CO600 Project: Jarlang
Personal Report

Andrew Johnson

apj8@kent.ac.uk

School of Computing

University of Kent

United Kingdom

Word Count: 1,500 (via texcount)

March 29, 2018

Abstract

This report reflects my personal experience as a member of the Jarlang project, and my assessment of its outcomes, successes and shortfalls. As detailed in the Technical Report Jarlang has been very successful, surpassing the pre-existing projects in functionality and performance. However, some mistakes were made and as I detail in this report they were mainly the result of our hubris.

1 Project appraisal

The Jarlang Project was a venture into unknown territory for us, the pre-existing work being either a composition of third-party modules (Tonpa, 2014) or intended for code written with JavaScript in mind (Guthrie, 2014). Neither of these fulfilled our goal of taking any given Erlang file and transpiling it to human-readable, valid, JavaScript. As such, and as these projects were not written in Erlang as we intended to do, we had no example with which to predict how the project would progress.

1.1 Starting out

Our first step was to divide exploratory work between the group, initially we were unsure how best to do this and several weeks of back-and-forth followed as we developed our understanding of the project.

Once we had a more complete understanding of the requirements of our transpiler we divided our work again. We decided Chris was to develop the process of creating JavaScript output from an Erlang program. I was to develop a parser for the Core Erlang AST, determining how Erlang functionality would relate to JavaScript syntax and calling Chris's module to implement the output. Nick was to examine the JavaScript source mapping and find a way to extract line information from the Core Erlang AST so that bugs in code produced by our transpiler could be debugged by the user.

1.2 Adjusting expectations

As the project progressed we began to understand how much work was required of us and accepted that some of our more ambitious goals were unrealistic. We had originally intended to demonstrate our project using an advanced clone of Pac-Man, but we realised that an ambitious example was out of our reach along with the source mapping that Nick was working on. It is my understanding that Nick has completed the required line numbering but we have not had time to integrate that.

1.3 Final state

Although our over-ambitious goals have not been met the value and accomplishment in our progress so far is considerable. The Jarlang transpiler is capable of producing legible, functional, JavaScript that conforms to the functionality of its source code. We have contributed a useful, unique, resource to the Erlang and JavaScript communities by making this project open source.

That said our project has some limitations and features that we have been unable to implement by the deadline. Despite my efforts I have not been able to remove all the bugs from `asttrans.erl`, as such we cannot guarantee that any Erlang source file will transpile correctly. We have also not been able to substitute the Erlang Standard Library in JavaScript in the time available, this is in part because the bugs remaining in the Jarlang transpiler prevent the transpilation of modules written in Erlang, but also because much of the Standard Library is not written in Erlang and would have to be implemented manually by us.

2 Lessons learnt

Although quite satisfied with the state of the project at this date, as expected we have made some mistakes in getting to this stage. The most prominent in my mind is the misapplication of our time during the first half of the project. I attempted to create files that used every possible language feature in an attempt to prepare tests in advance of development. While test driven development is a successful paradigm in most projects the experimental nature of Jarlang meant that we did not realize until we engaged in testing that the most useful tests would be the ones that require only one feature not yet implemented.

In addition to unnecessary test files we attempted to implement automatic testing, for which Nick was responsible. We've since realised the futility of designing an automatic testing program when the entry points of our project application were still uncertain and the process of calling Erlang programs from the command line is mostly undocumented.

In order to improve our work flow we sought a method of calling Erlang programs from the command line, unfortunately the lack of documentation in this area made the process quite difficult. Nick attempted to find a solution while considering automatic testing, and later I was able to call a single function without arguments but could not determine a means by which to pass the required arguments to the function. It was only half-way through the project when Chris dedicated a week to the problem that we had a solution. From this I learnt that a little persistence goes a long way, and `ErlPKG` has greatly improved our efficiency.

My primary contribution to the project has been the `asttrans.erl` module. As the point in Jarlang that bridges the Core Erlang AST and the JavaScript AST it recursively parses the Core Erlang AST, and conceptually it is easy to visualize this as a single function. However as I implemented one feature at a time that single function grew beyond simple understanding without my notice. Eventually Chris encouraged me to re-factor the module and the result was code that was easier to understand and error messages that were easier to read. I regret not re-factoring `asttrans.erl` sooner and would like to think I will be more inclined to re-factor my code in the future.

3 Evaluation

This report should be an assessment of the progress of the project, reflections on what you have learnt from undertaking it.

These reflections should include a critical appraisal of the project, indicating the rationale for any design/implementation decisions,

lessons learnt during the course of the project,

evaluation (with hindsight) of the product and the process of its production.

It should not be a repeat of other material delivered as part of the project.

It should contain at most 1500 words and should be submitted both as a paper document and electronically. The format of the individual report should be the same as the technical report.

References

Guthrie, G. (2014). Luvvie script github archive. `https://github.com/hypernumbers/LuvvieScript`.

Tonpa, N. (2014). Shen erlang javascript parse transform. `https://github.com/synrc/shen`.