Open Automata meet Session Types

Advisors: Rabéa Ameur-Boulifa[1] and Ludovic Henrio[2]

Place:
  (1) Eurecom – Sophia Antipolis
  or
  (2) Laboratoire de l'Informatique du Parallélisme (LIP) – ENS de Lyon

contact: rabea.ameur-boulifa@telecom-paris.fr   ludovic.henrio@ens-lyon.fr

Context

In the previous years, we have studied theoretical foundations for open systems and we defined open automata [5, 2, 4] that can be seen as labelled transition systems (LTSs) with parameters and holes. The transitions of open automata are much more complex than transitions of an LTS: they include guards expressing the relations between the parameters of the automaton with the actions of the holes, and assignments encoding their effects. We proved that our models have good properties, namely that composition preserves some properties like bisimulation.

The composition of open automata refers to filling holes, it requires to check the compatibility of their sorts. Nowadays, we simply verify the compatibility of two automata to be composed based on action labels and arity (comparing actions exposed by one LTS with actions expected to fill the hole). But, we are interested in enriching holes with description of behaviours; this description should look like a communication protocol. Precisely, the theory of session types and behavioural types [3] are intended to describe dynamic aspects of the behaviour of processes.

To tackle this issue, we ask: *Can we synthesise a global type from a collection of automata in order to ensure the composability of an open automaton with a compatible surrounding automaton?*

Objectives

The main objective of the internship is to deal with the characterization of interactions of an automaton with its environment. These interactions can be characterised as behavioural types. More specifically, the internship would follow the following steps:

- Familiarization with technical material: the semantics of open automata and the notion behavioural types [1] that allows the description of the dynamic aspects of processes. In particular the notion of session types that focus on process interactions should be studied.

- Design of an adequate characterisation of compatibility condition suitable for the composition open automata, and study the guarantees brought by this condition.

- Prove absence of deadlock upon composition.

- We have defined a notion of bisimulation between open automata. One additional objective of the internship could be to prove that behavioural types behave well relatively to bisimilarity.

- Design of examples demonstrating the benefit of this work.

The internship can be tackled only theoretically on paper, or formalised in an interactive theorem prover, or integrated with existing tools for open automata based on SMT solving.

References

[1] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luìs Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, Antònio Ravara, Emilio Tuosto, Hugo Torres Vieira, Gianluigi Zavattaro: Foundations of Session Types and Behavioural Contracts. ACM Comput. Surv. 49(1): 3:1-3:36 (2016)

[2] Ludovic Henrio, Eric Madelaine, Min Zhang: A Theory for the Composition of Concurrent Processes. In Albert, E., Lanese, I., eds.: 36th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE). Volume LNCS-9688 of Formal Techniques for Distributed Objects, Components, and Systems., Heraklion, Greece (June 2016) 175–194. `https://hal.inria.fr/hal-01299562`

[3] Alessandro Fantechi, Elie Najm, Jean-Bernard Stefani: From Behavioural Contracts to Session Types. Models, Languages, and Tools for Concurrent and Distributed Programming 2019: 278-297

[4] Zechen Hou, Eric Madelaine: Symbolic Bisimulation for Open and Parameterized Systems. In: PEPM 2020 - ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, New-Orleans, United States (January 2020). `https://hal.inria.fr/hal-02406098`

[5] Rabéa Ameur-Boulifa, Ludovic Henrio, Eric Madelaine: Compositional equivalences based on Open pNets. Journal of Logical and Algebraic Methods in Programming, 2022, 131, p.100842.. `https://hal.science/hal-03894031`