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

Assignment 3: Kachua Movement Optimization

Submitted on 22/9/2021 22:51

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 3: Kachua Movement Optimization

Question:

Assignment 3: Kachua Movement Optimization

This assignment is to be implemented in **one** person team.

Objective

We want to perform dataflow analysis and generate an optimized program that makes fewer moves of Kachua while generating the same figure.

Consider the following program; the variables **:x**, **:z**, and **:w** are inputs to the program:

```
penup
:y1 = :x + 20
forward :y1
if (:y1 > 42) [
  :y2 = :z - 20
  forward :y2
] else [
  :y2 = :w - 20
  forward :y2
]
pendown
```

It can be optimized to:

```
penup
:y1 = :x + 20
~~forward :y1~~
if (:y1 > 42) [
  :y2 = :z - 20
  :t = :y1 + :y2
forward :~~y2~~ :t
] else [
  :y2 = :w - 20
```

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

It reduces the movements of the Turtle while having no change in the generated figure.

The aim of the assignment is to implement (intraprocedural) Kachua movement analysis for Turtle programs. Your implementation should identify all move statements that can be reduced, and generate an optimized program.

In particular, you will have to define the **optimize(ir)** function in submission.py.

Once your implementation is done, you can invoke Kachua as follows:

1) Optimize the program test.tl and generate an optimized IR (in binary form) called "optimized.kw".

```
./kachua.py -O example/test.tl
```

2) Load the optimized IR (in binary form) and run it.

```
./kachua.py -b -r ./optimized.kw
```

Kachua can be run on the original program as well:

```
./kachua.py -r example/test.tl
```

The optimized version should have fewer movements of the Kachua but till draw the same figure.

Minimum Kachua version to use: **v1.0**

Input

A program in Turtle.

Output

Transform the IR into an optimized IR. The output IR should be correct, i.e., you should be able to execute it and get the same output as the original program on all inputs.

Deliverables

The source code of your implementation. Only submission.py and any other files created by you must be submitted.
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.)
A set of test cases (at least 5) with the expected output.

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:

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

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).

Uploaded Files:

assignment_3_21111037.zip

Grades:

Marks: 20

Feedback: Could not run during evaluation and could not run in TA's system also.