Title: Model Checking for open concurrent systems.

Context: We have developed a new semantic framework for the analysis of composition operators in concurrent systems. Open pNets (parameterized networks of synchronized automata) are new semantic objects used to define in a symbolic way the semantics of composition operators, but also of various type of "open" programs, like parallel and distributed programming skeletons, or generic distributed algorithms. We define the operational semantics of open pNets, using "open transitions", and "open automata" that include symbolic hypotheses on the behavior of the pNets "holes". Data in pNets and in open transitions is handled symbolically, using logical predicates, and reasoning on pNets behaviours is done by using an SMT solver dealing with these predicates.

One important tool for the analysis of concurrent systems is Model Checking (MC), that is checking whether some property expressed as a temporal logic formula is valid on the states of a (finite) model of its behaviour. Building a Model Checker for pNets (more precisely for open automata) has a challenging perspective to build a finite algorithm for checking properties of systems involving unbounded data. Typical MC algorithms are traversing the model (together with the formula), matching each action in the model transitions with "actions predicates" in the formula. For open transitions, matching could be (naively) replaced by checking the satisfiability of a formula, using a Satisfiablity Modulo Theories (SMT) solver. The main difficulties with this approach is that a positive reply from an SMT solver gives an instantiation of the predicate variables validating the formula, not all possible instantiations, and that each instantiation would correspond to a specific unfolding of the automaton for the next step.

Subject: after familiarization with the pNet framework and with classical MC algorithms, the objective of the work is to answer some of the following questions:

- is it possible to define a (bounded) MC algorithm using satisfiability on individual open transitions, through a finite "unfolding" of the open automaton based on the results provided by the SMT solver?

- is it possible, based on the predicates in the transitions, to construct an abstract unfolding (that may be finite in some cases)?

- is it possible to delay the satisfiability checking, simply assembling the predicates along the way, and submit it only at the end? (hint: for this we may need to define some restrictions on the structure of the pNets to get termination) ?

- is it possible to achieve one of the above compositionally, which is by working with sub-components of an open pNet, instead of using the composed open automaton?

and (eventually) to define / implement a prototype.

Advisor: Eric Madelaine, INRIA Sophia-Antipolis, http://www-sop.inria.fr/members/Eric.Madelaine/
Email: eric.madelaine@inria.fr
Co-advisor: Simon Bliudze, INRIA Lille – Nord Europe, http://www.bliudze.me/simon/,
Email: simon.bliutze@inria.fr
Place of work : INRIA Sophia-Antipolis

This document: http://www-sop.inria.fr/members/Eric.Madelaine/stages2017/MC-pNets.pdf

More détails :

Background :

Originally the ***(closed) pNet model*** had been defined to give a hierarchical model of data-sensitive, parameterized systems, and serve as a core format for specification and verification tools for (closed) distributed systems. It has been used to give a behavioural semantics to various process algebras, calculi, languages and libraries for concurrent and distributed systems.

Open pNets is an extension of this model, with the long term goal to offer a novel compositional approach allowing for the mastering of state explosion, by proving properties directly on contexts (operators, process expressions, program skeletons, …). Later systems can be build by composing or instantiating such contexts, preserving their properties.

We have shown that this model can be used to encode synchronization operators of many different types, including (value-passing) CCS, Lotos, Active objects, distributed components, etc.

Open automata (and thus pNets) are endowed with a notion of (strong) bisimulation, that has been proved compositional, meaning preserved by instantiation of pNets holes.

A closely related approach with very similar goals is that of "BIP architecture patterns", though they have different characteristics and use very different techniques. There is some work ongoing to relate them, and this work will be a good source of inspiration for this master subject.

Refs:

1) hal-01432917v1 (on open pNets):
Ludovic Henrio, Eric Madelaine, Min Zhang. A Theory for the Composition of Concurrent Processes
Elvira Albert; Ivan Lanese. 36th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE), Jun 2016, Heraklion, Greece. Lecture Notes in Computer Science, LNCS-9688, pp.175-194, 2016, Formal Techniques for Distributed Objects, Components, and Systems. 〈10.1007/978-3-319-39570-8_12〉

Extended version: RR-8898, INRIA Sophia Antipolis - I3S. 2016, pp.23. https://hal.inria.fr/INRIA/hal-01271684v1

2) hal-01139432v1 (on expressiveness of closed and open pNets):
Ludovic Henrio, Eric Madelaine, Min Zhang. pNets: an Expressive Model for Parameterised Networks of Processes
Formal Approaches to Parallel and Distributed Systems (4PAD)-Special Session of Parallel, Distributed and network-based Processing (PDP), 2015, Turku, Finland

3) On VerCors, verification platform based on closed pNets:

Ludovic Henrio, Oleksandra Kulankhina, Siqi Li, Eric Madelaine. **Integrated environment for verifying and running distributed components.** *Fundamental Approaches to Software Engineering*, Apr 2016, Eindhoven, Netherlands. Springer, LNCS, 9633, pp.66-83, 2016. hal-01303557v1

Extended version RR-8841. hal-01252323v1

4) On BIP architecture patterns:

A. Mavridou, E. Stachtiari, S. Bliudze, A. Ivanov, P. Katsaros, and J. Sifakis, "Architecture-based Design: A Satellite On-board Software Case Study," 2016.