Internship Proposal

Implementing Reversible Process Algebras

Keywords	Process Algebra, Concurrency, Reversible Calculi, Process Semantics
Location	Augusta University, GA, USA
Advisor	Clément Aubert, School of Computer and Cyber Sciences, Augusta University
Preferred Skills	Interest in formal languages and specification, curiosity for distributed computation, abstract and logic reasoning, programming skills.

Context. Process algebras (π-calculus, CCS, Ambiant calculus, etc.) are an abstraction of concurrent systems useful to study, specify and verify distributed behaviors. Implementing process calculi serves three overlapping goals:

- It allows to machine-check theorems and definitions [1,2] using proof assistants such as Coq [3], resulting sometimes in simplications [2] or the finding of regrettable imprecisions and errors [4].
- Using it as an actual programming language, it enables the implementation of toy programs [5] that exemplifies the purpose and expressivity of the calculus.
- It can also be used as a specification language: typically, the Proverif tool [6], which implements the applied π-calculus [7], has been used to certify and model security protocols in a variety of areas [8].

The CCS language undergoes two different efforts making it amenable to represent computation that can move backward and forward: Reversible CCS (RCCS) [9] and CCS with keys (CCSK) [10] were both developed with the goal of becoming *the* extension to CCS providing a better understanding of the mechanisms underlying reversible concurrent computation—they actually turned out to be the two faces of the same coin [11]. Reversible computation in general has received a lot of attention from different communities [12], and the study of reversible process calculi has made important progresses in the recent years [12, Sect. 6]. However, aside from SimCCSK [13]–which is not publicly available and not maintained since 2008–no implementation of concurrent, reversible CCS exists.

Goals. The student will work toward an implementation of either CCSK, RCCS, or a declension of them [14], possibly taking inspiration of existing implementation of forward-only CCS (among wich this project or this web-interface [15]) or of intermediate languages such as HOcore [4]. A great care will be required toward good software engineering practices, the development of good examples, the possible certification of some results using machine-checked proof, and / or the implementation of an efficient mechanism to distribute the generation of keys and identifiers.

Perks. Dr. Aubert has an history of involving undergraduate, graduate and post-graduate students in his research, and can tune the level and nature of his engagement in the student's research based on their needs and tastes. Even if the advisor have a strategy in mind, they remain open to suggested deviations from this program based on mutual interest. The student will be working on a ground-breaking topic that is still within reach with limited theoretical background.

References.

[1] D. Hirschkoff, A full formalisation of pi-calculus theory in the calculus of constructions, in: E.L. Gunter, A.P. Felty (Eds.), Springer, 1997: pp. 153–169. https://doi.org/10.1007/BFb0028392.

[2] J. Despeyroux, A higher-order specification of the pi-calculus, in: J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, T. Ito (Eds.), Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings, Springer, 2000: pp. 425–439. https://doi.org/10.1007/3-540-44929-9_30.

[3] Coq documentation, (n.d.). https://coq.github.io/doc/ (accessed February 9, 2020).

[4] P. Maksimovic, A. Schmitt, HOCore in coq, in: C. Urban, X. Zhang (Eds.), Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings, Springer, 2015: pp. 278–293. https://doi.org/10.1007/978-3-319-22102-1_19.

[5] R. Affeldt, N. Kobayashi, A coq library for verification of concurrent programs, Electron. Notes Theor. Comput. Sci. 199 (2008) 17–32. https://doi.org/10.1016/j.entcs.2007.11.010.

[6] B. Blanchet, Modeling and verifying security protocols with the applied pi calculus and ProVerif, Foundations and Trends in Privacy and Security. 1 (2016) 1–135. https://doi.org/10.1561/3300000004.

[7] M. Abadi, B. Blanchet, C. Fournet, The applied pi calculus: Mobile values, new names, and secure communication, J. ACM. 65 (2018) 1:1–1:41. https://doi.org/10.1145/3127586.

[8] M.D. Ryan, B. Smyth, Applied pi calculus, in: V. Cortier, S. Kremer (Eds.), Formal Models and Techniques for Analyzing Security Protocols, IOS Press, 2011: pp. 112–142. https://doi.org/10.3233/978-1-60750-714-7-112.

[9] V. Danos, J. Krivine, Reversible communicating systems, in: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings, Springer, 2004: pp. 292–307. https://doi.org/10.1007/978-3-540-28644-8_19.

[10] I. Phillips, I. Ulidowski, Reversing algebraic process calculi, in: L. Aceto, A. Ingólfsdóttir (Eds.), Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings, Springer, 2006: pp. 246–260. https://doi.org/10.1007/11690634_17.

[11] I. Lanese, D. Medić, C.A. Mezzina, Static versus dynamic reversibility in CCS, Acta Inform. (2019). https://doi.org/10.1007/s00236-019-00346-6.

[12] B. Aman, G. Ciobanu, R. Glück, R. Kaarsgaard, J. Kari, M. Kutrib, I. Lanese, C.A. Mezzina, L. Mikulski, R. Nagarajan, I.C.C. Phillips, G.M. Pinna, L. Prigioniero, I. Ulidowski, G. Vidal, Foundations of reversible computation, in: I. Ulidowski, I. Lanese, U.P. Schultz, C. Ferreira (Eds.), Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action Ic1405, Springer, 2020: pp. 1–40. https://doi.org/10.1007/978-3-030-47361-7_1.

[13] G. Cox, SimCCSK: simulation of the reversible process calculi CCSK, Master's thesis, University of Leicester, 2010. https://leicester.figshare.com/articles/thesis/ SimCCSK_simulation_of_the_reversible_process_calculi_CCSK/ 10091681.

[14] C. Aubert, D. Medić, Explicit identifiers and contexts in reversible concurrent calculus, in: S. Yamashita, T. Yokoyama (Eds.), Reversible Computation - 13th International Conference, RC 2021, Virtual Event, July 7-8, 2021, Proceedings, Springer, 2021: pp. 144–162. https://doi.org/10.1007/978-3-030-79837-6_9.

[15] J.-F. Gillet, D. Willame, Calculus of Communicating Systems: A web based tool in Scala, Master's thesis, Université de Namur, 2017. https://researchportal.unamur.be/files/30127909/ GILLET_WILLAME_Memoire.pdf.