Abstract:
Software designs involve different notations and different views which have to be integrated consistently in a system implementation. We first outline an algebraic heterogeneous software modelling approach exemplified by the UML: The semantics of partial models are captured by institutions, consistency conditions are represented by institution co-morphisms. On the programming level, we then consider the Single-Instruction Multiple Threads (SIMT) model of execution used on GPUs which can be combined with CPU-based programs for highly parallel applications. We show that SIMT computations are correct with respect to an interleaving multi-thread semantics, but also demonstrate the unfairness of this model of execution. If time permits, we finally discuss an abstract scheme of specifying software components and their compatibility.