State Machines

Advanced Systems Programming (H) 2022-2023 – Laboratory Exercise 5

Dr Anna Lito Michala & Dr Colin Perkins, School of Computing Science, University of Glasgow

1 Introduction

The Advanced Systems Programming (H) course uses the Rust programming language (`https://rust-lang.org/`) to illustrate several topics in systems programming. However, as programmers you will need to develop critical thinking skills that challenge when an approach is suitable for the software being developed. To assist this learning, this exercise is introducing the concept of graphical tools for state machine implementation. **This is a formative exercise and is not assessed.**

State machines have been used for many years to help software developers create correct programs that will always display deterministic behaviour. As we have seen in SP(H) this is not always true though for C implementations where one can introduce undefined behaviour through the wrong implementation of memory management. In an attempt to enforce deterministic behaviour for C programs that implement state machines the community has developed graphical tools that enable software developers in designing the state machine and automatically producing correct and deterministic code skeletons. One example is the `Qt` state machine framework (you can read more here `https://doc.qt.io/qt-5/statemachine-api.html`).

In this session we will discuss the pros and cons of such frameworks versus strict languages in producing deterministic and memory management error free code.

The expected timeline for this laboratory exercise is that you complete the all the tasks in Section 2 by the end of the timetabled lab session on Monday 6 February 2023.

2 Formative Exercise

To improve your critical reasoning about programming languages such as C and Rust, complete the following exercises. **These exercises are not assessed, and you do not need to submit your solutions**.

1. Organise yourselves into groups of 5 or 6 (10 mins).
2. Use the link provided above to familiarise yourself with the `Qt` framework. Specifically focus on how the graphical interface forces the developer to think about all possible transitions. Also read about the produced code and what editing rights the software developer has to alter the produced code (15 mins - individual).
3. Discuss your findings with your group and come up with an answer to the questions "Can the developer bypass the enforced design by editing the code directly?" and "Would Rust improve the strictness of the messages passed and functions executed along each transition?".
4. Each group should present their main argument as a quick stand up presentation (10 mins).
5. The lab will end with an open discussion about the variety of language features and issues that the Qt framework brings (5 mins).

Discuss your findings with the lecturer or lab demonstrators to make sure you are working towards the task and allow everyone in the group to speak and contribute.