A Graphical Representation of the Solos Calculus

CM30082 Project Proposal

Adam Lassiter

October 2017

Contents

1 Problem Description

1.1 Current Problem

As computer processor clock speeds have begun to stagnate in performance increase per year, manufacturers have begun to shift focus to multiple cores in search of greater performance. Supercomputers for decades have operated around the idea of clusters and massive parallelism. For this reason there is now more than ever a need to study the mathematical effects of concurrency in computation.

1.2 Existing Solutions

1.2.1 λ-calculus

Developed by Alonzo Church in the 1930s, the λ-calculus is a more mathematical representation of a computable function. It forms a universal model of computation and can contain an encoding of any single-taped Turing machine. The λ-calculus became a popular formal system within which to study properties of computation. The λ-calculus consists of an expression M built from the following terms:

$$M \quad ::= \quad a \quad | \quad (\lambda x.M) \quad | \quad (MN)$$

where the expressions describe a variable, an abstraction and an application of a function on an argument. From this, any computable function can be constructed and computation is achieved through a series of reductions.
While the λ-calculus has been successful and been studied by various areas of academia outside of Computer Science, it is limited in application by its fundamentally 'single-process' model and struggles to describe multiple systems working and communicating together.

1.2.2 π-calculus

Similar to the λ-calculus as described in 1.2.1, there exists the π-calculus for the study of concurrent computation.
The π-calculus is constructed from the recursive definition of an expression P:

$$P \quad ::= \quad P|P \quad | \quad c(x).P \quad | \quad \bar{c}(x).P \quad | \quad vx.P \quad | \quad !P \quad | \quad 0$$

where the expressions describe each of concurrency, waiting for input, providing output, name binding (similar to $\lambda x.P$), replication (which itself is $!P :- P|!P$) and process termination respectively. While it provides the expressiveness required for Turing-completeness, it does not lend itself to understandability nor clarity of the problem encoding when presented as a standalone expression.

1.2.3 Solos Calculus

As defined by Laneve and Victor [1999], the Solos calculus is constructed from *solos* ranged over by $\alpha, \beta, \ldots$ and *agents* ranged over by $P, Q, \ldots$ as such:

$$\begin{array}{lll} \alpha ::= & u\tilde{x} & \text{(input)} \\ & \bar{u}\tilde{x} & \text{(output)} \end{array}$$

$$\begin{array}{lll} P ::= & 0 & \text{(inaction)} \\ & \alpha & \text{(solo)} \\ & Q|R & \text{(composition)} \\ & (x)Q & \text{(scope)} \\ & [x = y]Q & \text{(match)} \end{array}$$

This was further developed by Laneve et al. [2001] to provide a one-to-one correspondence between these expressions and 'diagram-like' objects. This provides a strong analog to real-world systems and an applicability to be used as a modelling tool for groups of communicating systems.

1.2.4 Others

There also exist many further variants, such as the Fusion calculus — itself a superset of π-calculus — and CSP (or communicating sequential processes). These aim to improve on the π-calculus in making the problem encoding more understandable, but at the cost of a more complicated language, making them harder to implement and prove reductions in.

1.3 Project Aims

The project seeks to provide an implementation of both the Solos calculus, for which there exists an encoding of the π-calculus within itself, and also of Solos diagrams, an intuitively graphical representation of the calculus. These diagrams can then aid in understanding of reductions of process calculi, while maintaining a clean and simple language with strong provable properties. Furthermore, the design of the Solos calculus is such that it is fully asynchronous. That is, unlike in the π-calculus where processes 'block' while waiting for inputs/outputs, it is shown through the construction of the Solos calculus that no such system is required — however there still exists a way of building such a system to give the effects of the π-calculus should it be desired. For these reasons, the Solos calculus is found to be an interesting alternative to the more common π-calculus and is therefore chosen as the project focus.

1.4 Objectives

Throughout the project, the following objectives and landmarks are hoped to be achieved:

- Research current models of concurrency calculus and understand common use-cases and environments.
- Build a system capable of computing Solos calculus expressions.
- Be able to convert expressions to diagrams.
- Visualise and interact with/modify diagrams and watch reductions step-by-step.
- Evaluate qualities of the system as a learning tool.

2 Requirements Specification

The following seeks to provide an estimate for the progress path the project will take, judging by initial research and planning. These are likely to evolve over time, as are priorities of each requirement ($must > should > may > might$).

2.1 Solos Expressions

The first functional part of the system will be focused on text-based Solos Expressions only. This will form a basis and back-end to build upon to implement other features. This system:

- *Must* support input and output of text representing Solos Expressions.
- *Must* be able to perform reductions on inputted expressions and return them.
- *Should* provide this functionality through an accessible user interface.
- *May* be implemented to evaluate expressions concurrently.

2.2 Solos Diagrams

This will be the first extension of the system to allow interoperability with Solos Diagrams, an equivalent representation of the calculus, which has intuitive graphical representations. This system:

- *Must* support input and output of said Solos Diagrams.
- *Must* provide a conversion between Diagrams and Expressions.
- *Should* be able to provide the same reductions as in the Expressions.
- *May* be able to produce step-by-step reductions.
- *Might* be implemented to evaluate diagrams concurrently.

2.3 Diagram Visualisation and Interaction

The second extension will build on the previous to provide a graphical interface for input/output/manipulation of Solos Diagrams giving a clear visual representation of how the calculus works. This system:

- *Must* support visualisation of Solos Diagrams through a GUI.
- *Must* support input/output of a (potentially manipulated) Diagram from/to an Expression.
- *Should* support manipulation of a Diagram through use of the GUI only.
- *Should* be both intuitive and feature-complete to provide an obvious improvement over text-based input as described in Section 2.1.
- *May* show step-by-step reductions of Diagrams.

3 Project Plan

3.1 Phases

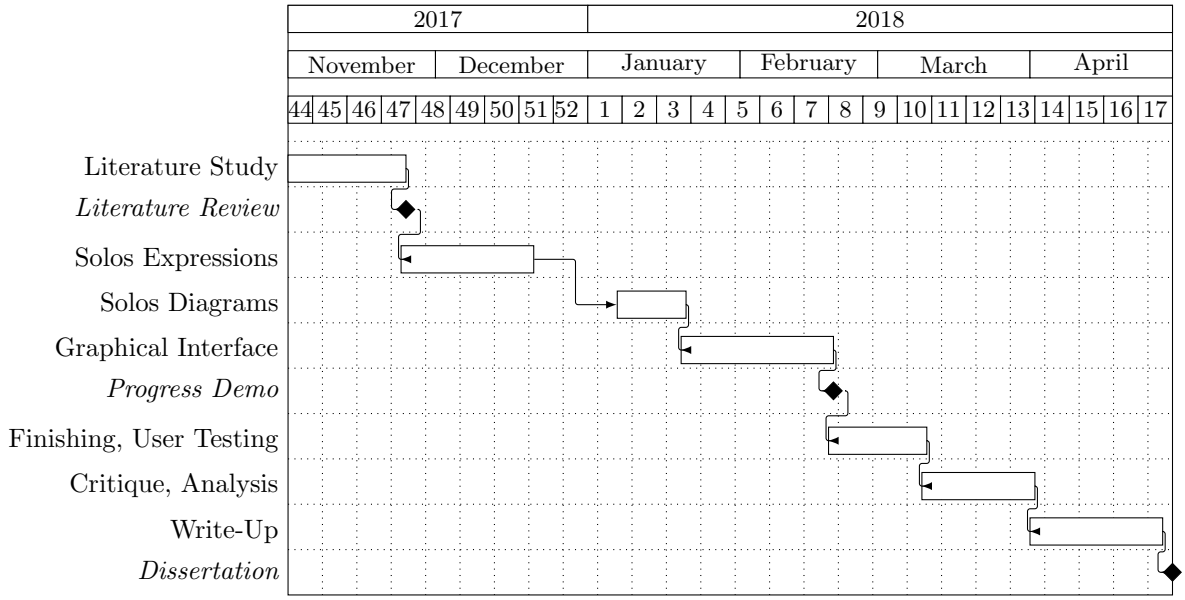
The project will be broken down into the following stages, with each building on the previous. This allows flexibility if there is not enough time as an incomplete extension will still leave a working system.

1. Literature review
2. Core system (Solos Expressions) implementation and unit testing
3. Extension 1 (Solos Diagrams) implementation and integration testing
4. Extension 2 (Graphical Visualisation) implementation and integration testing
5. Analysis, write-up and conclusion

3.2 Milestones and Deliverables

Milestone	*Target Date*
Project proposal	27th October, 2017
Literature review	24th November, 2017
Minimum viable product — Solos Expressions	20th December, 2017
Extension 1 — Solos Diagrams	20th January, 2018
Extension 2 — Graphical representation	19th February, 2018
Progress demonstration	19th February, 2018
User testing and critical analysis	10th March, 2018
Dissertation	4th May, 2018

3.3 Time Breakdown



4 Resources

4.1 Technology

The system will be developed as a Python application, as this allows for quick, modular development as performance is not critical, but still allows for multiple processes without hassle. The GUI is likely to be written using Tcl/Tkinter, as it is the common choice for Python graphical interfaces and is suited to 2D visualisation well-enough to be useful, while allowing easy implementation of interactive features.

4.2 Literature

Most literature for the project has already been obtained through both digital copies of papers and journals as well as physical books on the subject. While the topic of Solos calculus forms a niche area of research, the study of π-calculus is well-studied and has a large amount of relevant information that may be carried over.

4.3 Testing and Evaluation

In the scope of a learning and teaching aid on the subject, the system will need to be assessed for ease-of-use and for how well it explains the topic. For this, a small group of volunteers with moderate Computer Science understanding must be sourced, likely from a group of peers.

5 References

Cosimo Laneve and Björn Victor. Solos in concert. In *Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999 Proceedings*, pages 513–523. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-48523-0.

Cosimo Laneve, Joachim Parrow, and Björn Victor. Solo diagrams. In *Theoretical Aspects of Computer Software: 4th International Symposium, TACS 2001 Sendai, Japan, October 29–31, 2001 Proceedings*, pages 127–144. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-45500-4.