

Modularity and Petri Nets

Laure Petrucci

LIPN, CNRS UMR 7030, Université Paris XIII
99, avenue Jean-Baptiste Clément
F-93430 Villetaneuse, FRANCE
petrucci@lipn.univ-paris13.fr

Abstract. The systems to model are nowadays very large. Their specification is often decomposed into several steps. This leads to modularly or incrementally designed models. Petri nets analysis is generally achieved via state space analysis, which is often impossible to perform due to the so-called *state space explosion problem*. Several methods allow to reduce the occurrence graph size, e.g. using partial orders, symmetries, . . . Here, we focus on techniques which take advantage of the modular design of the system, and hence builds the state space in a modular or incremental way.

Introduction

Nowadays, systems become larger and larger. In order to cope with a large model, high-level Petri nets have been designed, where data is carried by tokens and manipulated by transitions, while the Petri net structure describes the information flow. However, the use of such high-level models proves not to be sufficient. When designing a system, the modeller often decomposes it into different parts with separate functionalities, e.g. a sender and a receiver in a protocol. This natural decomposition in the design can also be used in the model, leading to a modular Petri net. It possesses the additional advantage of designing an entity once and eventually reusing it when several copies occur in the system, e.g. several receivers behaving exactly in the same way.

A desirable verification issue would then be to check the properties of the modules separately and infer those of the whole system. Unfortunately, this often does not work or only for undesired properties. Hence, verification based on a modular approach is not straightforward and led to several research trends.

Several techniques have been proposed to push further the use of modularity for verification purposes. They allow to compute the invariants of the whole system from those of the modules [CP92,CP00] or give compact and/or modular representation of the occurrence graph, thus avoiding the state space explosion problem.

Here, we will focus on different compositional/modular techniques for state space construction.

Modular State Spaces

Let us consider a modular Petri net in which the modules are connected via *synchronised transitions*. The *modular state space* was introduced in [CP95,CP00], further refined and implemented in [LP04].

Instead of having a single flat occurrence graph, which may quickly become quite large, the modular state space is composed of several graphs. Each module has its own state space which reflects only local behaviour: the states are restricted to the places within the module, while the arcs captures the firing of local transitions only, i.e. transitions of the module which are not synchronised with transitions of other modules. In addition to these local state spaces, a *synchronisation graph* describes the interactions between the individual modules. The arcs provide information on the firing of synchronised transitions only, and the nodes are labelled by a product of strongly connected components of the modules' state spaces. Hence, interleavings are avoided, and the representation can be quite compact.

Experiments have been conducted in [LP04,Pet05] showing that when modules are rather loosely coupled, the modular state space is small (and thus amenable) compared to the flat occurrence graph. This is the case for models of flexible manufacturing systems, workflows, ... encountered in practice.

A drawback could be that verifying properties requires traversing several graphs, collecting the information scattered in different structures. However, [CP00,LP04] provide algorithms to check the classical Petri net properties (boundedness, deadlocks, liveness, ...), and [LM04] addresses the verification of LTL properties and its implementation within the MARIA tool [Mar].

Compositional Verification

Complementary techniques have been introduced in [Kla03]. Let us assume we have a Petri net and a LTL\X formula it should satisfy. [Kla03] constructs an *observation graph* where the transitions that appear in the formula are represented while the other ones are not. The states thus represent sets of states that can be reached from another one by firing unobservable transitions only.

This approach is property-dependant as an observation graph must be built for each formula where different transitions are concerned. It is basically the same idea as the one mentioned in [LP04]: the Petri net to be analysed can be regarded as a modular Petri net where the synchronisation transitions are those of interest, i.e. the observable ones. Then the synchronisation graph of [LP04] and the observation graph of [Kla03] coincide.

[Kla03] also proposes sufficient conditions for deducing properties of the whole system from those of its subsystems. The subsystems are not completely considered in isolation: the other ones are abstracted away by *abstraction places*. Thus, when analysing a module, information about its environment behaviour is present as well. The abstraction places are determined through invariants computation for the complete system.

Incremental Verification

The design of a system is often incremental: it is first done at an abstract level, and then parts of it are refined. A similar approach to state space construction is presented in [LL01]. Three types of refinement are used: *type*, *subnet* and *node refinements*. The first one consists in extending type definitions, the second in adding a subnet to the abstract one, and the third in replacing a place (or transition) by a place-bordered (transition-bordered) subnet satisfying some flow criteria. The construction of the state space exploits those of the subnets and abstract net. According to the sort of refinement used, some of the nodes and arcs in these state spaces are reused, thus avoiding additional computation, or, for node refinement, the technique is similar to modular state spaces. Local state spaces are built for the refining nets while a global state space captures the markings of the other places and points onto the local state spaces, as the synchronisation graph of modular state spaces does.

References

- [CP92] S. Christensen and L. Petrucci. Towards a modular analysis of coloured Petri nets. In *Proc. 13th Int. Conf. Application and Theory of Petri Nets (ICATPN'92), Sheffield, UK, June 1992*, volume 616 of *Lecture Notes in Computer Science*, pages 113–133. Springer, 1992.
- [CP95] S. Christensen and L. Petrucci. Modular state space analysis of coloured Petri nets. In *Proc. 16th Int. Conf. Application and Theory of Petri Nets (ICATPN'95), Turin, Italy, June 1995*, volume 935 of *Lecture Notes in Computer Science*, pages 201–217. Springer, 1995.
- [CP00] S. Christensen and L. Petrucci. Modular analysis of Petri nets. *The Computer Journal*, 43(3):224–242, 2000.
- [Kla03] K. Klai. *Réseaux de Petri : vérification modulaire et symbolique*. PhD thesis, University Paris 6, France, 2003.
- [LL01] C. Lakos and G. Lewis. Incremental state space construction of coloured Petri nets. In *Proc. 22nd Int. Conf. Application and Theory of Petri Nets (ICATPN'01), Newcastle, UK, June 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 263–282. Springer, 2001.
- [LM04] T. Latvala and M. Mäkelä. LTL model-checking for modular Petri nets. In *Proc. 25th Int. Conf. Application and Theory of Petri Nets (ICATPN'04), Bologna, Italy, June 2004*, pages 298–311, June 2004.
- [LP04] C. Lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *Proc. 4th Int. Conf. on Application of Concurrency to System Design (ACSD'04), Hamilton, Canada, June 2004*, pages 185–194. IEEE Comp. Soc. Press, June 2004.
- [Mar] MARIA: the modular reachability analyser. <http://www.tcs.hut.fi/Software/maria/index.html>.
- [Pet05] L. Petrucci. Cover picture story: Experiments with modular state spaces. *Petri Net Newsletter*, 2005. To appear.