Confluence in active objects

Ludovic HENRIO – CNRS CASH/LIP
Yannick ZAKOWSKI – Inria CASH/LIP

2023-2024

1 Context

Confluence properties capture relaxed forms of determinism for rewriting systems. Intuitively, they capture that although two rewriting rules can be applied on the same system, diverging to two distinct states, it is always possible to converge back to the same state after a few more rewritings. Confluence properties are central in various settings, from traditional rewriting systems to the semantics of programming languages and logical systems. Here, we are interested in their use in the design of concurrent programming languages where they can capture conditions under which a concurrent program behaves observationally deterministic.

It is often useful to state confluence or partial confluence (i.e. confluence under some conditions) for a programming language [3]. This gives to the programmer the guarantee that a single behaviour can be observed and makes the execution of the program predictable, simplifying both reasoning and testing. Partial confluence may also be used to characterise a given execution in a minimal way [6].

This internship takes place in the context of previous formalisations of confluence properties for actor and active object languages [4, 5, 1], a popular programming model for concurrency. We seek to extend these early results to more complex settings, extending their applicability. Doing so however requires to adapt the underlying proof techniques, and to define precisely the dynamic conditions under which they hold, as well as static analyses soundly ensuring they hold.

In particular, we have identified as a promising lead to develop proof techniques suitable in this context : an old result due to De Bruijn [2] capturing the confluence of a rewriting system by layers, that we have recently mechanized in Coq. We wish to start from this state of art to explore further this space design.

2 Internship objective

The goal of this internship is to prove the confluence of programs written in a language based on active objects.

In particular, IFM [4] proves that when the (dynamic) graph of references between objects is a tree, then the computation is guaranteed to be confluent. A simpler observation is that a computation involving only pure objects (in a sense to be made precise) is also confluent.

However, relaxing the first result to allow more general graphs by introducing additional pure objects in a « core tree of impure objects » immediately appears challenging. One is tempted to build the proof of confluence of the global system by assuming the confluence of sub-systems made of sub parts of the graph, for instance only made of pure objects.

The candidate is expected to suggest a first conjecture for such a result of confluence and investigate whether De Bruijn's weak diamond property [2] could lead to an elegant layered proof.

Depending on the initial success of this first phase, and of the personal interest of the candidate, the internship may evolve into one or several of these aspects :

— finding incrementally more general classes of computations for which confluence holds ;
— designing static analyzes to characterize such classes ;
— formalizing the semantics of the language and the confluence results in the Coq proof assistant.

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / Inria – 69007 Lyon
E-mail : ludovic.henrio@cnrs.fr and yannick.zakowski@inria.fr

3 Internship

The internship will take place in the CASH Team, LIP lab, in Lyon France.

Candidate profile The candidate should ideally be familiar with (operational) semantics of programming languages. Previous knowledge of the Coq proof assistant is a plus if this aspect of the internship is to be considered, but not a requisite at all.

Références

[1] Denis Caromel and Ludovic Henrio. *A Theory of Distributed Objects*. Springer-Verlag, 2004.

[2] Jörg Endrullis and Jan Willem Klop. De Bruijn's weak diamond property revisited. *Indagationes Mathematicae*, 24(4) :1050–1072, 2013. ISSN 0019-3577. doi : https://doi.org/10.1016/j.indag.2013.08.005. URL https://www.sciencedirect.com/science/article/pii/S0019357713000657. In memory of N.G. (Dick) de Bruijn (1918–2012).

[3] Laure Gonnord, Ludovic Henrio, Lionel Morel, and Gabriel Radanne. A survey on parallelism and determinism. *ACM Computing Surveys*, 55(10) :1–28, feb 2023. doi : 10.1145/3564529. URL https://doi.org/10.1145%2F3564529.

[4] Ludovic Henrio, Einar Broch Johnsen, and Violet Ka I Pun. Active objects with deterministic behaviour. In Brijesh Dongol and Elena Troubitsyna, editors, *Integrated Formal Methods - 16th International Conference, IFM 2020, Lugano, Switzerland, November 16-20, 2020, Proceedings*, volume 12546 of *Lecture Notes in Computer Science*, pages 181–198. Springer, 2020. doi : 10.1007/978-3-030-63461-2_10.

[5] Marten Lohstroh and Edward A. Lee. Deterministic actors. In *2019 Forum for Specification and Design Languages, FDL 2019, Southampton, United Kingdom, September 2-4, 2019*, pages 1–8. IEEE, 2019. doi : 10.1109/FDL.2019.8876922. URL https://doi.org/10.1109/FDL.2019.8876922.

[6] Lars Tveito, Einar Broch Johnsen, and Rudolf Schlatte. Global reproducibility through local control for distributed active objects. In *Proc. 23rd International Conference on Fundamental Approaches to Software Engineering (FASE 2020)*, volume 12076 of *Lecture Notes in Computer Science*, pages 140–160. Springer, 2020.