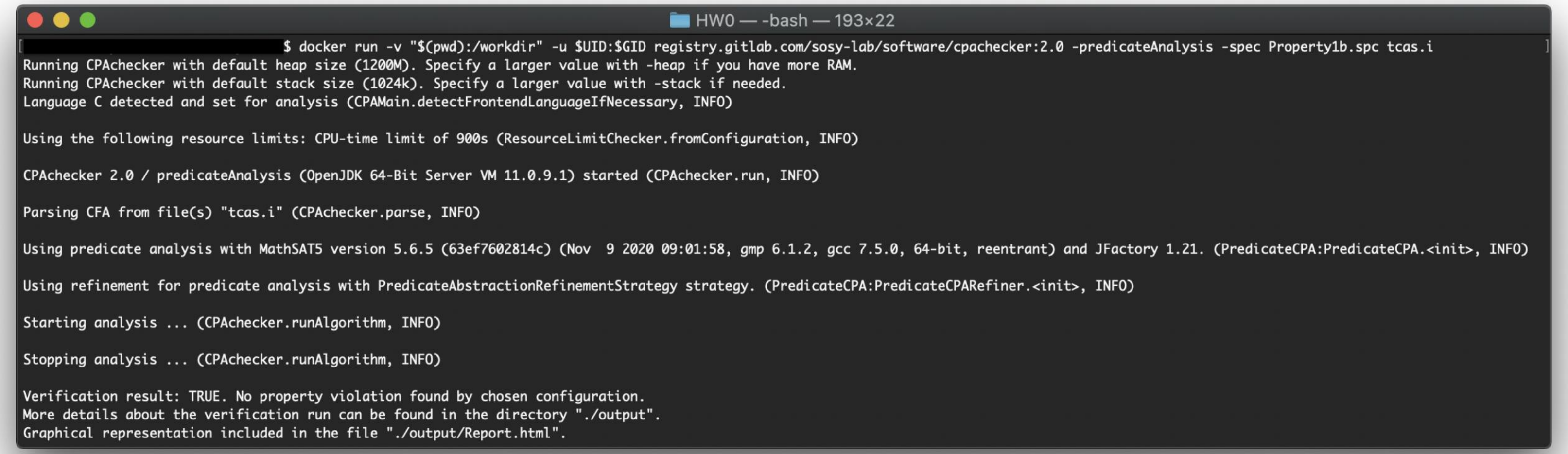
Both your ideas and also the clarity with which they are expressed (i.e., your English prose) matter. A reader should be able to identify your main claim, the arguments you are making, and your conclusion.

Output for first command:



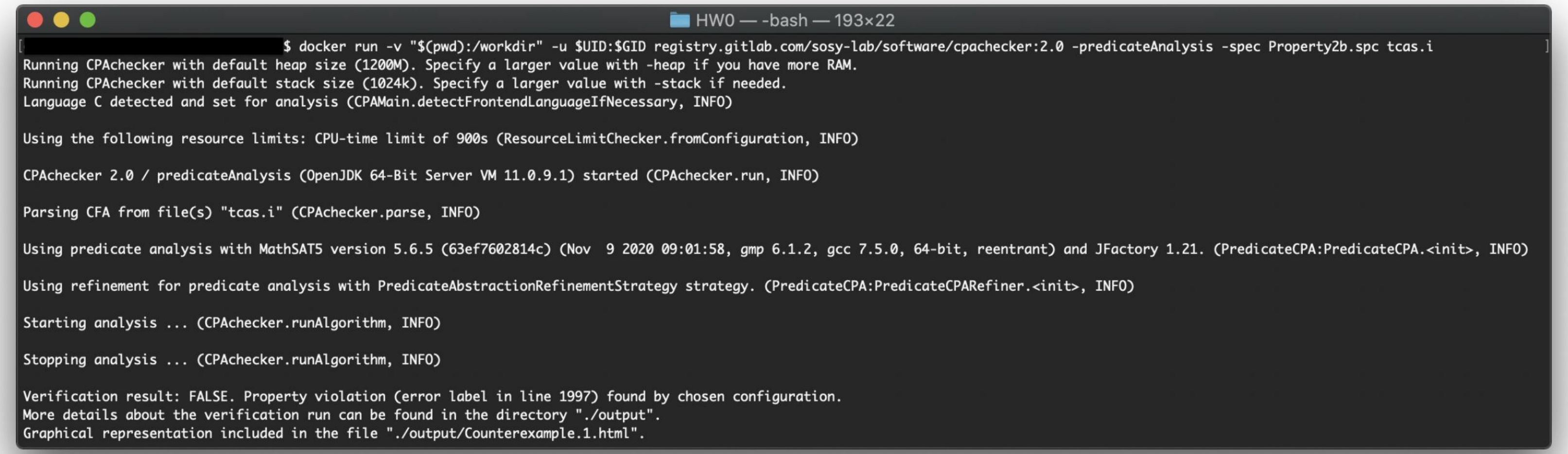
```
[redacted] $ docker run -v "$(pwd):/workdir" -u $UID:$GID registry.gitlab.com/sosy-lab/software/cpachecker:2.0 -predicateAnalysis -spec Property1a.spc tcas.i
Running CPAchecker with default heap size (1200M). Specify a larger value with -heap if you have more RAM.
Running CPAchecker with default 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 11.0.9.1) 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".
```

Output for second command:



```
[redacted] $ docker run -v "$(pwd):/workdir" -u $UID:$GID registry.gitlab.com/sosy-lab/software/cpachecker:2.0 -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 11.0.9.1) 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".
```

Output for third command:



HW0 — -bash — 193×22

```
[[redacted]$ docker run -v "$(pwd):/workdir" -u $UID:$GID registry.gitlab.com/sosy-lab/software/cpachecker:2.0 -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 11.0.9.1) 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".
```

Running CPAChecker with the listed commands causes the CPAChecker program to analyze the code using whatever properties are specified on the command line as command line arguments. The CPAChecker program took tcas.i as an input and produced a flow chart for control. Using this control chart it can check for certain areas of the code, specifically the error portions, and see if the model it has created allows for code to reach those error portions. Property1a itself refers to whether or not the control flow graph allows for a viable path to an error part of code with the Property1a label. The tcas.i test suite itself seemed to provide decent as it tests edge cases using specific boolean variables to check. In this case, CPAChecker found violations for Property1a and Property2b but not for Property1b. This means that we can be confident that the real program model never reaches an error state for Property1b but could reach errors for Property1a and Property2b.

I found CPAChecker to be a tool with many setup requiremnts and issues, but once the setup was done it was very easy to run on programs. I was running this on a Mac which meant I had to get the right version of Java running as well as run CPAChecker using Docker to get it to work. Once running, the output provided in the command line was brief yet informative and easy to read. The program also outputs the control flow graphs in an html file which is automatically saved. These graphs are highly detailed and have a lot of information but it was hard to truly understand and grasp what all the different states meant. The html output is also overwritten after every run which is a mild inconvenience.

I feel that CPAChecker is actually a highly useful tool. As a model checker it does everything it says it does and everything expected from a model checker. Any threats to validity in my opinion do not stem from CPAChecker specifically but rather from the idea of having model checkers that we accept may not always produce correct information about our programs. Reaching error states in the model are a good indication that our program might be faulty but this is not 100% accurate. Additionally, CPAChecker works well with

C programs only, but many programs and code bases in the modern world use different languages that might not even be object oriented. In those situations CPAChecker may not be useful and we would need to find a model checker for all the different languages which may not be possible yet.

1 HW0

- 0 pts Correct