CS639A: Program Analysis, Verification And Testing
Users Online : 14

Assignment-6A: Programs verification using Abstract Interpretation

Due on : 20/11/2021 23:59

Instructions

- You are given an extra 10 minutes after due time to submit your assignment.
- However, please note that any submissions made after the due time are marked as late submissions.

Assignment-6A: Programs verification using Abstract Interpretation

Question:

Problem Statement

In this assignment, we will try to track for safe movements of Kachua. Kachua would move in accordance to the progra instructions, and eventually rests at the final position reached at the end of the program. If Kachua rests into an area ir magarmach's (crocodile's) region, the magarmach can eat it. Such a dangerous region is usually rectangular in shape ar can be described by x-coordinate interval and y-coordinate interval e.g, ($[x_1, x_2]$, $[y_1, y_2]$). Given a Turtle program, students need to verify that the Kachua never rests in the magarmach's region. You need to verify this using abstract interpretation with the interval domain.

Details

We have implemented work-list algorithm in Kachua-v4.0. Work-list algorithm calls different modules (i.e., TransferFunction, Initialize, SelectNext, isChanged, etc.). Students need to implement the incomplete functions preser in the **`Submission/submissionAI.py`**. The entry point of this assignment will be **`AnalyzeUsingAI()`** which studen need to implement. It will take the intermediate representation as input and emit the result of the analysis.

The magarmach's region will be provided as a JSON file with the same name as the Turtle program. Eg. if the Turtle file "foo.tl", then the magarmach's region will be in "foo.json"; a sample JSON file is given below:

```
{
        "reg" : [
                [2, 5],
                [8, 9]
        ]
}
```

where (2,5) is the top-left corner and (8,9) is the bottom-right corner of the rectangle. The Turtle programs provided t you will only contain arithmetic operations + (plus) and - (minus), and all the relational operators. The rotations can on be in multiples of 90 degrees.

Note

Do not change the code snippet which is already present in the **`Submission/submissionAI.py`**.

Running abstract interpretation

```
python3.8 kachua.py -ai examples/test.tl
```

CS639A: Program Analysis, Verification And Testing
Users Online : 14

Deliverables

The source code of your implementation, **including the KachuaCore directory of the version you used** along with the Submission folder.
A brief report (less than 5-pages) describing your implementation, assumptions and limitations. (Understand the difference between the tool's limitation and a bug: any error or missing feature that is caught during evaluation is a bu unless it is listed under the "limitations" section of your tool. Please do include Kachua version used in the submission)
A set of test cases (at least 5) with the expected output. **(tests folder, KachuaCore, Submission, README)**

The quality of all the above would affect your marks. The quality of all the above would affect your marks.

Submission Format

Your submission **MUST** be in the following format:

The submission should be a **zip** file.
The zip file should be named as **assignment_"number"_"Roll-of-student"**.
Zip the content of the **source** as is and submit. **Don't refactor the base code or move the files around.** (KachuaCore Folder, Submission Folder, README)

Please note that your submission will **NOT** be graded if you do not follow the format. Furthermore, we will use the **Readme** file provided by you to build and run your code. Therefore, please make sure that the Readme is clear. We cannot grade your submission if we cannot run it on our system.

Some important comments

Before doing anything "extra" (which might fetch bonus marks), first, complete the basic expectations from your implementation.
Program analysis tools are expected to display their results in a user-friendly manner; a user would never like to use a tool that simply spits out a bunch of numbers. So, display the results from your tool suitably.
Discussion is healthy, copying is not. You are encouraged to discuss the assignments, but you must implement the assignments individually. If any two students are found with "similar" pieces of code, both of them will be failed (with n concern as to who was the source).


Screenshot from 2021-11-04 23-27-42.png

Due time is over.