Cyclo-static data-flow network analysis for SC Java task sets.

Jean-Pierre.Talpin@inria.fr

Anyone experienced with multi-threaded programming would acknowledge the difficulty of designing and implementing such concurrent software. Resolving concurrency, synchronisation, causality and coordination issues, tackling the non-determinism germane in multi-threaded software is extremely difficult. Ensuring correctness with respect to the specification and deterministic behaviour is necessary for safe execution of such code. It is therefore desirable to synthesise multi-threaded code from formal specifications using a provably `correct-by-construction' approach.

The proposed PhD program starts from previous results pertaining to the abstract affine scheduling of Java tasks connected by specified UCSDF graph (ultimately periodic cyclo-static data-flow graph) by the automated synthesis of real-time schedulers (i.e. code generation of the main init method).

We wish to push forward on this topics by developing the necessary program analysis that would allow to infer that UCSDF graph from a given SCJ (SCJ: safety-critical Java) task set, namely, the quantitative analysis of communication and computation costs of individual tasks in a way that refines an initial SDF structure (possibly constructed from the input-output communication structure of the original task set) into the UCSDF graph rendering the quantitative communication and computation patterns to be scheduled.

The thesis program will consist of both the elaboration of a theoretical framework, from understanding existing techniques to representing communicating Java tasks using automata, elaborating abstractions of these automata into an adequate model of data-flow graphs and/or affine relations, to finally reuse and possibly extend previous results on affine abstract scheduling.

The thesis shall comprise the experimental validation of a proof of concepts through a working prototype. Fluent French and English, as well as in-depth, Master-level, knowledge in theoretical computer-science: program analysis and compilation, abstract interpretation, concurrency theory, scheduling theory, is mandatory. Knowledge of concurrent programming in Java (RTJ/SCJ) is a plus.

REFERENCES

"Timed behavioural modelling and affine scheduling of embedded software architectures in the AADL using Polychrony". L. Besnard, A. Bouakaz, T. Gautier, P. Le Guernic, Y. Ma, J.-P. Talpin, H. Yu. In Science of Computer Programming. Elsevier, 2014

"Design of Safety-Critical Java Level 1 Applications Using Affine Abstract Clocks". A. Bouakaz and J.-P. Talpin. International Workshop on Software and Compilers for Embedded Systems (M-SCOPES'13). ACM, June 2013.

"Buffer minimization in earliest-deadline first scheduling of dataflow graphs". A.

Bouakaz and J.-P. Talpin. Conference on Languages, Compilers and Tools for Embedded Systems (LCTES'13). ACM, June 2013