Short-Term Scientific Mission Grant
- APPLICATION FORM[1] -

Action number: CA20111

Applicant name: Luca Ciccone

Details of the STSM

Title: Mechanized Type Inference in the Linear π-Calculus

Start and end date: 22/05/2022 to 29/05/20222

Travel cost: ~400e

Accommodation cost: ~650e

Living cost: ~150e

Requested grant: 1200e

Goals of the STSM

The linear π-calculus is a core language of communicating processes in which (some) channels can be used for one-shot communications. The ability to create messages that contain channels makes it possible to encode structured communication protocols that can be arbitrarily long and may include branching points [1]. Since the complexity of protocols is reflected in the structure of types, the ability of inferring the type of (linear) channels from the structure of the processes that use them is of paramount importance. This STSM addresses the problem of mechanizing a type inference algorithm for the linear π-calculus. This problem has been partly addressed in previous works [2], although the existing algorithm is not entirely satisfactory for several reasons: (1) no mechanization of the algorithm and of its properties are available; (2) the algorithm can only analyze closed systems in which all the components of a system are fully specified; (3) the algorithm is based on a type system that does not guarantee deadlock and lock freedom. We aim to provide an Agda mechanization of a compositional type inference algorithm for the linear π-calculus based on a co-contextual type system that ensures deadlock and lock freedom for possibly recursive linear pi-calculus processes.

Working Plan

To tame the complexity of the mechanization, the work plan is structured in four subsequent phases so that each phase builds on the previous ones and focuses on a specific feature of the inference algorithm.

Compositional type inference

The first phase will be the identification of an appropriate formulation of the linear pi-calculus that is amenable to be mechanized and for which it is possible to define a compositional type inference algorithm based on a co-contextual type system. In traditional co-contextual type systems, type constraints are accumulated and solved when

1 This form is part of the application for a grant to visit a host organisation located in a different country than the country of affiliation. It is submitted to the COST Action MC via-e-COST. The Grant Awarding Coordinator coordinates the evaluation on behalf of the Action MC and informs the Grant Holder of the result of the evaluation for issuing the Grant Letter.

the whole program has been taken into account. The challenge is to be able to localize the solution of type constraints so that independent system components can be analyzed in isolation.

Support for recursive processes and types

Many communication protocols describe arbitrarily long conversations. The encoding of such protocols in the linear pi-calculus requires support for recursive types. In this phase we will extend the process language to include recursive processes, we will identify a suitable Agda representation of recursive (possibly infinite) types and will extend the inference algorithm accordingly. While it makes sense to find a finite representation for recursive processes, it is likely the case that the most convenient Agda representation of infinite types is based on coinductive data types. In this way, we can reuse much of the constraint resolution part of the inference algorithm defined in the previous phase.

Type inference with subtyping

Subtyping for session types has been shown to be important for broadening the range of processes that are well typed [4]. Previous works have shown that subtyping for session types is the relation induced by subtyping for channel types [1]. In this phase we will enrich the inference algorithm with support for subtyping. This will require a generalization in the representation of type constraints from equalities to subtyping relations and a corresponding modification of the constraint resolution algorithm with a type merging operator so that the type inference algorithm yields the most precise type of the channels used by a process.

Type inference with fair subtyping

The standard subtyping relation for session types [4] preserves safety properties (such as deadlock freedom) but not liveness properties (such as lock freedom). In this phase we will study a characterization of fair subtyping for types of the linear pi-calculus that preserves liveness properties. Existing characterizations of fair subtyping for session types are quite complex to define and particularly to mechanize [3]. We will investigate the origins of the complexity of the existing mechanization and whether it originates from theoretical design choices or from unsuitable choice of Agda elements for its mechanization. We will check the correctness of the new characterization in terms of soundness and completeness with respect to a reference semantic definition. Finally, we will adapt the constraint solving part of the inference algorithm so that it solves subtyping constraints using fair subtyping.

Expected outputs and contribution to the Action MoU objectives and deliverables.

Working Group: WG3 – Program Verification

At the end of the mission, we expect to obtain an Agda mechanization of a type inference algorithm for the linear pi calculus that is compositional (allowing for the analysis of independent processes in isolation) and that supports recursive types. We will also pave the way to the integration of subtyping in the inference algorithm and we expect to define a new characterization of fair subtyping that is easier to mechanize compared to previous ones. Overall, these efforts will contribute to the promotion of “the output of detailed, checkable proofs from automated theorem provers” in the specific context of communication-oriented program verification. We also expect that all the attempts in finding the right mechanization of the type inference algorithm will serve as a strong basis for the community concerning future related works. Hopefully, all such notions will be reusable into other proof assistants.

The STSM will also be the occasion to foster the collaboration between two different research groups that so far have only interacted indirectly with one another. Indeed, Ornela Dardha has been organizing the first two editions of the VErification of Session

Types (VEST) workshop. The workshop aims at gathering people from two different research communities, that working on session types and that working on proof assistants. The recent increase of interest in these tools highlighted the lack of a reference modus operandi on which all the community can agree. The same proof assistants are constantly being updated which means that sometimes existing mechanizations do not work anymore due to inconsistencies that are discovered at the tool level. However, both editions of the workshop have only been held online because of the spread of the COVID pandemic. In the meantime, our research groups have been making progress on related and partially overlapping topics [5,6] on the formalization of session-based calculi. In the end, I never had a chance to work personally and closely with people other than my own supervisor on topics related to my research interests. So, the STSM is in line with the capacity building objectives of EuroProofNet, by bringing “together members of the different communities working on proofs in Europe” and by “actively supporting young researchers”. We expect that the collaboration between the involved research groups will continue even after the mission has ended by inspecting novel research topics such as type inference involving coinductive types, mechanization of a type system using the properties that we studied and so forth.

References

1. Ornela Dardha, Elena Giachino, Davide Sangiorgi: **Session types revisited**. Inf. Comput. 256: 253-286 (2017)
2. Luca Padovani: **Type Reconstruction for the Linear π-Calculus with Composite Regular Types**. Log. Methods Comput. Sci. 11(4) (2015)
3. Luca Ciccone, Luca Padovani: **Inference Systems with Corules for Fair Subtyping and Liveness Properties of Binary Session Types**. ICALP 2021: 125:1-125:16
4. Simon J. Gay, Malcolm Hole: **Subtyping for session types in the pi calculus**. Acta Informatica 42(2-3): 191-225 (2005)
5. Uma Zalakain, Ornela Dardha: **π with Leftovers: A Mechanisation in Agda**. FORTE 2021: 157-174
6. Luca Ciccone, Luca Padovani: **A Dependently Typed Linear π-Calculus in Agda**. PPDP 2020: 8:1-8:14