Internship subject (Master 2)

Advanced study of dataflow explicit futures

Main advisor: Ludovic Henrio

Co-advisors: Matthieu Moy and Christophe Alias

Place: Laboratoire de l'Informatique du Parallélisme (LIP)
École Normale Supérieure de Lyon

Context

A future is a place-holder for a value being computed, and we generally say that a future is resolved when the associated value is computed. In existing languages futures are either implicit, if there is no syntactic or typing distinction between futures and non-future values, or explicit when futures are typed by a parametric type and dedicated functions exist for manipulating futures. We defined in [1] a new form of future, named data-flow explicit futures, with specific typing rules that do not use classical parametric types. The new futures allow at the same time code reuse and the possibility for recursive functions to return futures like with implicit futures, and let the programmer declare which values are futures and where synchronisation occurs, like with explicit futures. We prove that the obtained programming model is as expressive as implicit futures but exhibits a different behaviour compared to explicit futures.

Actors and active object languages [2, 3, 4] are based on asynchronous communications between mono-threaded entities and massively use futures to represent replies to asynchronous messages. We illustrate our proposal on a active object languages but the approach is generalisable to other languages using futures.

Objectives

The research report [1] describes a type system and a semantics for dataflow explicit futures, the first objective is to provide an implementation for them:

- either as a modification of the ABS [2, 3] language and its type system. This is not too difficult because the ABS framework is not too big and provides the right entry points
- or as a library: in that case the objective would be to design the object constructs that enforce the same constraints as the type system. At runtime, the synchronisation patterns could reuse the dataflow synchronisation of the ProActive library [4], which would allow for distributed execution.

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / INRIA – 69007 Lyon
Contact E-mail : Christophe.Alias@ens-lyon.fr et Matthieu.Moy@univ-lyon1.fr

From this point, several independent directions can be envisioned:

- explore the usefulness of the newly implemented construct by implementing new examples or revisiting the programs existing in the chosen implementation context
- experiment on the efficiency of the construct and design optimisations for the execution of programs with dataflow explicit futures. Depending on the implementation solution, different optimisations will be possible, possibly inspired by existing results.
- prove the correctness of the optimisations based on (extensions of) the existing semantics
- Formalise the language and prove existing results or the correctness of optimisations in a theorem prover.
- study formally or in practice the coexistence of dataflow explicit futures and classical explicit futures in the same programming language

The internship subject is quite flexible and can focus either on theoretical aspects or implementation and software engineering contributions. Depending on the orientation of the internship, the initial implementation phase could be realised more or less thoroughly before considering the next steps.

References

[1] "Data-flow Explicit Futures" – Ludovic Henrio. `https://hal.archives-ouvertes.fr/hal-01758734`

[2] The Abstract Behavioral Specification Language: A Tutorial Introduction. Reiner Hähnle. FMCO 2012.

[3] `http://docs.abs-models.org/`

[4] `https://team.inria.fr/scale/software/proactive/`