New solvers for asymmetric systems in GreatSPN

S. Baarir, M. Beccuti, G. Franceschinis *
Dip. di Informatica, Univ. del Piemonte Orientale
15100 Alessandria, Italy.
{baarir,beccuti,giuliana}@mfn.unipmn.it

1 Introduction

The Well-formed Net (WN) formalism and its stochastic extension (SWN) [3] takes advantage from the system symmetries to cope with *state space explosion problem*, and hence provide efficient solution techniques for both qualitative and quantitative analyzes.

Based on these symmetries, a quotient graph, called *Symbolic Reachability Graph (SRG)* [3], is automatically constructed from an (S)WN. Each node of the SRG, called Symbolic Marking (SM), represents a set of ordinary markings. In case of *highly symmetric systems*, the expected size of the SRG is exponentially reduced with respect to the one of the ordinary Reachability Graph (RG). However, it is less effective, in case of partially symmetric systems, i.e., systems with mostly symmetric behavior and occasional locally asymmetric behavior. In fact, in the SRG approach asymmetries are always taken into account, even if the asymmetric behavior of the system is local.

To cope with this limitation, Haddad & all [6] have proposed a more compact structure, called *Extended SRG (ESRG)*. The idea is to group into Extended SMs (ESM) sets of (partially) similar SMs. In this representation, and in case of *locally symmetric behavior*, the set of SMs captured in an ESM are kept implicit and represented by a unique symbolic Symmetric Representation (SR).

Unfortunately, the derivation of a CTMC from the ESRG is not straightforward: in the general case, the SMs aggregation suggested in the ESRG approach does not satisfy the (strong or exact) lumpability condition as on the SRG. Hence, the ESRG structure have to be refined according to the desired lumpability condition. In [5] an efficient refinement algorithm is proposed. It is based on the Paige and Tarjan partition refinement algorithm and exploits the information contained in the ESRG.

Another approach, called *Dynamic SRG (DSRG)*, was proposed in [1]. It relies on a separate representation of the system asymmetries in a so called *control automaton*. The DSRG is then obtained by synchronizing the transitions of a (symmetric) SWN with the corresponding control automaton. It is worth noting that DSRG satisfies the exact lumpability condition by construction, so that it does not need further refinement.

In this paper we will present the ESRG/DSRG framework to model and solve (asymmetric) SWN models. This framework combines several tools: *GreatSPN* [4] for the model design, *WNESRG* to build the ESRG of the designed model, *ESRG2MC* to refine the ESRG and generate the corresponding MC, *WNDSRG* to build the DSRG and the corresponding MC. *MCSolver* is used to solve the MC and compute the steady state marking probability. The following section is dedicated to the detailed presentation of this new framework.

2 Framework architecture

The architecture of the framework is depicted in Fig. 1 where the framework components are presented by rectangles, the component invocations are shown as solid arrows, and models/data exchanges are represented by dotted arrows.

GreatSPN is used as graphical interface and solution manager. It allows the design of the SWN model and activates the solution process. The solution manager executes in the correct order the framework components, and manages the models/data exchanges between them. The solution process comprises three steps:
1. *WNESRG/WNDSRG* computes the ESRG/DSRG from an SWN model. For the DSRG, we have to highlight that the asymmetries of the model are encoded in an external file produced directly by the user with a text editor (see Fig.1), since at the moment in GreatSPN we cannot directly represent the control automaton.
2. *ESRG2MC* refines the ESRG with respect to the exact or strong lumpability condition and generates the corresponding CTMC (called RESRG in Fig.1).
3. $MCSolver$ solves the MC, from the refined ESRG or

*The work of these authors was partially supported by Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), under the EU CRUTIAL project.

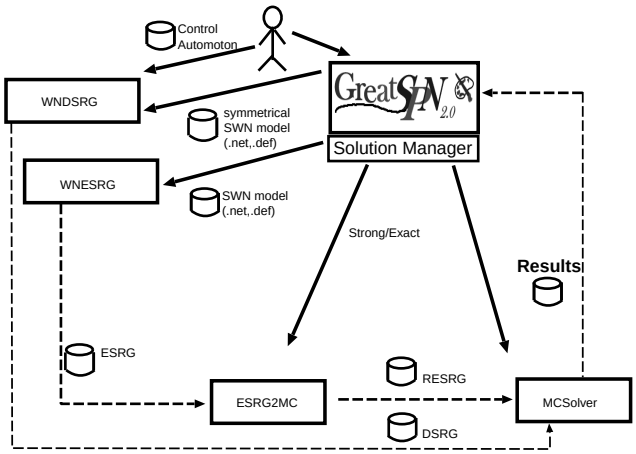

Figure 1. ESRG/DSRG Framework architecture.

from the DSRG, by computing the steady state marking probabilities. The current version of $MCSolver$ allows also to compute some performance indices: for example, the throughput of the transitions, the mean number of (colored) tokens in places.

3 An application example

As an application example, we used our tool to compute ESRG and DSRG on a well-known model of the literature: a critical section access algorithm with priorities [2].

Table 1 summarizes the obtained results for the different approaches: SRG, ESRG, RESRG and DSRG.[1]

The columns labeled by $Prio$ (the number of different priorities levels), GC (the allowed number of concurrent access requests), represent the parameters of the model. Columns labeled $St.$ represent the number of constructed states for each structure. $Peak$ is the total number of intermediate states stored to obtain the final structure (only for ESRG and RESRGs).

We observe here that the final number of states in the RESRG(strong) is less than the one of the RESRG(exact). This is counterbalanced by the type of performance indices that could be computed on the RESRG(exact), and not possible on the RESRG(strong).

Also, the number of states in the DSRG is comprised between the final number of states and the peak of the RESRG(exact) structure. Hence, the DSRG offers an alternative solution to tackle the peak problem of the ESRG-based structures.

[1] Experiments are performed on a PC/Linux machine with 3.2 GHz and 4 Gb of RAM.

		SRG	ESRG		RESRG (Strong)		RESRG (Exact)		DSRG
Pr	GC	St.	St.	Peak	St.	Peak	St.	Peak	St.
3	2	37	10	15	10	12	14	16	14
3	3	45	14	16	14	18	21	27	23
5	2	271	14	140	14	23	22	31	22
5	3	441	20	170	20	67	40	94	49
5	5	573	32	175	32	153	78	228	169
8	2	4681	20	2240	20	47	34	61	34
8	3	10337	29	4368	29	278	76	347	103
8	5	19409	47	5768	47	2515	295	3030	1189

Table 1. Comparing ESRG vs. DSRG

4 Conclusion

In this paper, we presented two new solvers to deal with asymmetric SWN models, the ESRG and DSRG solvers. Actually, these tools are available in a beta version in: *http://www.di.unito.it/∼greatspn*.

Currently, we are working to improve this version. In particular, to extend the GreatSPN frontend so that the control automaton will be automatically generated from a partially symmetric SWN model, and to extend $MCSolver$ to compute more complex user defined performance indices.

References

[1] S. Baarir, C. Dutheillet, S. Haddad, and J-M. Ilié. On the use of exact lumpability in partially symmetrical Well-formed Nets. In *Proc. of Int. Conf. on the Quantitative Evaluation of Systems (QEST)*, pages 23–32, Torino - Italy, September 2005.

[2] S. Baarir, S. Haddad, and J-M. Ilié. Exploiting Partial Symmetries in Well-formed nets for the Reachability and the Linear Time Model Checking Problems. In *Proc. of the Int. Workshop on Discrete Event Systems*, pages 223–228, Reims-France, September 2004.

[3] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. A symbolic reachability graph for coloured Petri nets. *Theoretical Computer Science*, 176(1-2):39–65, 1997.

[4] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic petri nets. *Performance Evaluation, special issue on Performance Modeling Tools*, 24(1-2):47–68, November 1995.

[5] M. Beccuti, S. Baarir, G. Franceschinis, and J-M. Iliè. Efficient lumpability check in partially symmetric systems. In *QEST*, pages 211–220, Riverside, CA, USA, September 2006.

[6] S.Haddad, J.M Ilié, M. Taghelit, and B. Zouari. Symbolic Reachability Graph and Partial Symmetries. *Proc. of Int. Conf. on Application and Theory of Petri Nets. Turin, Italy.*, 935:238–251, June 1995.