Non-commutative logic and categorical semantics for resources in new systems programming languages

Advisor: Guillaume Munch-Maccagnoni, Inria, Nantes

9th November 2021

Topics

• Programming languages
• Semantics and logic
• Category theory

Location

• Nantes, France
• Inria, LS2N Lab, Gallinette team

Advisor

• Guillaume.Munch-Maccagnoni@Inria.fr

General presentation of the topic  A major evolution in programming languages in the recent years has been the advent of modern resource-management features in the systems programming languages C++11 (Stroustrup et al., 2015) and Rust (Matsakis and Klock II, 2014). In what one might initially think unrelated, many recent advances in the theory of programming languages are due to the Curry-Howard correspondence between functional programming, logic and category theory.

In our team, we have been studying some of these systems programming features, via the Curry-Howard correspondence, from the point of view of linear logic (Girard, 1987; Melliès, 2009; Baker, 1994), and established a formal connection with its sibling ordered (or non-commutative) logic (Combette and Munch-Maccagnoni, 2018). This opened further research directions on the general theme of merging of systems programming and functional programming (Munch-Maccagnoni, 2018) (with the ultimate goal of creating the Next Best Programming Language).
Of these research directions, some can interest mathematically-inclined young researchers who are interested in advancing the theory of programming languages and its connection to logic, others can interest those who are more looking for a hands-on approach to programming language development. This internship subject has as a starting point recent approaches to the categorical semantics of logic and computation, and can evolve into a PhD subject in either direction, in accordance with the tastes of the student.

**Context**  Recently, we have proposed the notion of ordered algebraic data types to model the types of resources in Rust/C++11: those types with an associated destructor, a clean-up action (such as freeing some memory, closing a file . . .) that has to be called reliably and predictably at the end of the value’s lifetime.

Ordered (or non-commutative) logic (Lambek 1968) is a linear logic that removes the isomorphism between $A \otimes B$ and $B \otimes A$. The types from Rust/C++11 are ordered because the two types can be distinguished by the order of destruction (first $B$ then $A$ vs. first $A$ then $B$).

Ordered algebraic data types are expressive compound types whose associated destructor is determined by the type; for instance there are two types of list: $\mu X.(1 \oplus A \otimes X)$ (where $\mu$ denotes the smallest fixed-point operator) defines the type of lists whose elements are destroyed in list order, whereas $\mu X.(1 \oplus X \otimes A)$ defines the type of lists whose elements are destroyed in reverse order.\(^1\) This proposition has deeper roots in Combette and Munch-Maccagnoni (2018) where we proposed a categorical model construction that interprets these resource types in a semantics of non-commutative logic.

These results are stated in the framework of call-by-push-value (CBPV) (Levy, 2004), a model of higher-order computation with side-effects, and more precisely linear CBPV developed by the advisor and his colleagues (Curien, Fiore, and Munch-Maccagnoni 2016), which is the focus of the internship.

Linear call-by-push-value is a model of computation with higher-order functions that describes the proper way for combining side-effects (Moggi, 1991) and resources (Girard, 1987), featuring a neat correspondence between (linear) logic and (effectful) computation. However, its results are currently limited for our purposes since they only concern (commutative) linear logic so far.

**Goals** We would like to extend the results from Curien, Fiore, and Munch-Maccagnoni (2016) into a “non-commutative call-by-push-value”, to serve as a starting point for integrating into the theory of functional programming the modelling of phenomena and features of systems programming (e.g. ordered algebraic data types, borrowing à la Rust, and more).

The internship task is to understand the paper Curien, Fiore, and Munch-Maccagnoni (2016) with the help of the advisor, and, as a research in team with the advisor, to investigate

\(^{1}\)In Rust:

```rust
pub struct List<A> { node: Option<Box<(A, List<A>)>>, }
pub struct List2<A> { node: Option<Box<(List2<A>, A)>>, }
```
the difficulties in the non-commutative case, and lastly, if successful, to contribute to the publication of the results in an international conference or journal.

Pre-requisites

Skills of any future young researcher:

- outstanding creativity,
- taste and capacity for acquiring a bibliographic knowledge of a topic,
- teamwork,
- clear and rigorous writing,
- good oral presentation skills.

More specifically for this topic, and less importantly:

- starting knowledge and strong interest in categorical semantics of logic and computation,
- interest or curiosity in systems programming (e.g. Rust), or proof theory, or linguistics (where non-commutative logic originates from), etc.

Notes

- Our team Gallinette provides a nice atmosphere. It is a large and young team of researchers in Nantes, specialised in logic, programming languages, and the formalisation of mathematics. Many colleagues work on the development and application of the Coq proof assistant.

- My speciality is in logic and denotational semantics (e.g. lambda-calculus). More recently, I have found this area of application of my works to programming languages\(^2\) and I started hacking and contributing to the OCaml language runtime. I interact both with researchers in logic/semantics and with OCaml/Rust researchers and implementers in the industry.

- Funding is available for travel (e.g. workshops and visits abroad). There are collaboration opportunities with M. Fiore (Univ. Cambridge, UK).

- In addition, a student with taste in programming will also have the opportunity, if they want, to get involved in side-projects meant to improve OCaml’s support for safe resource management in a concrete manner.

---

\(^2\)See for instance my presentation of this topic at the Collège de France: [https://www.college-de-france.fr/site/xavier-leroy/seminar-2018-12-19-11h30.htm](https://www.college-de-france.fr/site/xavier-leroy/seminar-2018-12-19-11h30.htm)
References


Guillaume Combette and Guillaume Munch-Maccagnoni. 2018. A resource modality for RAII. In *LOLA 2018: Workshop on Syntax and Semantics of Low-Level Languages* (2018-04-16). [https://hal.inria.fr/hal-01806634](https://hal.inria.fr/hal-01806634)


Guillaume Munch-Maccagnoni and Rémi Douence. 2019. Efficient Deconstruction with Typed Pointer Reversal (abstract). In *ML Family workshop* (2019). [https://hal.archives-ouvertes.fr/hal-02177326](https://hal.archives-ouvertes.fr/hal-02177326)
