sujet de stage de Master 2 recherche (2023)

Implicit computational complexity in Pi-calculus

Encadrant : Patrick Baillot (patrick.baillot@univ-lille.fr)
Page web : `https://pro.univ-lille.fr/patrick-baillot/`

Laboratoire d'accueil : CRIStAL, Université de Lille, Équipe *Sycomores*
Site web du laboratoire : `https://www.cristal.univ-lille.fr/?lang=fr`

Localisation : Lille

Context :

How can one define some simple and modular programming disciplines in a high-level programming or specification language in such a way that the corresponding programs exactly characterize a certain complexity class of functions, for instance polynomial time (FP) or polynomial space (FPSPACE) ? This is the goal of *implicit computational complexity* (ICC) which aims at providing such machine-independent characterizations for a large variety of complexity classes and of source languages, without refering to any explicit bound on time or space usage. This research area uses ideas and techniques coming from logic, recursion theory and type systems.

Most of ICC results have been obtained for sequential languages, initially for λ-calculus or the functional computing paradigm, and then for imperative and object-oriented languages. One of the classical results [BC92] is that of safe (or ramified) recursion which is a restriction of primitive recursion that characterizes the class FP of polynomial time functions. This approach is also illustrated by a type system for λ-calculus characterizing FP [LM93] .

Just as λ-calculus represents sequential computation, process calculi such as π-calculus [SW03] have been introduced to represent parallel and concurrent computation. This language represents processes communicating by messages sent through channels. However up to now ICC characterizations in π-calculus have hardly been explored.

Objective :

The goal of this internship is to give an ICC characterization of a complexity class in π-calculus. The paper [DY18] has defined a criterion on π-calculus processes inspired by safe recursion [BC92] and which ensures a polynomial bound on *causal complexity.* However it has not provided any expressivity result for the processes satisfying the criterion, for instance stating that say all FP or all FPSPACE functions would be representable. On the other hand the paper [HMP13] has given a criterion on programs of an imperative language with threads which characterizes the class PSPACE (which is equal to parallel polynomial time, as shown by complexity theory). However this imperative language is not as general as the π-calculus.

More recently [BG21, BGK21, Ghy21, BG22] have provided type systems allowing to obtain bounds on the parallel time complexity of π-calculus processes. Note that these are *not* ICC characterizations because the bounds do not belong to a specific class (e.g. polynomial bounds).

We propose to look for a typing criterion in π-calculus in the style of safe recursion which would characterize a complexity class. The most natural candidate would be the class FPSPACE. We suggest to use the techniques of [BG21, BGK21] to prove an upper bound on the complexity of the processes.

Expected background : It is expected to have some knowledge about λ-calculus or functional programming languages as well as about type systems. Some basic background on computational complexity is also necessary. Some notions of concurrent process calculi such as π-calculus or CCS would also be welcome but are not compulsory.

Références

[BC92] Stephen J. Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2 :97–110, 1992.

[BG21] Patrick Baillot and Alexis Ghyselen. Types for complexity of parallel computation in pi-calculus. In *Proceedings of ESOP 2021*, volume 12648 of *LNCS*, pages 59–86. Springer, 2021. https ://hal.archives-ouvertes.fr/hal-03126973.

[BG22] Patrick Baillot and Alexis Ghyselen. Types for complexity of parallel computation in pi-calculus. *ACM Trans. Program. Lang. Syst.*, 44(3) :15 :1–15 :50, 2022.

[BGK21] Patrick Baillot, Alexis Ghyselen, and Naoki Kobayashi. Sized types with usages for parallel complexity of pi-calculus processes. In *Proceedings of CONCUR 2021, International Conference on Concurrency Theory, CONCUR 2021*, volume 203 of *LIPIcs*, pages 34 :1–34 :22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[DY18] Romain Demangeon and Nobuko Yoshida. Causal computational complexity of distributed processes. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS 2018, pages 344–353. ACM, 2018.

[Ghy21] Alexis Ghyselen. *Sized Types Methods and their Applications to Complexity Analysis in Pi-Calculus.* PhD thesis, ENS de Lyon, September 2021.

[HMP13] Emmanuel Hainry, Jean-Yves Marion, and Romain Péchoux. Type-based complexity analysis for fork processes. In *Foundations of Software Science and Computation Structures - 16th International Conference, (FOSSACS 2013), Proceedings*, volume 7794 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2013.

[LM93] Daniel Leivant and Jean-Yves Marion. Lambda calculus characterizations of poly-time. *Fundam. Inform.*, 19(1/2), 1993.

[SW03] Davide Sangiorgi and David Walker. *The pi-calculus : a Theory of Mobile Processes.* Cambridge university press, 2003.