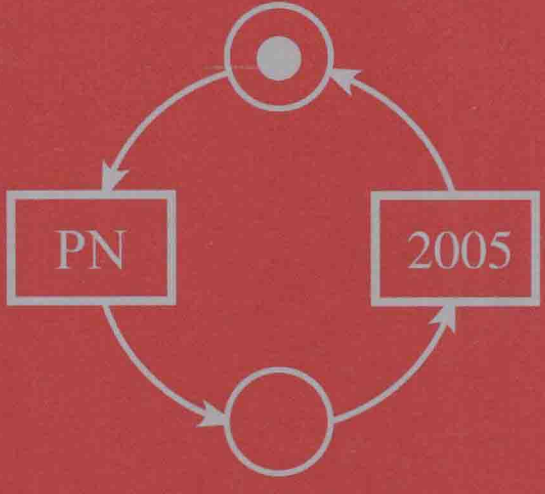
Gianfranco Ciardo
Philippe Darondeau (Eds.)

LNCS 3536

Applications and Theory of Petri Nets 2005

26th International Conference, ICATPN 2005
Miami, USA, June 2005
Proceedings



Springer

Gianfranco Ciardo Philippe Darondeau (Eds.)

Applications and Theory of Petri Nets 2005

26th International Conference, ICATPN 2005
Miami, USA, June 20-25, 2005
Proceedings

Springer

Volume Editors

Gianfranco Ciardo
University of California at Riverside
Department of Computer Science and Engineering
Riverside, CA 92521, USA
E-mail: ciardo@cs.ucr.edu

Philippe Darondeau
INRIA, IRISA, Campus de Beaulieu
35042 Rennes Cedex, France
E-mail: darondeau@irisa.fr

Library of Congress Control Number: 2005927320

CR Subject Classification (1998): F.1-3, C.1-2, G.2.2, D.2, D.4, J.4

ISSN 0302-9743
ISBN-10 3-540-26301-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26301-2 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11494744 06/3142 5 4 3 2 1 0

Preface

This volume contains the proceedings of the 26th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2005). The Petri net conferences serve to discuss yearly progress in the field of Petri nets and related models of concurrency, and to foster new advances in the application and theory of Petri nets. The conferences typically have 100–150 participants, one third from industry and the others from universities and research institutions, and they always take place in the last week of June. Successive editions of the conference are coordinated by the Steering Committee, whose members are listed on the next page, which also supervises several other activities—see the Petri Nets World at the URL `www.daimi.au.dk/PetriNets`.

The 2005 conference was organized in Miami by the School of Computer Science at Florida International University (USA). We would like to express our deep thanks to the Organizing Committee, chaired by Xudong He, for the time and effort invested to the benefit of the community in making the event successful. Several tutorials and workshops were organized within the conference, covering introductory and advanced aspects related to Petri nets. Detailed information can be found at the conference URL `www.cs.fiu.edu/atpn2005`.

We received altogether 71 submissions from authors in 22 countries. Two submissions were not in the scope of the conference. The Program Committee selected 23 contributions from the remaining 69 submissions, classified into three categories: application papers (6 accepted, 25 submitted), theory papers (14 accepted, 40 submitted), and tool presentations (3 accepted, 4 submitted). We thank all authors who submitted papers. We would like to express our gratitude to the members of the Program Committee and the other reviewers for the extensive, careful evaluation efforts they performed before the Program Committee meeting in Rennes; their names are listed on the next two pages. We also gratefully acknowledge Martin Karusseit, of the University of Dortmund, for his technical support with the Online Conference Service, which relieved many administrative tasks in the management of the review process.

This volume contains the papers that were presented at the conference after selection by the Program Committee, plus a few other papers that summarize invited lectures given at the conference. The invited papers included in this volume reflect the lectures given by Giuliana Franceschinis, Ken McMillan, Manuel Silva, and Jeannette Wing. Two more lectures were delivered at the conference: *New Dimensions for Nets* by Carl Adam Petri, and *Processes for a Service Oriented World* by Francisco Curbera. We are much indebted to Springer for smoothing all difficulties in the preparation of this volume.

April 2005 Gianfranco Ciardo and Philippe Darondeau

Organization

Steering Committee

Wil van der Aalst, The Netherlands
Jonathan Billington, Australia
Jörg Desel, Germany
Susanna Donatelli, Italy
Serge Haddad, France
Kurt Jensen, Denmark (chair)
Jetty Kleijn, The Netherlands
Maciej Koutny, UK
Sadatoshi Kumagai, Japan
Tadao Murata, USA
Carl Adam Petri, Germany (honorary member)
Lucia Pomello, Italy
Wolfgang Reisig, Germany
Grzegorz Rozenberg, The Netherlands
Manuel Silva, Spain

Organizing Committee

Alicia Bullard
Shu-Ching Chen
Peter Clarke
Yi Deng
Xudong He (chair)
Steven Luis
Tadao Murata (advisor)
Sol Shatz

Tool Demonstration

Shu-Ching Chen (chair)

Program Committee

Luca Bernardinello, Italy
Soren Christensen, Denmark
Gianfranco Ciardo, USA (co-chair, applications)
José Manuel Colom, Spain
Philippe Darondeau, France (co-chair, theory)
Luis Gomes, Portugal
Roberto Gorrieri, Italy
Xudong He, USA
Kees M. van Hee, The Netherlands
Petr Jančar, Czech Republic
Guy Juanole, France
Gabriel Juhás, Germany
Hanna Klaudel, France
Jetty Kleijn, The Netherlands
Maciej Koutny, UK
Sadatoshi Kumagai, Japan
Charles Lakos, Australia
Johan Lilius, Finland
Daniel Moldt, Germany
Madhavan Mukund, India
William H. Sanders, USA
P.S. Thiagarajan, Singapore
Enrico Vicario, Italy
Hagen Voelzer, Germany
Alex Yakovlev, UK

Referees

Bharat Adsul
Alessandra Agostini
Marcus Alanen
Joao Paulo Barros
Marek Bednarczyk
Marco Bernardo
Bernard Berthomieu
Eike Best
Jerker Björkqvist
Andrea Bobbio
Frank de Boer
Tommaso Bolognesi
Marcello Bonsangue
Roland Bouroulet
Cyril Briand
Didier Buchs
Cécile Bui Thanh
Frank Burns
Nadia Busi
Lawrence Cabac
Antonella Carbonaro
Josep Carmona
Antonio Cerone
Michel Combacau
Jordi Cortadella
Anikó Costa
Deepak D'Souza
Zhengfan Dai
Pierpaolo Degano
Salem Derisavi
Jörg Desel
Raymond Devillers
Michel Diaz
Roxana Dietze
Junhua Ding
Susanna Donatelli
Zhijiang Dong
Michael Duvigneau
Joost Engelfriet
Rob Esser
Alessandro Fantechi
Berndt Farwer
Carlo Ferigato
Gianluigi Ferrari
Joern Freiheit
Yujian Fu
Guy Gallasch
Mauro Gaspari
Edgar de Graaf
Mark Griffith
Stefan Haar
Serge Haddad
Bing Han
Keijo Heljanko
David Hemer
Kunihiko Hiraishi
Hendrik Jan Hoogeboom
Ying Huang
Guillaume Hutzler
Dorothea Iglezakis
Andreas Jakoby
Ryszard Janicki
Kurt Jensen
Jens Bæk Jørgensen
Kaustubh Joshi
Tommi Junttila
Mohamed Kaaniche
Seiichi Kawata
Victor Khomenko
Ekkart Kindler
Nicolas Knaak
Geoffrey Koh
Michael Köhler
Walter Kosters
Martin Kot
Lars Kristensen
Vinh Lam
Kristian Bisgaard Lassen
Kolja Lehmann
Kamal Lodaya
Robert Lorenz
Roberto Lucchi
Olga Marroquin Alonso
Axel Martens
Cecilia Mascolo
Vesna Milijic
Toshiyuki Miyamoto
Lian Mo
K. Narayan Kumar
Christian Neumair
Apostolos Niaouris
Katsuaki Onogi
Jan Ortmann
Elisabeth Pelz
Laure Petrucci
Sibylle Peuker
Giovanni Michele Pinna
Lucia Pomello
Franck Pommereau
Ivan Porres
Christine Reese
Heiko Rölke
Olivier Roux
Eric Rozier
Luigi Sassoli
Zdenek Sawa
Karsten Schmidt
Carla Seatzu
Yang Shaofa
Christian Shelton
Tianjun Shi
Natalia Sidorova
Jason Steggles
S.P. Suresh
Tatsuya Suzuki
Koji Takahashi
Shigemasa Takai
Mikko Tiusanen
Fernando Tricas
Kohkichi Tsuji
Dietmar Tutsch
Naoshi Uchihira
Robert Valette
Somsak Vanit-Anunchai
Daniele Varacca
Marc Voorhoeve
Lisa Wells
Michael Westergaard
Luke Wildman
Dianxiang Xu
Gianluigi Zavattaro

Table of Contents

Invited Papers

Full Papers

Tool Papers

Expressiveness and Efficient Analysis of Stochastic Well-Formed Nets

Giuliana Franceschinis

Dip.Informatica, Univ. del Piemonte Orientale, Alessandria, Italy
giuliana@mfn.unipmn.it

Abstract. This paper is a survey of the Stochastic Well-formed Net (SWN) formalism evolution, in particular it discusses the expressiveness of the formalism in terms of ease of use from the modeler point of view, and briefly presents the main results that can be found in the literature about efficient (state space based) analysis of SWN models. Software tools supporting SWN design and analysis are also mentioned in the paper. The goal of the paper is not to present in details the formalism nor the analysis algorithms, but rather to recall the achieved results and to highlight open problems and possible directions for new developments in this research area.

1 Introduction

Well-formed Nets (SWN) and their stochastic extension (SWN) were first presented at the beginning of '90s [15, 16], as an evolution of (Stochastic) Regular Nets [34, 29]. SWN belong to the class of High Level Petri Nets (HLPN), and includes (stochastic) timed transitions as well as immediate transitions, with a semantics inherited from Generalized Stochastic Petri Nets (GSPN) [1]. A peculiar feature of the SWN formalism is the ability to capture in its structured color syntax the behavioral symmetries of the model which can be automatically exploited for building a smaller state space, where markings are aggregated according to an equivalence relation that preserves the interesting model properties (both qualitative and quantitative).

This paper is a survey of the SWN formalism evolution, in particular it discusses the expressiveness of the formalism in terms of ease of use from the modeler point of view, proposing some useful extensions, and briefly presents the main results that can be found in the literature about efficient (state space based) analysis of SWN models. Software tools supporting SWN design and analysis are also mentioned in the paper. The ideas presented in this paper and the cited bibliography mainly refer to the work developed in this field by researchers at the University of Piemonte Orientale (Dip. di Informatica), Alessandria, and at the University of Torino (Dip. di Informatica), Torino in Italy, at the University of Paris VI (LIP6), the University of Paris Dauphine (LAMSADE) and at the University of Reims Champagne-Ardenne in France: these research groups are actively cooperating in developing both new theoretical results, application examples and software tools for SWN.

G. Ciardo and P. Darondeau (Eds.): ICATPN 2005, LNCS 3536, pp. 1–14, 2005.

The paper is organized as follows: Sections 2 and 3 contain the discussion of the SWN expressiveness, in particular the first section considers possible extensions to the original syntax that could ease the process of building SWN models, without sacrificing the efficiency in the SWN model analysis, while the second section discusses the issue of compositional SWN model construction methods. Sections 4.1 and 4.2 surveys both consolidated and more recent methods for efficient (state space based) analysis of SWN models. Section 5 presents some existing tools that support design and analysis of SWN models. Section 6 outlines some promising directions for future development.

2 On the SWN Formalism Syntax and Possible Extensions

The Stochastic Well-formed Net (SWN) formalism has been introduced in 1990 [15] as an extension of the Regular Net (RN) formalism: it is a high level Petri net formalism similar to Coloured Petri Nets (CPN) [41, 42], but featuring a constrained colour stucture syntax which allows to automatically discover and exploit the behavioural symmetries of the model for efficient state space based analysis. The Symbolic Marking and Symbolic Firing concepts defined for SWNs lead to a Symbolic Reachability Graph construction algorithm [17] whose size can be orders of magnitude smaller than the complete RG (examples have been presented in [16, 2, 18]).

The constrained color syntax of SWN is based on the definition of a finite set of basic types, called *basic colour classes*, and a limited set of basic functions and basic predicates defined on such classes: the places and transition colour domains, and the arc functions are then constructed upon these basic objects. In fact the colours associated with places and transitions are tuples of typed elements, where the element types are the basic colour classes, and the arc functions are (sums of) tuples whose elements are basic functions defined on basic colour classes. Basic predicates can be used to restrict the possible colours of a transition or to obtain arc functions whose structure may change when applied to different color instances of a given transition.

Basic color classes are finite sets of colors, can be ordered (circular order defined through a successor function) and can be partitioned into *static subclasses* (the partition into static subclasses is called the *static partition* of a basic color class). Intuitively, colors in a basic color class are homogeneous objects (processors, processes, resources, message types, hosts in a network, etc.) and each static subclass groups objects of the same nature with similar behaviour (so if all objects in a class behave in the same way, the static partition will contain a unique static subclass).

The definition of basic color classes static partition is a delicate step in the model definition, because this is the part of the model specification that defines the possible symmetries that can be exploited by the SRG. If the modeler fails to define the coarsest partition allowing to correctly express the system behaviour, the state space aggregation automatically achieved by the SRG algorithm might be less effective.

Observe that the static partition may need refinement if the model is built in a compositional way: in fact the static partition of a given class defined for one submodel might not be the same needed in another submodel, and when the submodels are composed a new static partition taking into account those of the two component submodels must be defined.

The above considerations suggest that the modeler should be supported in the color structure definition phase by some automatic tool able to check the model color inscriptions (initial marking, arc functions, transition guards) and decide whether the static partition definition is adequate, and propose a change if need be. A work in this direction is that presented in [49, 48], where the proposal is to impose less constraints on the syntax, and let the modeler directly refer to (subsets of) color elements within basic color classes when specifying the model: an algorithm is provided that is able to automatically derive the optimal static partition based on the analysis of the model color structure. In the same work some new useful extensions to the formalism are also proposed (e.g. the possibility of considering an ordering among elements in a class, not the circular one allowed in SWN, and as a consequence the introduction of new comparison operators in basic predicates: this is actually only syntactic sugar because it can be expressed using static subclasses, but it is often very convenient for the modeler).

Up to now the "stochastic" part of the formalism has not been mentioned: in SWN transitions can either be timed or immediate (fire in zero time after enabling), the timing semantics of SWN reflects that of Generalized Stochastic Petri Nets (GSPN), and in fact the net obtained by unfolding a SWN is a GSPN. The complete specification of a GSPN model includes the definition of average firing times of timed transitions, and the priority and weights of immediate transitions: the latter task is needed to give a deterministic or probabilistic criteria for immediate transition conflict resolution (required to construct the underlying stochastic process, or to simulate the model behavior in time, for performance evaluation purposes). This task requires a clear knowledge of the potential (direct and indirect) conflicts between immediate transitions: in [47] a technique is proposed that can support a modeler in this complex task. The transition times, priority and probability definition in SWN must take into account the color structure of the model, and may influence the color classes static partition (in fact different instances of a given transition may have different average firing times or different priority/weight). Moreover the priority and weight definition method proposed in [47] can be directly applied only at the level of the net unfolding: to overcome this problem a way of expressing in a parametric and symbolic way the potential conflict relations between different instances of SWN (immediate) transitions is required, together with a symbolic calculus allowing to compute such symbolic expressions using the net color structure information. In [22, 11, 12] a syntax for expressing such structural conflict relations, and a calculus for their computation has been proposed: the chosen syntax is very similar to the SWN arc function syntax, and hence it should not be difficult for the modeler to interpret them: the extension to SWN of the priority/weight

definition method developed for GSPN exploiting the above mentioned symbolic expressions is still under development.

3 Composing SWN Models

When modeling complex systems, possibly designing several models of a given system, a compositional model construction method is more convenient than a *flat* and monolithic modeling approach. In the CPN formalism [41] this idea has led to the definition of hierarchical CPNs, with the possibility of defining submodels (pages) with well defined interface nodes (ports), of including *substitution transitions* in a model, that are an abstraction for a submodel and are connected to socket nodes (to be related with port nodes in the submodel represented by the transition), and of using fusion places, which are a mechanism to merge places within or across submodels. In the Stochastic Activity Network (SAN) formalism [44], replicate and join constructs can be used to create instances of submodels (possibly several replicas of the same submodel) and compose them by merging common places. Other approaches to high level Petri nets composition have been defined, for example the Box Calculus[8] or the Cooperative Nets[45, 46], or the compositional WNs (cWN)[43]. In the context of SWN, the compositionality issue has emerged as a natural consequence of the application of the formalism and related tools to non toy case studies: the problem has been tackled both from a pragmatic point of view and from a theoretical point of view. Composition operators working by *superposition of nodes with matching labels* have been defined first for GSPNs [27] and then extended to SWNs: a composition tool, called *algebra*[7] has been implemented and integrated into the *GreatSPN* software package to support such operators.

A relevant aspect to be considered when composing colored submodels is the treatment of the color structure: one possibility that highly simplifies composition is to assume that the submodels to be composed have already a "compatible" color structure (same definition of common color classes), that the identifiers of the color elements coming from the component submodels that are intended to represent the same object match, while identifiers that represent *local* and independent objects in the two submodels are disjoint. In *algebra*, compatible color definition is assumed, shared identifiers are preceded by a special symbol, and must match in the submodels to be composed, while non shared identifiers are automatically renamed in case of a clash. This pragmatic approach however is not flexible enough: in [4] a more parametric method has been proposed, where the submodels to be composed may have *parametric color classes*, which can be instantiated when a parametric submodel is instantiated and composed with another submodel; the original idea proposed in [4] is that color definitions can *flow* from one submodel to another one, by means of color import/export specification: colors exported by one model can be imported by another model when they are composed.

Trying to generalize the idea of parametric submodels, to allow the flexible definition of submodel interface and composition operators, in the OsMoSys

framework [52] the concept of *model class* has been defined. Model classes are parametric models represented using a given formalism: they have a precise interface, and they can be instantiated and connected to elements of a larger model to build a complete (solvable) model or a more complex model class. When a model class becomes a part of a larger model class, each parameter can either be istantiated or become a parameter of the new model class, and each interface can either be hidden or become an interface of the new model class (possibly with different name).

Another important issue when defining model composition, is the definition of composition operators: on one hand a minimal set of composition operators allowing to express all interesting composition schema should be defined, on the other hand if the "final user" is not an expert in the use of formal languages but rather an expert in a given application domain which wants to easily build models of different scenarios by composing submodels taken from a library, a rich and application oriented set of composition operators might be desirable. In OsMoSys, composition operators can be part of the formalism used to express the model classes to be composed, or can be an extended version of that formalism, or can be a completely new formalism, defined for the purpose of composition (this is easily realizable in OsMoSys because it is a multi-formalism framework featuring a meta-formalism, that is a language for defining formalisms supporting inheritance between formalisms). In [31] and [30] two examples of application oriented SWN composition formalism defined within the OsMoSys framework are presented.

Last but not least, (de)composition of models can be used for analysis efficiency purposes (specifically, in case of SWNs decomposition can be combined with state space aggregation in a very effective way, see [37, 38, 23], this aspect shall be further discussed in Sec. 4.2): also in this case a flexible language for describing the (de)composition structure of a model to be used for analysis purposes is needed, the OsMoSys framework can be applied also to this case.

4 Efficient Analysis

In this section analysis methods for SWNs are discussed; we focus on state space based analysis methods for both qualitative and quantitative evaluation. In Subsection 4.1 the Symbolic Reachability Graph (SRG), automatically exploiting SWN models behavioral symmetries, and the Extended Symbolic Reachability Graph (ESRG), able to deal with partially symmetric systems, are presented: the aggregate state spaces generated by these algorithms can be used both for qualitative properties analysis, through model checking, and for quantitative properties analysis, by generating and solving a *lumped* Continuous Time Markov Chain (CTMC). Some recent results on the possibility of using very efficient data structures (Data Decision Diagrams [21] and Set Decision Diagrams [50]) for storing the Symbolic Markings and the SRG to increase the state space size that can be generated for model checking purposes are also mentioned in this subsection. In Subsection 4.2 algorithms combining SRG aggregation meth-

ods and decomposition methods (based on tensor algebra representation of the CTMC) are presented: the combination of these two methods allow to increase the size of models that can be solved for performance evaluation purposes.

4.1 SRG, ESRG, and Further Evolution of ESRG for Model Checking and Performance Analysis

The main motivation for introducing the SWN formalism has been the possibility of automatically exploiting behavioral symmetries to build an aggregate state space, which can be orders of magnitude smaller than the complete state space (the maximum achievable reduction is the product of the factorial of all static subclasses cardinalities for non ordered classes, multiplied by the product of the cardinalities of ordered classes with only one static subclass): this reduction is achieved *automatically* through the definition of the Symbolic Marking and Symbolic Firing concepts. Each Symbolic Marking represents in a compact way an equivalence class of ordinary markings. Symbolic transition instances can be checkled for enabling directly on the Symbolic Marking representation and a Symbolic Firing Rule can be used to compute the new reached Symbolic Marking without ever explicitly representing the underlying ordinary markings and transition firings. A canonical representation is defined for Symbolic Markings, so that it is easy to check whether a reached Symbolic Marking corresponds to an already reached equivalent class. Hence a Symbolic Reachability graph generation algorithm can be defined [16, 17] and the analysis can be performed on this graph instead of the complete RG. When the model has to be used for performance evaluation purposes, a stochastic process must be derived and analysed: in the case of SWN the underlying stochastic process is a CTMC isomorphic to the RG. It has been proven that, due to the constraints imposed by the formalism on the color dependence of transition firing times and weights, the CTMC of a SWN satisfies both the strong and exact lumpability conditions[9] with respect to the aggregation induced by the SRG, so that performance analysis can be performed on a CTMC which is isomorphic to the SRG, and its transition rates can be directly computed from the Symbolic Marking and Symbolic Firing information without need to ever expanding the complete RG.

It is also interesting to observe that the Symbolic Marking and Symbolic Firing can be successfully exploited also in discrete event stochastic simulation of SWN [32]: the gain in this case is due to the event list management, in fact the event list contains the set of enabled transition instances in the current state, and since one symbolic transition instance may group several ordinary transition instances, fewer enabling tests are required at each state change, furthermore the resulting "symbolic" event list can be much smaller than the ordinary one thus saving both time and space.

Unfortunately the effectiveness of the SRG method vanishes if the behaviour of a model is not completely symmetric: as already discussed in Sec. 2 this is related to the number and cardinality of static subclasses within each basic color class. In the worst case where all non ordered basic color classes are partitioned in cardinality one static subclasses and all ordered basic color classes are par-

titioned into two or more static subclasses, the SRG coincides with the RG. Often however, it happens that the asymmetries in the model behavior involve only a small part of the state space, but they influence the level of aggregation of the whole RG. This consideration led to the evolution of the SRG called Extended SRG (ESRG) [36]: it exploits symmetries in a more flexible way, and takes into account the partition into static subclasses only when actually needed. The ESRG structure uses a two level representation of the aggregate states: the Extended Symbolic Markings in fact have a Symbolic Representation (SR) (first level) that groups all the states that *would be equivalent* if all color classes had only one static subclass, and a set of Eventualities (Ev) (second level) that are the SRG Symbolic Markings included in the ESM SR. The gain that can be obtained from the ESRG is due to the fact that eventualities are explicitly represented only when actually needed, that is only when asymmetries actually arise in the behavior (typically because some transition which depend on the static subclasses partition is enabled). Hence the effectiveness of ESRG aggregation depends on how often asymmetries influence the system evolution. A weak point of the ESRG, with respect to the SRG, is that not all properties can be checked on the ESRG, and in some case eventualities that were not created by the ESRG algorithm, need to be generated in a second phase when properties are checked on the ESRG or when a lumped CTMC has to be generated for quantitative evaluation [13, 14, 5]. In particular when a CTMC has to be generated from the ESRG, the lumpability condition must be checked (while in the SRG it is ensured by construction) on some markings: depending on the type of performance index to be computed, one can choose whether to check for strong or exact lumpability, achieving different degrees of aggregation. The lumpability check algorithms working on the ESRG [14, 5] are in general more efficient than those that may be applied directly on the SRG (without a clue on the possible additional aggregations not captured by the SRG).

A further evolution of the ESRG consists in allowing a *partial refinement* of a SR into its eventualities, which in some sense means adding a third level in the ESM representation laying between SR and Ev: in fact subsets of eventualities could remain aggregated, and this may happen when in a given state only the static partition of a few classes is involved, while those of the other classes do not influence the local behavior of the system (see [10, 35, 3, 39]). The E^2SRG first introduced in [10] and applied in [6] is an example of such three level representation of a ESM: it has not yet been implemented, and still needs some work to be completely worked out. This is one possible direction of further evolution of these SRG-ESRG methods.

Combining two types of symmetries: SRG and Decision Diagrams The type of symmetries exploited by the SRG are based on a notion of equivalence of similar markings, however another type of symmetry can be exploited based on the decomposition of the state into sub-states and factorisation of substates: this type of symmetry in the states structure can be efficiently captured and exploited by using specific data structures, the Decision Diagrams. The SRG technique and DD based techniques however are not easily combined because of the mutual

dependence of sub-states in Symbolic Markings: this dependence is due to presence of Dynamic Subclasses in its representation, whose definition and evolution depend on the whole state (so that the representation of a given sub-marking after a symbolic firing may change even if the fired transition had no connections with such sub-marking). Recently the problem has been faced by means of a new type of Decision Diagram called Data Decision Diagrams (DDD) and Set Decision Diagrams (SDD)[21, 48, 50]) that have shown to be very effective for storing the states of huge SRG. Up to now these new techniques have been exploited only in the context of model checking algorithms: it would be interesting to investigate the possibility of maintaining the same efficiency in the CTMC generation phase.

4.2 De-composing SWN Models

In this section the analysis methods proposed in [37, 38, 23] are briefly summarized: the main idea behind these methods consists of combining two orthogonal techniques for coping with the state space explosion problem that often arises when the CTMC of a complex SWN model is generated for performance analysis. The first technique is *state aggregation*, which in the context of SWN can be obtained using the SRG generation algorithm, the second technique is *decomposition* of a model into submodels and representation of the CTMC infinitesimal generator (the matrix of state transition rates) as an expression of much smaller generators, obtained from each submodel in isolation and combined through tensor algebra (Kronecker) operators (see [25, 24] for the application of this technique to GSPN models). In the latter technique, the CTMC is solved using its *decomposed* representation, so that the space required to generate and solve the CTMC is considerably smaller than that required to store the complete CTMC explicitely.

Two way of (de)composing a SWN model are considered: synchronous and asynchronous. In both cases the component submodels need to be extended in order to compute from each of them in isolation a local SRG. The method is not completely general: in fact it can be applied to a decomposed SWN model only if some constraints are satisfied: the constraints are needed to ensure that the lumped CTMC resulting from Kronecker composition of local lumped CTMCs is equivalent to the lumped CTMC corresponding to the SRG of the whole model. The subclass of SWN models that satisfy the constraints may appear as rather restricted, but it is representative of a relevant class of models from interesting application domains. In [37, 38] both semantic conditions (defined on the state space) and syntactic (sufficient) conditions (defined on the net structure and its decomposition) are defined to ensure that the method can be applied.

These composition/decomposition methods have been implemented (see Sec. 5) and partially integrated in GreatSPN: the most difficult part for the application of the method is that currently no automatic support is provided to decompose the model, build the extended subnets, and check that the syntactic conditions for the applicability of the method are satisfied. Some theoretical results about structural analysis of SWNs are available, but more work is needed to adapt them to this specific problem, and to develop an automatized (or at least semi-automatized) procedure for the application of these methods.