The Open Challenge of Typed Expressiveness in Concurrency

Jorge A. Pérez
University of Groningen, The Netherlands
`http://www.jperez.nl`

Context *Communication* and *types* are increasingly relevant in (concurrent) programming. To bear witness of this trend, several languages promoted by industry offer advanced type systems (or type-based analysis tools) and/or support (message-passing) communication. For instance, Facebook's Flow [1] is a type checker for JavaScript based on gradual typing; Mozilla's Rust [4] exploits affine, ownership types to balance safety and control; Google's Go [3] supports process concurrency and channel-based communication. Other languages (e.g., Erlang [2]) also offer forms of (message-passing) communication.

If communication and types are here to stay, on what foundations languages integrating both features should rest? Much research within formal techniques in distributed systems has been devoted to models for concurrency and communication. In particular, *process calculi* have been widely promoted as a basis for type systems for concurrent programs. Indeed, building upon the π-calculus, a variety of *behavioral type systems* have been put forward [5, 6]: by classifying behaviors (rather than values), these type structures abstract structured protocols and enforce disciplined message-passing programs. Existing work suggests that rather than a shortage of foundations for types and communication, we have the opposite problem: there are many formal foundations and it is unclear how to build upon them to transfer analysis techniques into practice.

The Challenge The current situation calls for rigorous comparisons between well-established (but distinct) behavioral typed frameworks. Besides revealing bridges between different models of typed processes, such comparisons should clarify the complementarities/shortcomings of analysis techniques based on types. Developing a theory of *typed expressiveness* is thus a challenge for the specification and analysis of distributed systems. The consolidation of *communication-centered* software systems (collections of interacting, heterogeneous services) and the renewed interest of software practitioners in communication and types endow this challenge with practical significance.

We argue that the much needed formal comparisons may draw inspiration from results and frameworks for *relative expressiveness*, as studied in concurrency theory. This area has elucidated important formal relations between *un-*

typed process languages (see [7] for a survey); it thus appears as an adequate basis for analogous formal results in a typed setting.

This Presentation Building upon the recent position paper [8], this presentation first overviews main achievements in untyped expressiveness. Then, it briefly reviews expressiveness results that consider behavioral types and/or behaviorally typed processes. It concludes by discussing promising research directions.

References

[1] Flow: A Static Type Checker for JavaScript, `http://flowtype.org`

[2] The Erlang Programming Language, `http://www.erlang.org`

[3] The Go Programming Language, `https://golang.org`

[4] The Rust Programming Language, `https://www.rust-lang.org`

[5] Dezani-Ciancaglini, M., de'Liguoro, U.: Sessions and session types: An overview. In: WS-FM 2009. LNCS, vol. 6194, pp. 1–28. Springer (2010)

[6] Huttel, H., Lanese, I., Vasconcelos, V., Caires, L., Carbone, M., Deniélou, P.M., Mostrous, D., Padovani, L., Ravara, A., Tuosto, E., Vieira, H.T., Zavattaro, G.: Foundations of session types and behavioural contracts. ACM Computing Surveys (2016), to appear

[7] Parrow, J.: Expressiveness of process algebras. ENTCS 209, 173–186 (2008), `http://dx.doi.org/10.1016/j.entcs.2008.04.011`

[8] Pérez, J.A.: The challenge of typed expressiveness in concurrency. In: Proc. of FORTE 2016. LNCS, vol. 9688, pp. 239–247. Springer (2016)