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 of Exercise 0F-2 for this problem, I represent the sets A and B using matrix format as a preparatory work. We use $\mid X \mid$ and $\mid Y \mid$ to seperately represent the number of elements in sets X and Y.

For the set A, we use a $\mid X \mid \times \mid Y \mid$ binary matrix to represent the elements in set A. Each row represent an element in X without repetition, as a result, there are $\mid X \mid$ rows. For each row, if the row's corresponding element in X is x, the $\mid Y \mid$ columns means the binary representation of the corresponding powerset, naming it as p_x. If the value of (i,j) position is 1, it means that for x_i, the corresponding powerset of Y has the element y_j. It's reasonable to claim that all possibilities of the values of the $\mid X \mid \times \mid Y \mid$ binary matrix could find the one-to-one corresponding function in set A.

Similarly, we still use a $\mid X \mid \times \mid Y \mid$ binary matrix to represent the set B. For (i,j) position, if the value of this position in the matrix is 1, it means that in this powerset element of set B, the relation (x_i, y_j) is selected. If the value of this position in the matrix is 0, it means that in this powerset element of set B, the relation (x_i, y_j) is not selected. As we can easily tell, there's also a bijection from the $\mid X \mid \times \mid Y \mid$ binary matrix to the set B.

In the end, we compose the bijection function from set A to the binary matrix with the function from the binary matrix to set B. As a result, we could get a bijection function from set A to set B.

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.

Commands' Results: trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property1a.spc hw0/tcas.i

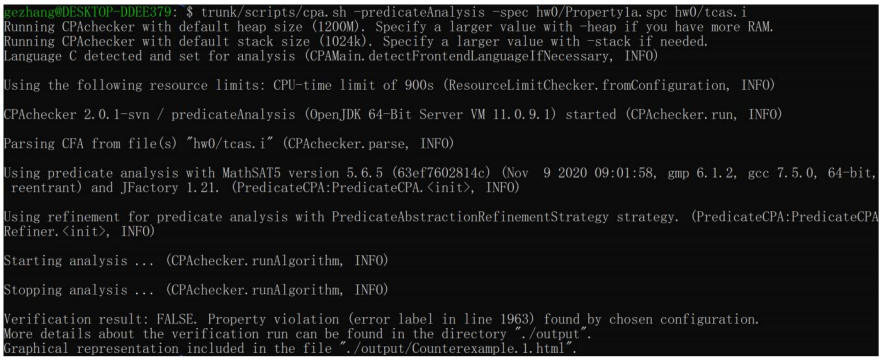

Figure 1: result of command : trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property1a.spc hw0/tcas.i

trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property1b.spc hw0/tcas.i

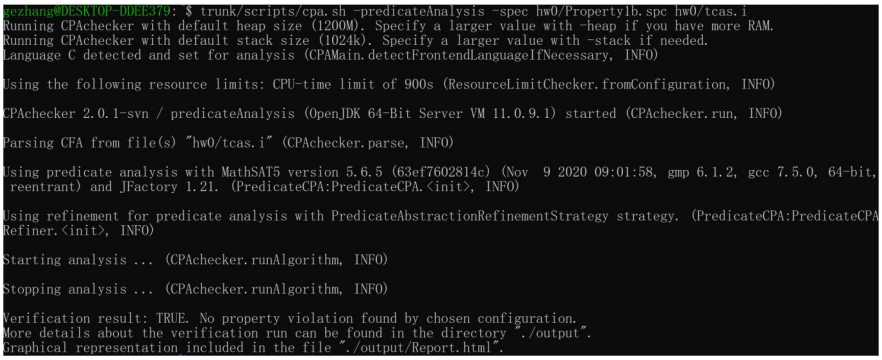

Figure 2: result of command : trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property1b.spc hw0/tcas.i

trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property2b.spc hw0/tcas.i

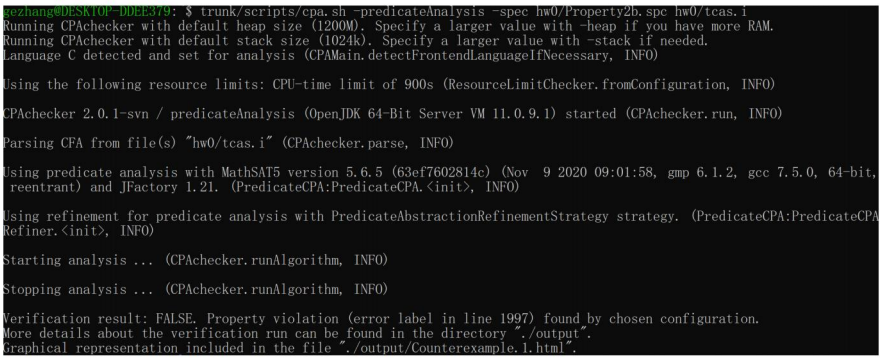

Figure 3: result of command : trunk/scripts/cpa.sh -predicateAnalysis -spec hw0/Property2b.spc hw0/tcas.i

Then I propose my using experience of CPAChecker. Property1a is actually a function containing specific reason of causing error and the definition also works for Property1b and Property2b. Porperty1a.spc actually means that program matches the Property1a label in USEFIRST state and throw out an error then in the location. tcas.i is a reasonable test suite. Because it contains those functions in its program. And it's not too complicated for us the double-check the result. We can denote that property1a is reachable in line 1963. It's proved that Property1b will never cause the error. CPAChecker is definitely a very useful tool. I met some difficulties of installing it in Windows system. Finally, I use an Ubuntu 18.04 LTS virtual machine to tackle these problems and use winSCP to share needed documents between my windows system with the virtual machine. In my opinion, although its short conclusion which can be seen in the command interface is useful as well, its website report means much more. Because its website report points out the suspects leading to errors, we can analyze all the suspects based on real condition and decide whether it's worthy to debug it.(In real conditions, sometimes it may be too hard to debug and has very little possibility to happen or pimping harm. As a result, it may not be worthy to debug it in real conditions.)

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