Liquid types for stream processing functions

Jean-Pierre.Talpin@inria.fr

A cyber-physical system is an entity of heterogeneous constituents: software, embedded in hardware, interfaced with the physical world. Time takes different forms when observed from each of these viewpoints: it is continuous in physics or discrete, event-based in software, time-triggered in hardware. Moreover, modelling and programming paradigms used to represent time in software (synchronous), hardware (RTL, TLM) or physics (ODEs) significantly alter the perception of time. On top of that, heterogeneous timing constraints need not only be mitigated across system components, but so do all relations in time of its other quantas and metrics: speed, frequency, size, throughput, volume, pressure, capacity, heat, angle, \dots

In light of recent advances in type theory and formal verification, with the definition of decidable classes of value-dependant type systems and the development of SAT/SMT solving functionalities in theorem provers, and taking advantage of past experiences applying these concepts to concurrent reactive languages, our project is to define a simple data-flow language equipped with a strongly type inference system and capable of representing and analysing, not just abstract notions of time called clocks in synchronous languages, but every logical and quantitative aspect of the program behaviour and its communication interfaces with the physical execution environment.

Our goal is to design and apply a type-based program analysis framework to, first, formally verify programs to guarantee global safety properties (e.g. determinism or constructivity, as in synchronous language), but, more generally, to implement a type-based translation validation and requirement engineering methodology, using 1/ inferred type properties to trace and validate compilation steps, and 2/ user-specified type annotations to trace and validate requirements.

The thesis program will consist of both the elaboration of sound theoretical foundation suitable to the envisioned language and the implementation of a working prototype front-end, analyser and verifier. A prototype code generator will be developed by using the infrastructure of the Eclipse project POP (Polychrony on Polarsys). It is therefore suitable to a highly motivated student with a demonstrated solid background in theoretical computer science, and, more specifically, programming languages, compilation, type theory, formal verification, logic. Knowledge and internship experience in either of the above topics are pluses as well as a fluent spoken and written english and environments like Haskell, Coq, OCaml.

REFERENCES

“Liquid types for stream-processing functions”. J.-P. Talpin, et al. Draft manuscript, 2015.

"The type and effect discipline". Talpin, J.-P., Jouvelot, P. Conference on Logic in Computer Science (LICS'92). IEEE Press, June 1992. ACM-IEEE Test-of-Time Award 2012.

"Implementation of the typed lambda calculus using a stack of regions". Tofte, M., Talpin, J.-P. Symposium on Principles of Programming Languages (POPL'94). ACM Press, January 1994. ACM SIGPLAN most influential POPL paper Award 2004.