AutoV2

-Introduction and Use

Xiaoyu Jin

Overview

The tool AutoV2 is mainly used to automatically verifying and validating scientific software. The tool originally came from a project to automatically verifying and validating a scientific software called Adaptive Hydraulics Modeling System (AdH).

The general idea is that the tool will take requirement specification files as input and run a procedure or a set of interconnected procedures to verify or to validate if these specifications violated by the results and running situations. To be more specific, there are four categories of specifications: input/output; computation; constraint; and manual mistake. Input/output specifications mean to restrict the input to predefined scope and check if the output always falls into the expected range. Computation specification is especially important for scientific software since scientific software involves a lot of mathematical calculations which include some special functions. For example, a log() function requires the input must be real positive number. By checking these computation specifications, we aim to make sure the program will not run into situation to violate the computation specification. Constraint specification can be explicit or implicit. For example, the array size parameter should be equal to the actual size of an array parameter. Another example is that the sum of an array's values should be 1 if the array represents a distribution percentage. The manual mistake specification comes from the mistake identified from the original manual of the software system to be verified and validated.

Installation Requirements

The tool is written in a python script which is easily portable to any operating systems which support python. Currently the tool can only work on python version 3.0 and higher versions. The use of this tool is easy, which is just to install python 3.0+ on computer and to run the python script named AutoV2.py.

AutoV2 Usage Instructions

The figure1 is a glance of the tool's user interface. There are two parts of the tool: the upper part of figure 1 is used to verify an individual procedure's correctness and robustness; the lower part of figure 1 is used to validate a set of interconnected procedures. A detailed step by step explanations will be given in the following section.

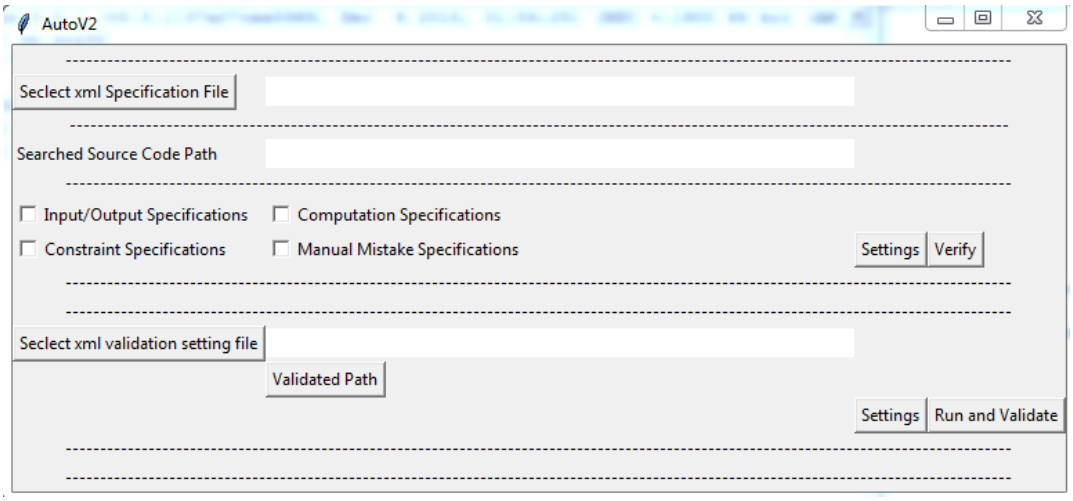

Figure 1. User interface of AutoV2 tool

Detailed steps for verification (the black box indicate the actions user need to perform):

Step1: click the button "Seclect xml Specification File" to select a specification file (e.g. 4-Procedure1.xml) for a procedure, then the tool will find corresponding procedure to be analyzed:

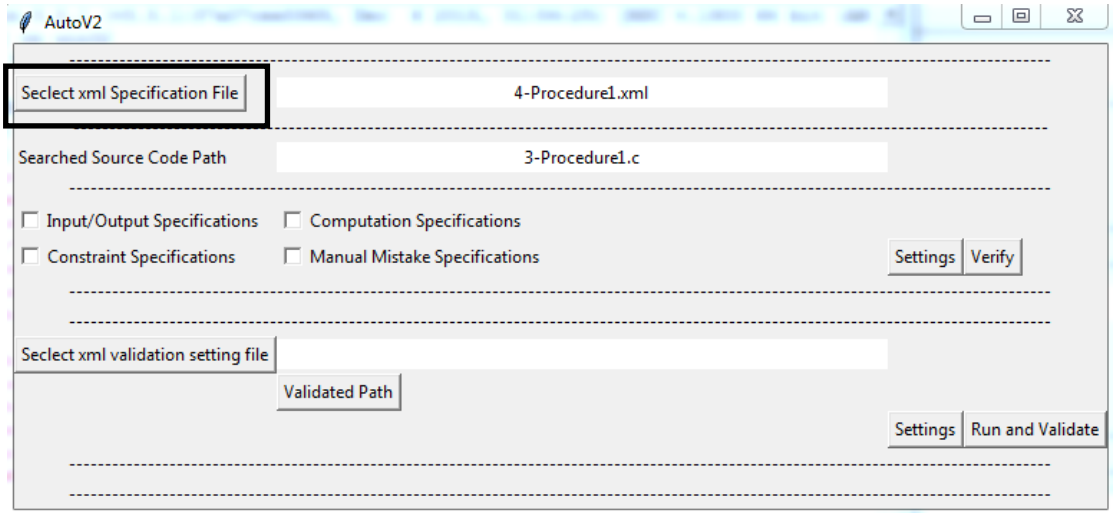

Figure 2. Select a specification file to verify a procedure

The specification file described all specifications for an individual procedure:

```
<SpecificationList>
    <procedurename> 3-Procedure1.c </procedurename>

    <InputOutput>
        <Specification>
            <Id>S1</Id>
            <Description> 5 > a > 0 </Description>
            <Type>Input</Type>
        </Specification>
```

Figure 3. An example of specification file

Step2: select the categories of verification need to be analyzed (e.g. Input/output and Computation):

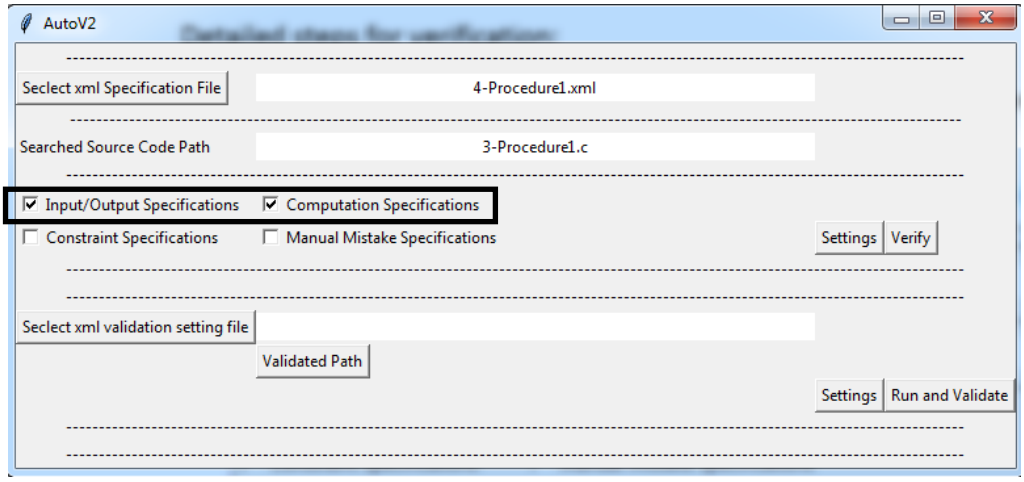

Figure 4. Select the categories of verification analysis

Step3: click the button "Verify" to perform the verification analysis and get the results:

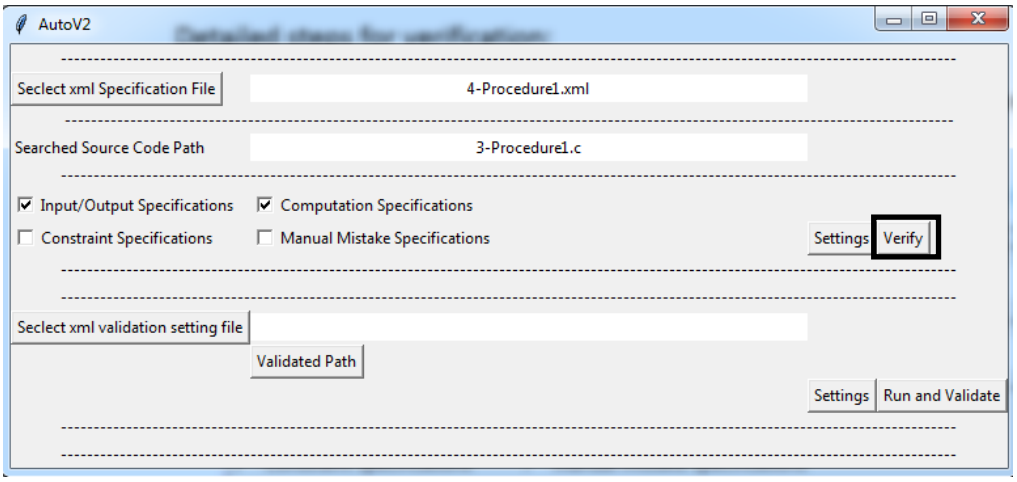

Figure 5. Perform verification analysis automatically

The results are displayed in a new window as shown in figure 6. Figure 6 shows an example of result in the computation category. The meaning of the result:

In source code Procedure1.c, there is a line of code: temp1 = -b / (2*a);

From this line of code, we add assertion code before this line: assert(2*a != 0);

Then the AutoV2 tool will run the procedure multiple times with different random input values;

The result shows that the assertion is verified which means in any situations 2*a will not be equal to 0. So this specification passed the verification test.

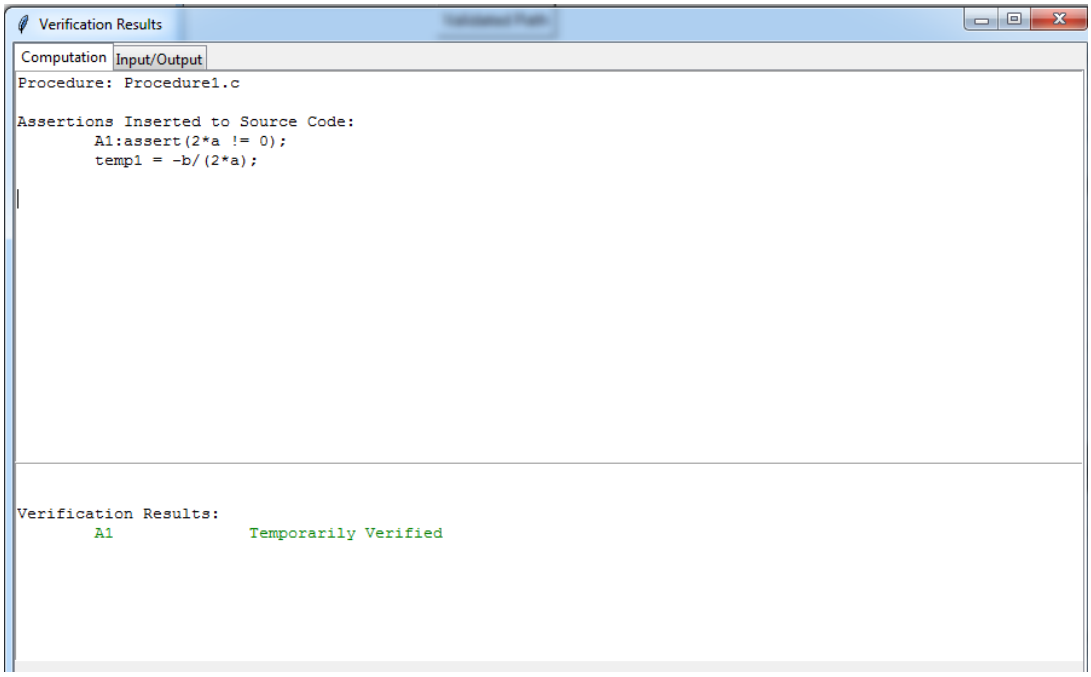

Figure 6. An example of verification result

Detailed steps for validation (the black box indicate the actions user need to perform):

Step1: click the button "Seclect xml validation setting file" to select a validation setting file (e.g. 5-Validation_setting.xml):

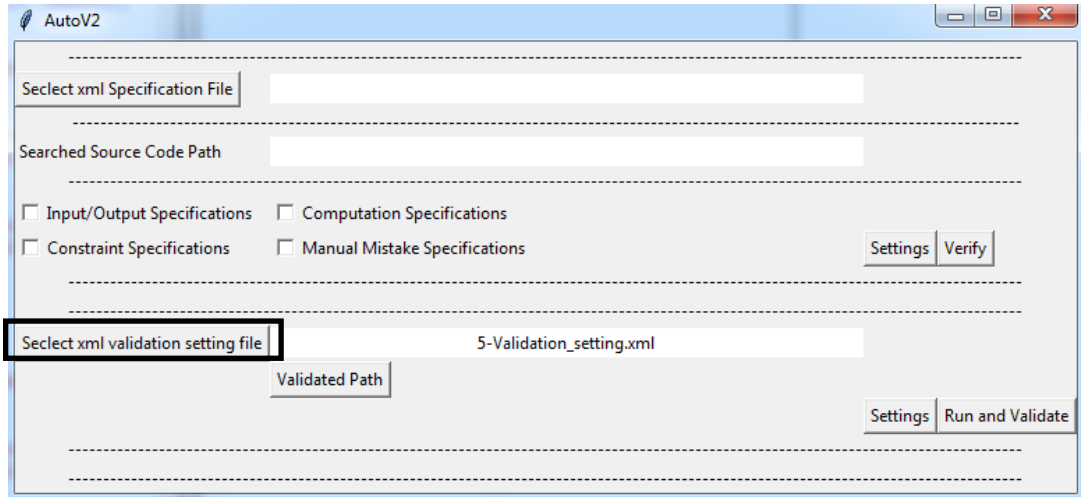

Figure 7. Select validation setting file

Example of validation setting file is shown in figure 8. The validation setting file includes two parts of information. First, it indicates which procedure this validation case start from. Second, it details the changed input specifications compared with verification specifications.

```
<ValidationSetting>
    <StartProcedure>3-Procedure1.c</StartProcedure>

    <Input>
        <Specification>
            <Id>S1</Id>
            <Name>a</Name>
            <Description>a = 3</Description>
        </Specification>
        <Specification>
            <Id>S2</Id>
            <Name>b</Name>
            <Description>b = 3 </Description>
        </Specification>
    </Input>

</ValidationSetting>
```

Figure 8. An example of validation setting file

Step2: click the button "Run and Validate" to perform validation analysis and display the results:

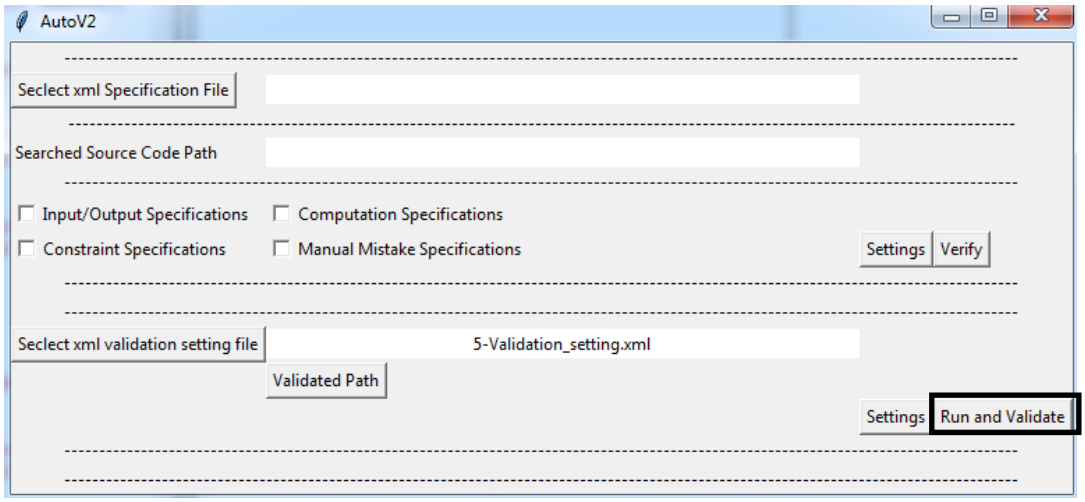

Figure 9. Perform automatic validation analysis

The results are composed of two parts. The first part is a displaying of the validation path of the set of interconnected procedures to be validated as shown in figure 10. Since we try to make our example as simple as possible, we only have two procedures. The result shows that Procedure1 calls Procedure2.

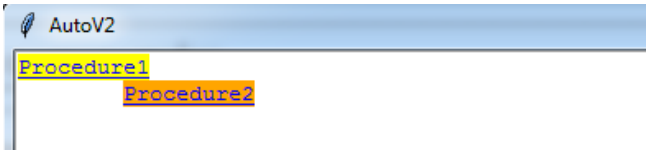

Figure 10. Perform automatic validation analysis

Step3: click a procedure (e.g. Procedure 2) to see the detailed validation results for a specific procedure:

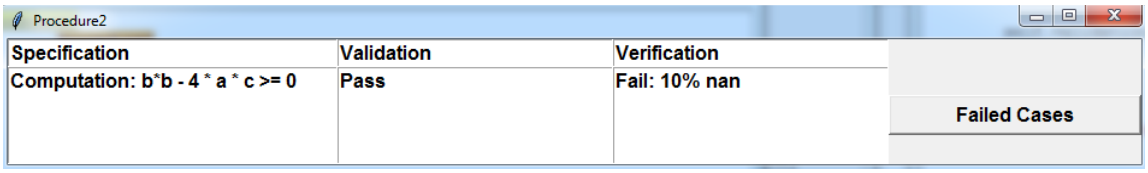

Figure 11. Detailed validation results for Procedure2

Step3: click the button "Failed Cases" to see the example cases of input values which failed the validation for this procedure:

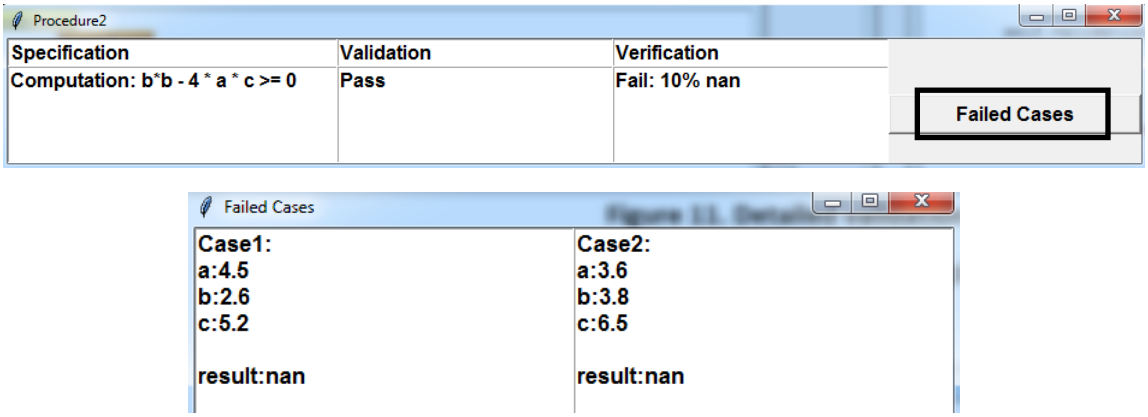

Figure 12. Failed validation input cases for Procedure2

Contact Information

- nan.niu@uc.edu