Imperial College London

3RD Year Computing

Interim Report

Implementation of a New Web Language

Author:
William de Renzy-Martin

Supervisor:
Sergio Maffeis

February 21, 2014

1 Introduction

The applied π-calculus is an expressive formal language describing computations as communicating processes. Its primary use in both industry and academia is to model security protocols. These models being built, they can then be statically analysed either by hand, or automatically by using a tool such as ProVerif.

For the purposes of static analysis, models built using applied π-calculus have proven very useful. Applied π-calculus has been used to verify numerous security protocols, including but not limited to [RS13]:

- Email certification
- Privacy and verifiability in electronic voting
- Authorisation protocols in trusted computing
- Authentication protocols and key agreement

However, these models are limited by the fact that they cannot currently be executed directly, as there is no existing language implementation.

Without an implementation, any models built using the applied π-calculus cannot be used to actually demonstrate protocols they are modelling, and so those models can be very difficult to debug.

1.1 Objective

The aim of this project is to provide an implementation of the applied π-calculus such that one might be able to build a model of a web protocol and then execute it interoperably with existing implementations written in PHP, Javascript or any other web language. The resulting implementation will hopefully not only be a very powerful and concise language for reasoning about and implementing protocols, but a useful scripting tool for the web.

At the very least, we would like to be able to write something similar to:

```
in(a,M);
out(b,M);
```

Compile it and have it execute successfully; receiving a message in on channel a, and sending the same message out again on channel b. However, ideally we would like to write something like the following:

```
out(net,httpRequest(uri,headers,httpGet()));
in(net,(hs : list(Header), message : String));
out(stdout, message);
|
in(net,(hs : list(Header), req: HttpReq));
out(process,req);
in(process, resp : HttpResp);
out(net,httpResponse(origin(hs),resp));
```

which would start one process which would send out an HTTP GET request on the channel net, and await a response. Once that response has been received, it will de-structure it and send the contents of the HTTP Response to stdout. Meanwhile, another process is set up to receive the same request on net, de-structure it into its component parts, then handle the request, and send back an appropriate HTTP response. The latter may well be out of the our abilities, however the we would aim to create something capable of handling something a little more impressive than the former.

1.2 Approach

We aim to build a compiler for the applied π-calculus. The language we plan to do this in is Haskell, due to familiarity using both the language itself and the parsec library [LM01], a powerful parser combinator library.

We will also be building a little web playground for our initial efforts to interact with. This will give us a good idea of how our implementation will interact with real world servers, if at all.

2 Background

2.1 CCS, π-calculus and the applied π-calculus

The applied π-calculus [AF01] is an extension of π-calculus [MPW92] which itself as an extension of the work Robert Milner did on the Calculus of Communicating Systems (CCS) [Mil82]. Here we will give a brief introduction to all three, and show how each extends upon the next.

$$
\begin{aligned}
P ::= \quad & \emptyset \\
& a.P_1 \\
& A \\
& P_1 + P_2 \\
& P_1 | P_2 \\
& P_1[\tfrac{b}{a}] \\
& P_1 \backslash a
\end{aligned}
$$

2.2 Haskell

2.2.1 Parsec

```
data Tree = Leaf | Node Int Tree Tree
```

2.2.2 Concurrency

3 Plan

Currently, we have completed most of our background research, and have started building a small web server in Haskell. The purpose of the latter is firstly to learn more about web frameworks in general, and secondly, once we have built a basic implementation, we can start to see how easy it will be to interact with conventional servers.

3.1 Milestone Dates

There are roughly 20 weeks between now and 23rd June, the preliminary archive submission deadline. We therefore aim to split that time into ten periods of two weeks, and at the end of each of those periods we would like to achieve the following:

3.1.1 7th March

Basic playground built

3.1.2 21st March

Able to parse a subset of the language `[ 0, in , out , let , ; , | ]`

3.1.3 4th April

Basic interaction with small servers i.e. `in(a,message);out(a,message)`

3.1.4 18th April

Able to parse `[if then else, new, !]`

3.1.5 2nd May

Basic handling of types

3.1.6 16th May

Able to parse basic functions such as `pair(x,y) , enc(x)`

3.1.7 30th May

3.1.8 13th June

3.1.9 20th June

3.1.10 23rd June

3.1.11 27th June

3.1.12 30th June

4 Evaluation

As mentioned before, there are already existing implementations of pure π-calculus like languages (Pict [PT97] [Pie96], occam-π). At this stage it is difficult to tell whether

References

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL'01: Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 104–115. ACM Press, 2001.

[LM01] Daan Leijen and Erik Meijer. Parsec: Direct style monadic parser combinators for the real world. Technical Report UU-CS-2001-27, Department of Computer Science, Universiteit Utrecht, 2001.

[Mil82] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts i and ii). *Information and Computation*, 100:1–77, 1992.

[Pie96] Benjamin C. Pierce. Programming in the pi-calculus: A tutorial introduction to pict (pict version 3.8d). Technical report, 1996.

[PT97] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. Technical report, Indiana University, 1997.

[RS13] Mark D. Ryan and Ben Smyth. Applied pi calculus, 2013. 2013 revision.