Implementing Reversible Concurrent Programs Specification Language

Fall 2022 Capstone Proposal

Keywords	Concurrent Programs, Reversible Programs Specification, Theory of Computation
Client	Clément Aubert, School of Computer and Cyber Sciences, Augusta University
Contact	caubert@augusta.edu
Preferred Skills	Interest in formal languages and specification, curiosity for distributed computation, abstract and logic reasoning, programming skills.
Tools	Java, Maven, git, markdown

1 Context

Process algebras (π-calculus, CCS, Ambiant calculus, etc.) are an abstraction of concurrent systems useful to study, specify and verify distributed programs. This project is concerned with the implementation of such an abstraction for *reversible* programs: in short, imagine that we are designing a language to test that reversible computers behave "as expected". We can then write a term describing what a program is supposed to do, and then check that the program indeed do what it is supposed to do, that is, *match its specification*. This is crucial in testing and debugging concurrent programs, and even more so for *reversible* concurrent programs.

Implementing such a specification language (called "process calculi") serves many overlapping goals, but in our case the main purpose is to then use it as an actual programming language, to enable the implementation of toy programs that exemplifies the purpose and expressivity of the calculus. CCSK [1] is the specification language for reversible systems that was developed with the goal of providing a better understanding of the mechanisms underlying reversible concurrent computation. Reversible computation in general has received a lot of attention from different communities [2], and the study of reversible process calculi has made important progresses in the recent years [2, Sect. 6]. However, aside from SimCCSK [3]–which is not publicly available and not maintained since 2008–no implementation of concurrent, reversible CCS exists until IRDC-CCSK was started.

Goals. This capstone project offers an opportunity to help with the development of IRDC-CCSK, starting by understanding its current codebase, tests, documentation and examples. Once some familiarity will be acquired, the main goals will be to help test, debug, document, and generally improve our tool. Ideally, solid progress toward new features would be developed and tested as well. A snapshot of our current issues and goals can be seen on our repository.

Perks. Throughout the project the students will have continuous support from the active mentor, Dr. Aubert. He is an expert in theory of computation, introduces to and pushes forward good practises in software engineering, and intend to submit a paper to the 15th Conference on Reversible Computation about this active work, listing as authors all the active contributors.

2 Learning Objectives

This project will familiarize the student with:

1. the operations and terminology of specification languages for concurrent programs (process algebras),
2. the importance and role of reversible programs,
3. modern software development practices and open source projects (from documentation to testing, from version control system to continuous integration / delivery).

In addition to helping them craft their programming skills.

During this project the student will also obtain experience working with these tools:

- The Java programming language – prior experience is not required, but will facilitate better outcomes,
- The Maven automation tool – to automate and ease the workflow of compiling, testing and deploying,
- `git` – version control system, and its hosting at Github. In particular, the project currently uses GitHub Actions as its continuous integration and continuous delivery (CI/CD) system.

3 Project Timeline and Phases

The student will progress toward learning objectives by completing four phases of study, as outlined below.

Phase 1: Familiarity with tools and resources Duration: 2-3 weeks

The primary focus in this stage is to develop familiarity with the existing workflow, and to understand the scientific importance of this project. Exercises and resources will be provided to help the student getting familiar with specification language for reversible systems, and guidance will be given to set-up the working environment correctly.

Phase 2: Understanding the existing codebase Duration: 2-3 weeks

In this stage the focus is in studying the existing codebase, and possibly documenting it: by reviewing the existing code and critically questioning it, the students are expected to find omissions, flaws, or corner case that needs to be accounted for. Reviewing and cataloguing the existing issues will also help in correctly understanding the distance between the goals and the current state of development.

Phase 3: Contributing to and reviewing submission draft Duration: 2-3 weeks

The students will be asked to review and edit the submission draft to the 15th Conference on Reversible Computation. This includes getting familiar with the process of article submission and the tools used in the community (LaTeX) if the student desires, and to contribute to the scientific writing of the paper describing the tool being developed.

Phase 4: Getting practical with the implementation Duration: until the end of the semester

After developing thorough understanding of the existing codebase, its shortcomings and its goals, it is time for practical implementation. In the final stage the focus is to write the first pull request(s), to clearly document the contribution, and to write test cases for the added features.

4 Getting Started

- Have a look at the existing codebase: https://github.com/CinRC/IRDC-CCSK
- Practice Java at Codewars: https://www.codewars.com/collections/java/
- Read *"Foundations of Reversible Computation*: https://doi.org/10.1007/978-3-030-47361-7_1

References

[1] 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.

[2] 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.

[3] 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.