PE Autumn 2017 – Statechart

August 8, 2017

1 Introduction

This is the project charter for the project elective for Harika.

2 Goals and Objective

To implement modules on:

1. Front end for Statechart
2. Define data structures for representation of abstract syntaxt tree (AST) and control flow graph (CFG)
3. Implement algorithm to derive CFG from AST
4. Implement data flow analysis algorithm to demonstrate the ability of the algorithm to detect specification bugs involving undefined variable errors on identified case studies
5. Prepare a ready-to-download package (code, documentation and case studies) to be uploaded in GitHub

3 Description

The project implementation work has to be carried out in close partnership with Karthika Venkatesan[1]. Her responsibility will be to

1. focus on the detailed design of all the algorithmic and architectural aspects of the tool

[1] karthika.venkatesan@iiitb.ac.in, +91 98443 34898

2. take lead to make ready the case studies on which the tool will be tested

3. track the progress closely and report any issue or concern on a timely basis.

The project involves the following stages:

3.1 Introduction

In this, the student will develop understanding of the problem, and take handover of any knowledge and resources already existing.

3.2 Front end

In this, the front end for Statechart has to be implemented. This involves a parser that can read a statechart written in a language form and generate a control flow graph for the same to be given as as the input to the data flow analysis algorithm.

3.3 Data structures

In this, two modules have to be implemented as follows:

3.3.1 Abstract syntax tree

In this part, the object oriented language library to represent the abstract syntax tree for Statechart representation has to be implemented.

3.3.2 Control flow graph

In this part, object oriented library to represent a control flow graph will be developed.

3.4 Algorithm to convert AST to CFG

In this part, the algorithm to convert the abstract syntax tree to the equivalent control flow graph will be implemented.

3.5 Data flow analysis

In this part, the data flow analysis algorithm (reaching definition) will be implemented to identify undefined variable bugs in the specifications.

3.6 Experiment

The proof of the concept will be demonstrated using homemade statechart specifications. The tool will be run on these specifications, and bugs – either naturally occuring or injected by design – will be identified automatically.

3.7 Downloadable package

The entire work done should culminate in a software package that can be downloaded and the experiments can be repeated at the push of a button by anyone.

4 Tentative Timeline

Timespan		**Deliverable**
Start	**End**	
Aug. 10, 2017	Aug. 15, 2017	Introduction
Aug. 15, 2017	Aug. 31, 2017	Front end + AST
Sep. 01, 2017	Sep. 15, 2017	AST to CFG
Sep. 15, 2017	Sep. 30, 2017	Data flow Analysis
Oct. 01, 2017	Oct. 15, 2017	Experiment
Oct. 15, 2017	Oct. 31, 2017	Package
Nov. 01, 2017	Nov. 15, 2017	Improvements
Nov. 15, 2017	Nov. 30, 2017	Buffer

5 Modus Operandi

1. There will at least one update meeting every week on a mutually convenient time slot, typically lasting for about half an hour.
2. These meetings will be used to give updates, plan for the subsequent week and discuss issues if any.
3. There will be more meetings if required.
4. Research student will keep a day-to-day tap on the progress and meet at least one more time a week on average with the project student.
5. Meeting cancellations, if any, must happen with prior notice.
6. Regular attendance of meeting is an important component of the project assessment.

7. All progress, thoughts, ideas discussed will be documented on a regular basis.