Aguado, Joaquín; Mendler, Michael; von Hanxleden, Reinhard; Fuhrmann, Insa
Denotational fixed-point semantics for constructive scheduling of synchronous concurrency.
(English) Zbl 1325.68135
Acta Inf. 52, No. 4-5, 393-442 (2015).

Summary: The synchronous model of concurrent computation (SMoCC) is well established for programming languages in the domain of safety-critical reactive and embedded systems. Translated into mainstream C/Java programming, the SMoCC corresponds to a cyclic execution model in which concurrent threads are synchronised on a logical clock that cuts system computation into a sequence of macro-steps. A causality analysis verifies the existence of a schedule on memory accesses to ensure each macro-step is deadlock-free and determinate. We introduce an abstract semantic domain $I(D, P)$ and an associated denotational fixed-point semantics for reasoning about concurrent and sequential variable accesses within a synchronous cycle-based model of computation. We use this domain for a new and extended behavioural definition of Berry’s causality analysis in terms of approximation intervals. The domain $I(D, P)$ extends the domain $I(D)$ from our previous work and fixes a mistake in the treatment of initialisations. Based on this fixed-point semantics we propose the notion of Input Berry-constructiveness (IBC) for synchronous programs. This new IBC class lies properly between strong (SBC) and normal Berry-constructiveness (BC) defined in previous work. SBC and BC are two ways to interpret the standard constructive semantics of synchronous programming, as exemplified by imperative SMoCC languages such as Esterel or Quartz. SBC is often too restrictive as it requires all variables to be initialised by the program. BC can be too permissive because it initialises all variables to a fixed value, by default. Where the initialisation happens through the memory, e.g., when carrying values from one synchronous tick to the next, then IBC is more appropriate. IBC links two levels of execution, the macro-step level and the micro-step level. We prove that the denotational fixed-point analysis for IBC, and hence Berry’s causality analysis, is sound with respect to operational micro-level scheduling. The denotational model can thus be viewed as a compositional presentation of a synchronous scheduling strategy that ensures reactivity and determinacy for imperative concurrent programming.

MSC:

68Q55 Semantics in the theory of computing
68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)
68Q85 Models and methods for concurrent and distributed computing (process algebras, bisimulation, transition nets, etc.)

Software:
StateFlow; SCCharts; Esterel; Quartz

Full Text: DOI

References:


© 2022 FIZ Karlsruhe GmbH

Haliswachs, N; Caspi, P; Raymond, P; Pilaud, D. The synchronous data-flow programming language LUSTRE, Proc. IEEE, 79, 1305-1320, (1991) · doi:10.1109/5.97400


Motika, C., von Hau validated deterministic reactive systems with Synchronous Java (invited paper). In: Proceedings of the 9th Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS
[80] Schneider, K.: The synchronous programming language Quartz. Internal Report 375, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany (2009)

This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.