Honours Project Background Chapter

Leon Lee

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2024

Abstract

Chapter 1

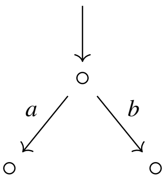
Background

1.1 Process Algebra

With the growing complexities of software and systems of the world, it is key to have methods of modelling more complex systems to get a better understanding of the underlying behaviour behind processes. Efforts have been made in sequential programming as early as the 1930s with Turing Machines, and the λ-calculus. Systems in real life are rarely sequential however, and usually involve multiple processes acting simultaneously, sometimes even synchronising to interact with each other to perform tasks. These tasks that involve modelling multiple processes at once are referred to as a *Concurrent System*. It is clear to see that brute forcing solutions to these problems are significantly harder than a sequential system - the processing time will grow exponentially as the number of processes increase, and modelling a system like a colony of ants is near impossible. Therefore, we will need some way to formalise these Concurrent Systems.

Concurrency has been studied in many different ways, though with the earliest the 1960s with some other notable models being Petri nets, or the Actor Model. Process Algebras are one such method of modelling a Concurrent System, where the process is modelled in such a way that it is akin to the Universal Algebras of mathematics - in which operations are defined in an axiomatic approach to create a structurally sound way of defining concurrent systems. (Baeten, 2005) It is easily possible to model simple systems as a flow chart or diagram as you will be able to see throughout this paper, but a formal approach like process algebras will make way for modelling more complex systems, and lays the groundwork to provide a solid foundation to prove and base claims for such systems.

A simple example in action is a process algebra where we only consider the alternative composition operator $+$, where applied to a process $a+b$ means "Choose a, or choose b". Process algebras can typically be modelled in a *Process Graph*, which are diagrams that employ "states", and "actions" to show the traces, or paths, that a process can take. In this case, the process $a+b$ can be modelled in the following way:



Where the graph begins at the top into the first node, and then can either progress to the left node via the action *a*, or the right node via the action *b*. *a* and *b* are the actions, e.g. “eat” and “drink”, while the nodes are the states, e.g. “apple” and “water”

The axioms of the $+$ operator of BPA are as follows:

- **Commutativity**: $a+b$
- **Associativity**: $(a+b)+c=a+(b+c)$
- **Idempotency**: $a+a=a$

Comparable to the operation axioms of a Group or Ring in Mathematics, every other operation in a process algebra is constructed similarly. In practice, most process algebras will have some form of alternative composition, but this is a very simplified example and the developed process algebras that exist are designed to handle a lot more complex situations such as unobservable actions, commonly referred to as τ-actions, recursion, which lets a process repeat itself or other processes, and deadlock, which is a state where no desirable outcomes can be reached.

There are many process algebras that exist, the most famous and seminal being CSP (Brookes et al., 1984), CCS (Milner et al., 1980), and ACP (Bergstra and Klop, 1984), (Bergstra and Klop, 1989), with some other popular calculi being the π-calculus and its various extensions (Engberg and Nielsen, 1986), (Parrow and Victor, 1998), (Abadi and Gordon, 1999) which have been used to varying degrees in fields like Biology, Business, and Cryptography, or the Ambient Calculus (Cardelli and Gordon, 1998) which has been used to model mobile devices.

1.2 Encodings of Process Algebra

With the growing number of process algebras, one might begin to ask if there is a way of comparing different process algebra to each other to find the single best one, as a parallel to Turing Machines and the Church-Turing thesis. However, the wide range of applications that different process algebra are used for makes that rather impractical, and the goal of unifying all process algebra into a single theory seems further and further away as more process algebra for even more specified tasks get created.

A more reasonable approach is to compare different process algebras and their expressiveness, two main relevant methods being *absolute* and *relative* expressiveness.(Parrow, 2008) Absolute expressiveness is the idea of comparing a specific process algebra to a question and seeing if it can solve the problem - e.g. if a process algebra is Turing

Complete. However, this merely biparts different algebra - the process algebra that are able to solve a specified problem, and the ones who aren't (Gorla, 2010). Therefore, the question of relative expressiveness - i.e. how one language compares to another is a lot more useful in terms of categorising different process algebras by expressiveness.

A well studied way of comparing expressiveness is through an "encoding", and whether an algebra can be translated from one to another, but not vice versa (Peters, 2019). The general notion of an encoding is not defined by clear boundaries, and the criterion for a valid encoding may vary language to language, but work has been made to try and generalise the notion of a "valid" encoding (Gorla, 2010), (Glabbeek, 2018).

1.3 CSP

CSP (Communicating Sequential Processes) (Brookes et al., 1984) is a Process Algebra developed by Tony Hoare based on the idea of message passing via communications. It was developed in the 1980s and was one of the first of its kind, alongside CCS by Milner. CSP uses the idea of action prefixing which is where operators are of the syntax $a \rightarrow P$, where a is an event and P is a process.

As taken from van Glabbeek (2017), a complete list of CSP expressions is as follows:

$$P,Q ::= STOP \mid \mathrm{div} \mid a \rightarrow P \mid P \sqcap Q \mid P \Box Q \mid P \lhd Q \mid \\ P||_A Q \mid P \backslash A \mid f(P) \mid P \triangle Q \mid P \theta_A Q \mid p \mid \mu p.P$$

where the operators are: *inaction*, *divergence*, *action prefixing*, *internal choice*, *external choice*, *sliding choice*, *parallel composition*, *concealment*, *renaming*, *interrupt*, and *throw*

1.4 ACP

ACP (Algebra of Communicating Processes) is a Process Algebra developed by Jan Bergstra and Jan Willem Klop (Bergstra and Klop, 1984). Compared to CSP, ACP isn't based on communications, and instead built up with an axiomatic approach in mind which does away with the idea of action prefixing and instead can allow for unguarded operations. ACP_τ (Bergstra and Klop, 1989) is an extension of ACP that includes an extra action τ which is used to represent actions that are unobservable, or changeable, from a human perspective.

Taken from the same paper as CSP, van Glabbeek defines the grammar of ACP with relational renaming (ACP_R) as such:

$$P,Q ::= a \mid \delta \mid E+F \mid E.F \mid E||F \mid E \underline{||} F \mid E|F \mid \partial_H(E) \mid \mathcal{R}(E) \mid X \mid \langle X|\mathcal{S}\rangle$$

where the operators are: *action*, *deadlock*, *alternative composition*, *sequential composition*, *merge*, *left merge*, *communication*, *restriction*, *relational renaming*, and X and $\langle X|\mathcal{S}\rangle$ being a recursive specification

Bibliography

Martin Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, January 1999. ISSN 0890-5401. doi: 10.1006/inco.1998.2740. URL `https://www.sciencedirect.com/science/article/pii/S0890540198927407`.

J. C. M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2):131–146, May 2005. ISSN 0304-3975. doi: 10.1016/j.tcs.2004.07.036. URL `https://www.sciencedirect.com/science/article/pii/S0304397505000307`.

J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1):109–137, January 1984. ISSN 0019-9958. doi: 10.1016/S0019-9958(84)80025-X. URL `https://www.sciencedirect.com/science/article/pii/S001999588480025X`.

J. A. Bergstra and J. W. Klop. ACPt a universal axiom system for process specification. In Martin Wirsing and Jan A. Bergstra, editors, *Algebraic Methods: Theory, Tools and Applications*, pages 445–463, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. ISBN 978-3-540-46758-8.

S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A Theory of Communicating Sequential Processes. *J. ACM*, 31(3):560–599, June 1984. ISSN 0004-5411. doi: 10.1145/828.833. URL `https://dl.acm.org/doi/10.1145/828.833`.

Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *Foundations of Software Science and Computation Structures*, pages 140–155, Berlin, Heidelberg, 1998. Springer. ISBN 978-3-540-69720-6. doi: 10.1007/BFb0053547.

Uffe Engberg and Mogens Nielsen. A Calculus of Communicating Systems with Label Passing. *DAIMI Report Series*, (208), May 1986. ISSN 2245-9316. doi: 10.7146/dpb.v15i208.7559. URL `https://tidsskrift.dk/daimipb/article/view/7559`. Number: 208.

Rob van Glabbeek. A Theory of Encodings and Expressiveness. *CoRR*, abs/1805.10415, 2018. URL `http://arxiv.org/abs/1805.10415`. arXiv: 1805.10415.

Daniele Gorla. Towards a unified approach to encodability and separation results for process calculi. *Information and Computation*, 208(9):1031–1053, September 2010. ISSN 0890-5401. doi: 10.1016/j.ic.2010.05.002. URL https://www.sciencedirect.com/science/article/pii/S0890540110001008.

Robin Milner, G. Goos, J. Hartmanis, W. Brauer, P. Brich Hansen, D. Gries, C. Moler, G. Seegmüller, J. Stoer, and N. Wirth, editors. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 1980. ISBN 978-3-540-10235-9 978-3-540-38311-6. doi: 10.1007/3-540-10235-3. URL http://link.springer.com/10.1007/3-540-10235-3.

J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.98CB36226)*, pages 176–185, June 1998. doi: 10.1109/LICS.1998.705654. URL https://ieeexplore.ieee.org/document/705654. ISSN: 1043-6871.

Joachim Parrow. Expressiveness of Process Algebras. *Electronic Notes in Theoretical Computer Science*, 209:173–186, April 2008. ISSN 1571-0661. doi: 10.1016/j.entcs.2008.04.011. URL https://www.sciencedirect.com/science/article/pii/S1571066108002260.

Kirstin Peters. Comparing Process Calculi Using Encodings. *Electronic Proceedings in Theoretical Computer Science*, 300:19–38, August 2019. ISSN 2075-2180. doi: 10.4204/EPTCS.300.2. URL http://arxiv.org/abs/1908.08633v1.

Rob van Glabbeek. A Branching Time Model of CSP. In Thomas Gibson-Robinson, Philippa Hopcroft, and Ranko Lazić, editors, *Concurrency, Security, and Puzzles: Essays Dedicated to Andrew William Roscoe on the Occasion of His 60th Birthday*, pages 272–293. Springer International Publishing, Cham, 2017. ISBN 978-3-319-51046-0. doi: 10.1007/978-3-319-51046-$0_1$4.*URL*https://doi.org/10.1007/978-3-319-51046-0_14.