# Undergraduate Research and Scholarship I

**CURS 2990 - Summer 2024**

Clément Aubert & Nate Schwartz

May 8, 2024

## Presentation

This class is aimed at fostering and developing abilities to conduct individual research in Computer Science. As such, its structure may be evolving and concrete expectations are hard to establish, but the main student learning outcomes are that students who successfully complete this course should be able to:

- Explain concisely the general context and interest of selected research papers and questions,
- Understand and reformulate the motivations and impact of the proposed research topic,
- Communicate in writing and orally the purpose of their research, using standard tools in the discipline (i.e., LaTeX manuscript and slides / poster presentations),
- Master the technologies used in their field (LaTeX, git, BiBTeX, markdown, etc.).

### Research Project Presentation

**In a nutshell:** As Ugo Dal Lago observed in a public message, there is no "reversible" machine capable of executing $\lambda$-terms. The goal of this course is to develop such a model of computation, using the approach designed in the "Non-Deterministic Abstract Machines" paper [1], to obtain machines capable of undoing their reductions of $\lambda$-terms (and possibly of $\pi$-calculus terms) for the first time.

**In more details:** Reversible computing is gaining a lot of traction as a debugging and forensic tool, but also due to its connections to quantum and adiabatic computing [2]. We also know since the 70's and *Bennett's trick* that it is possible to "revert" any Turing Machine, in a precise technical sense [3]. The question this class will investigate is: is it possible to "revert" another fundamental model of computation, namely the $\lambda$-calculus [4]?

Answering positively this question requires to

1. Define a reversible abstract model of computation $M$:
   a) Define a forward execution, i.e., rules expressing that the model $M$ in a state $s$ will reduce the $\lambda$-terms $u$ to $u'$ and move to state $s'$ in one forward step $:(M, s, u) \to (M, s', u')$,
   b) Define a backward execution, i.e., rules expressing that the model $M$ in a state $s$ will reduce the $\lambda$-terms $u$ to $u'$ and move to state $s'$ in one backward step $:(M, s, u) \rightsquigarrow (M, s', u')$,
2. Ensure that the model is "correct":
   a) By showing that it can correctly execute forward and backward some simple examples,
   b) By proving properties such as the loop lemma, i.e., $(M, s, u) \to (M, s', u') \leftrightarrows (M, s', u') \rightsquigarrow (M, s, u)$,
   c) By comparing it to existing reversible functional languages [3],
   d) By exploring if there are additional features and properties revealed by this model that were not previously accessible.

Our exploration will start with an attempt to "revert" the recently developed "Non-Deterministic Abstract Machines" [1]. A good starting point will be Logan Beatty's slide that he used to explain our research project during Dr. Medić's Visit.

## Goals

### Educational

The main educational goal of the project is to **give to Nate the taste of a scholarly culture, and to introduce him to research questions at the intersection of Computer Sciences and Mathematics**. More precisely, our goals are to:

- Foster Nate's independence, to sharpen his capacities to solve problems by himself.
- Improve his self-efficacy, to believe in his capacities but also to identify when and how to ask for help.
- Encourage his curiosity, to drive his own intellectual journey.
- Introduce to the administrative aspects of research[1] by e.g. searching for good-quality journals and venues to submit our work, or funds to sparkle other projects.
- Become confident with technologies (git, LaTeX, references managers, markdown, …) common in research.

### Mentoring Philosophy

Mentoring undergraduate students requires a very delicate blend of imposed structure and complete freedom. The student needs to be given realistic goals, carved out from a larger research project piece by piece, along with the necessary resources to understand the purpose and history of the discipline. This requires to constantly zoom in and out, to remind the importance of the context without discouraging the student: "while many researchers worked on connected problems in the past and may enlighten our way, none of them (but you!) *tackled this exact problem*". This structuring material needs to be given in a concise manner to the student, "by need", so that they are not drowned by resources taking months (or even years) to absorb.

The second crucial ingredient is to learn how not to be in the student's way. This begins, to me, by making sure that the student's research project is not "on my critical path": I always strive to carve out from larger research projects smaller pieces that are of interest to students and to me, but not crucial to the success of my overall project. This gives them an opportunity to be really creative, even playful, without requiring me to constantly steer them in a predefined direction ("Sure, injecting this mathematical element in there would be nice, but it doesn't deserve my immediate goal" is *not* something I want to hear myself ever say!). This also makes it very easy to me to withdraw "just enough", so that the student become rapidly the subject expert: even if some of the time I could easily unwedge them, I prefer to observe them, regularly discuss with them, but never engage in their exact subject unless invited to–or if I see that their struggles are becoming counter-productive.

Because, overall, I believe this is the whole point of a good mentoring philosophy: I am not mentoring them to have them do exactly what I would have done, but to trace their own path, explore their own creativity, in a safe space that I will create for them using existing resources and constant support through their efforts and struggles.

### Research

Our main goals, aside from the educational goals detailed above, are to produce a research document that will be shared with experts in the field through self-archiving on the arXiv.org and github repositories. All the material produced will be released under Creative Commons licenses (or similar open-source licence), to ease distribution and re-use by the community.

[1] Something that already started, since Nate had a chance to review this syllabus.

### Timeline

The program will start on May 28 and finish on June 28, 2024. Weekly meeting are scheduled on Wed., from 11 to 12am, in Dr. Aubert's office, and as needed.

A rough timeline of student involvement:

| Week | Topic |
|---|---|
| 1 | Discussion on use-cases for reversible computation, introduction to lambda-calculus [4, pp. 51–73] |
| 2 | Introduction to LaTeX and markdown (word processing software used in Computer Science), introduction to mathematical writing conventions, and presentation of the Non-Determinstic Abstract Machines [1] |
| 3 | First attempt at defining a Reversible Non-Determinstic Abstract Machines, study of reversible process algebras using resources developed locally. |
| 4 | Prove "sanity checks" on model developed (perform simple computation, review classical examples), draft poster presentation. |
| 5 | Test model developed on process calculi, finalize poster and practise poster presentation. |

In addition, Nate will be enrolled in the SSP, and will have to attend their workshops and other mandatory events (such as the Symposium).

### Tools

Will be used during this program, among other resources:

- References and our document will be shared on https://github.com/CinRC/reversible-lambda,
- teams will be our main method of "live" communication,
- Email will be our main method of asynchronous communication.

## Miscellaneous

- Reservation of rights: I reserve the right to change this syllabus without limitation and without prior notice. If I do substantially modify any item or policy, I will notify you during a lecture, or send an e-mail to your `augusta.edu` e-mail account.
- Download a `pdf` version of this page.
- Contact: caubert@augusta.edu
- Created with debian, pandoc and latex.
- All my documents are under Creative Commons Attribution 4.0 International License. Sources are available upon motivated request.
- You will need a `pdf` reader to consult some of the documents: I recommend choosing an open-source `pdf` reader.

## References

[1] M. Biernacka, D. Biernacki, S. Lenglet, A. Schmitt, Non-deterministic abstract machines, in: B. Klin, S. Lasota, A. Muscholl (Eds.), 33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022: pp. 7:1–7:24. https://doi.org/10.4230/LIPICS.CONCUR.2022.7.

[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] R. Glück, T. Yokoyama, Reversible computing from a programming language perspective, Theor. Comput. Sci. 953 (2023) 113429. https://doi.org/10.1016/J.TCS.2022.06.010.

[4] B.C. Pierce, Types and programming languages, MIT Press, London, England, 2002.