CST Part II Project – Progress Report

An observable OCaml, via C and `liballocs`

Cheng Sun <`cs799@cam.ac.uk`>

April 28, 2017

Project Supervisor: Stephen Kell

Director of Studies: John Fawcett

Project Overseers: Timothy Griffin & Pietro Lio

Progress

My project has been progressing well. All of the code I have written so far has been in OCaml, with the exception of portions of the stdlib that the generated C code has to link with. I have completed/made significant progress in the following:

- AST datatypes for the C language, representing statements and toplevel definitions.
- Translation module, translating OCaml lambda IR into "extended" C AST.

  This is now capable of translating basic types like ints, booleans, tuples; references; lists; functions and complete function application; polymorphic functions; linking against and accessing values in other modules.
- Fixup module, taking an "extended" C AST to a standard C AST, normalising invalid expressions that are created during translation.
- Emitcode module, transforming the C AST into a string representing the C code, and subsequently outputting this as a file.
- OCaml compiler driver changes: added a new flag `-target-liballocs` which enables my backend.
- Started a prototype standard library, initially supporting `Printf.printf`.
- Started writing my own corpus of self-contained OCaml files for testing purposes. These generally perform a mathematical computation of some sort, which can be verified/benchmarked.

I estimate that I am currently 1–2 weeks behind in my project, due to some personal time management issues last term (outlined below). However, whilst I have been working on it, my project has been progressing very quickly, and I am confident about being able to catch up with where I want to be by the end of this month. Hence, I do not need to make any changes to my project plan.

Difficulties

Time allocation during Michaelmas was more difficult than anticipated, and I ended up not starting my project in earnest until the term had ended. However, I managed to catch up rapidly, and the project is in far better shape now. As I am attending fewer courses in Lent, this term will allow more time for me to focus on project work.

It turned out that there was no perfect choice of Intermediate Representation in the compiler which immediately satisfied all of my needs. I ended up choosing to start with the so-called "lambda" IR, which is based on a (heavily augmented) untyped lambda calculus. This has been suitable for C translation purposes, as most constructs have a fairly direct mapping. However, I will need type information later in the project, which this IR does not provide. I'm working on extracting this information from typing environments instead.