Parallel Object-Oriented Specification Language

Oana Florescu Jeroen Voeten Bart Theelen Marc Geilen
Henk Corporaal

1 Introduction

The Parallel Object-Oriented Specification Language (POOSL) is an expressive modelling language for hardware/software systems [10]. It was originally defined in [7] as an object-oriented extension of process algebra CCS [6], supporting (conditional) synchronous message passing between (hierarchically structured) asynchronous concurrent processes. Meanwhile, POOSL has been extended with real-time [2] and probabilities [1] to evolve into a powerful general-purpose modelling language accompanied with simulation, analysis and synthesis techniques that scale to large industrial design problems.

2 Overview

Figure 1 illustrates the key role of the formal semantics underlying any POOSL model. The formal semantics defines a timed probabilistic labeled transition system [1] capturing both non-deterministic and probabilistic choices between alternative actions such as communicating messages or processing data as well as (deterministic) progress in time. This semantical model allows to exhaustively check correctness and performance properties like absence of deadlock and throughput [10] by expressing such properties in formalisms like MTL [5] or temporal rewards [11]. Because of state-space explosion problems, these techniques do not scale well to industrial-sized problems. Therefore, simulation-based evaluation is offered as an alternative in which case the analysis is based on a trace constructed from the transition system [2, 8]. This is done by representing different process behaviours as Process Execution Trees (PETs) [1]. Each PET informs a central scheduler about its possible transitions. Then the scheduler matches and selects one of the alternatives, after which the PETs change their behaviour accordingly. This technique is also used for efficient and interactive model simulation. In addition to model analysis technique, POOSL is equipped with synthesis techniques that allow models to be mapped onto an execution platform in a property-preserving way [3]. This technique generates a trace from the transition system in real-time by synchronizing the virtual (model) time with the physical time of the execution platform. The trace as observed in model time has a (slight) time difference with the trace as observed in physical time and this distance determines the extend to which real-time model properties are preserved in the implementation [12].

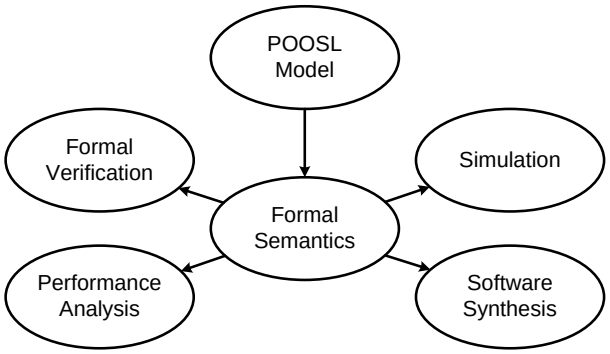

Figure 1: Simulation, analysis and synthesis with POOSL

POOSL has been applied in various academic and industrial case studies in different application domains. Examples include, but are not limited to, internet routers, packet switches, communication protocols, network processors, networks on chip, television applications, copiers and wafer scanners [10]. The accompanying software tools allowed the effective analysis of systems comprised of a million concurrent processes, demonstrating the scalability of the approach towards systems of industrial complexity.

3 Future and Links

The most important direction of future research is the development of design flows in which the different techniques are integrated. The idea is to start from abstract executable models, which are refined and synthesized in a property-preserving way, quantitatively guided by performance analysis and design-space exploration techniques. The first results in this direction are found in [4, 9].

References

[1] Bokhoven, L.v.: Constructive Tool Design for Formal Languages: From Semantics to Executing Models. Ph.D. thesis, Eindhoven University of Technology (2002)

[2] Geilen, M.: Formal Techniques for Verification of Complex Real-Time Systems. Ph.D. thesis, Eindhoven University of Technology (2002)

[3] Huang, J.: Predictability in Real-Time System Design. Ph.D. thesis, Eindhoven University of Technology (2005)

[4] Florescu, O.: Predictable Design for Real-Time Systems. Ph.D. thesis, Eindhoven University of Technology (2007)

[5] Koymans, R.: Specifying Real-Time Properties with Metric Temporal Logic. Real-Time Systems **2**(4), 255–299 (1990)

[6] Milner, R.: Communication and Concurrency. Prentice-Hall (1989)

[7] Putten, P.v.d., Voeten, J.: Specification of Reactive Hardware/Software Systems. Ph.D. thesis, Eindhoven University of Technology (1997)

[8] Theelen, B.: Performance Modelling for System-Level Design. Ph.D. thesis, Eindhoven University of Technology (2004)

[9] Huang, J., Voeten, J. Groothuis, M., Broenink, M., Corporaal, H.: A model-driven design approach for mechatronic systems. In: Proceedings of ACSD'07, pp. 127–136 (2008)

[10] Theelen, B., Florescu, O., Geilen, M., Huang, J., van der Putten, P., Voeten, J.: Software/Hardware Engineering with the Parallel Object-Oriented Specification Language. In: Proceedings of MEMOCODE'07, p. 139 (2007)

[11] Voeten, J.: Performance Evaluation with Temporal Rewards. Performance Evaluation **50**(2/3), 189–218 (2002)

[12] Huang, J., Voeten, J., Corporaal H.: Predictable real-time software synthesis. Real-time systems 36, 159–198 (2007)