Analysing Coloured Petri Nets by the Occurrence Graph Method

Jens Bæk Jørgensen

Computer Science Department, University of Aarhus
Ny Munkegade, Bldg. 540
DK–8000 Aarhus C, Denmark

Phone: +45 89 42 31 88
Telefax: +45 89 42 32 55
E-mail: jbj@daimi.aau.dk

Abstract

This paper provides an overview of the work done for the author's PhD thesis. The research area of Coloured Petri Nets is introduced, and the available analysis methods are presented. The occurrence graph method, which is the main subject of this thesis, is described in more detail. Summaries of the six papers which, together with this overview, comprise the thesis are given, and the contributions are discussed.

A large portion of this overview is dedicated to a description of related work. The aim is twofold: First, to survey pertinent results within the research areas of — in increasing generality — Coloured Petri Nets, High-level Petri Nets, and formalisms for modelling and analysis of parallel and distributed systems. Second, to put the results obtained in this thesis in a wider perspective by comparing them with important related work.

1 Introduction

The formalism of *Petri Nets* was founded in 1962 [70]. During the next decades, Petri Nets matured into a powerful, general, graphical language

for modelling and analysis of systems; in particular, parallel and distributed systems.

In the sixties, and mid and late seventies, Petri Nets was primarily used as a theoretical model for concurrency. In the late seventies, with the advent of the class of *High-level Petri Nets*, Petri Nets was augmented with concepts allowing more succinct description of models. As a consequence, the application area was extended to modelling and analysis of much larger systems. Two main kinds of High-level Petri Nets were published: The definition of *Predicate/Transition Nets* [31, 33, 34] in 1979 presented the seminal idea of tokens carrying data values. The definition of *Coloured Petri Nets* (*CP-nets* or *CPN*) [44, 45] in 1981 introduced a notion of elaborated data types as known from high-level programming languages like Pascal.

In 1989, CP-nets was extended with hierarchy constructs, allowing large models to be structured as sets of modules with well-defined relations between them [41, 45]. At the same time, the first computer tool to support CP-nets, the commercially developed, graphical tool *Design/CPN* [48] emerged.

The tool support meant that CP-nets was now applicable for practitioners working with systems design and analysis. In the nineties, many CPN projects were carried out in the industry and documented in the literature, e.g., in the areas of communication protocols [20, 30, 42], operating systems [12, 13], hardware designs [35, 82], embedded systems [73], software systems designs [62, 77], and business process re-engineering [66, 71].

Alongside with definition of the basic formalisms of Petri Nets and High-level Petri Nets, a set of verification methods was developed. A verification method allows dynamic properties of a model to be formally proved. However, for models of real-world systems, there are practical difficulties. This is the reason why, till now, only a few Petri Nets projects applying verification methods to anything but tiny models have been documented.

About thesis and overview

The motivation for this thesis has been to take a step closer to practical applicability of verification of CP-nets. The work done is a study of the tool support, practical use, and mathematical foundation of one of the verification methods, the so-called occurrence graph method. This study has, of course, taken place in a context, explaining why this thesis also deals with other aspects of CP-nets such as creation and simulation of models.

The work done for this thesis is documented in six papers:

- *Design/CPN — A Computer Tool for Coloured Petri Nets.* Joint work with Søren Christensen and Lars Michael Kristensen; reference [22]; in brief, referred to as the *Design/CPN* project/paper.

- *Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries.* Joint work with Lars Michael Kristensen; reference [51]; in brief, referred to as the *OS/Lamport* project/paper.

- *Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets.* Joint work with Kjeld Høyer Mortensen; reference [54]; in brief, referred to as the *DistBETA* project/paper.

- *Analysis of Bang and Olufsen's BeoLink® Audio/Video System Using Coloured Petri Nets.* Joint work with Søren Christensen; reference [21]; in brief, referred to as the *B&O* project/paper.

- *Verification by State Spaces with Equivalence Classes.* Joint work with Lars Michael Kristensen; reference [53]; in brief, referred to as the *Equivalence* project/paper.

- *Construction of Occurrence Graphs with Permutation Symmetries Aided by the Backtrack Method.* Reference [49]; in brief, referred to as the *Backtrack* project/paper.

The work done for this thesis was carried out under the auspices of the Devise project, a large and long-running research project at Computer Science Department, University of Aarhus. The headline for the Devise project is *Experimental Systems Development*, and the aim is to increase productivity and quality in the development of large, complex software systems. A major goal is to produce theoretically well-founded computer tools of a maturity allowing use, or at least experiments, outside the world of academia, i.e., in real industrial environments. Three research areas are involved: Object-oriented programming languages, systems development, and CP-nets. Within the setting of the Devise project, two major attitudes to the research done for this thesis came natural: It was largely based on experiments, and development of tools had high priority.

These attitudes carry over to this overview paper: When an analysis method is discussed and evaluated, it is important not to look at the un-

derlying theory in isolation, but also to consider tool support and sustaining practical experiments.

To read this overview paper, it is not required to have a detailed knowledge of CP-nets. It is, though, necessary to be familiar with the basic ideas of some kind of Petri Nets, e.g., Place/Transition Nets (PT-nets) as described in [67].

Terminology

In the literature about CP-nets, the term *occurrence graph* is used. For other kinds of Petri Nets and transition systems in general, it is more standard to say *state space* or *reachability graph* for the same thing. These three terms are used as synonyms in this paper, choosing the one that seems most appropriate in the given context, e.g., respecting the terminology preferred by the author, whose work is being discussed. Similarly, in contexts where the term *marking* is applicable, the term *state* is sometimes used as synonym.

The term *analysis* is in this paper used generically to mean investigation of the behaviour of a system, either using informal methods, formal methods, or both. The term *validation* means convincing a human being that a system behaves as expected. It is perfectly acceptable to do this informally. In contrast, the term *verification* and the synonym *formal analysis* mean proving in the rigourous, mathematical sense of the word that a system has certain properties.

Structure of overview

The remaining part of this overview paper is organised as follows: In Section 2, the main analysis methods for CP-nets are presented with special attention to the occurrence graph method. Section 3 provides summaries of the six papers comprising this thesis, and discussions of the contributions they spawn. Related work is covered in the next two sections. Section 4 describes a number of state space analysis methods, and compares them with a particular method focussed on in this thesis. Section 5 sums up some projects which applied occurrence graph analysis. These projects are subsequently compared with two related projects conducted as part of the work for this thesis. The conclusions are drawn in Section 6.

2 The Research Area

This section gives a brief presentation of the research area in which the work for this thesis was carried out. In Section 2.1, the main analysis methods for CP-nets are discussed. Sections 2.2 and 2.3 provide more details about the occurrence graph method.

2.1 Analysis of Coloured Petri Nets

A number of analysis methods are available for CP-nets. The basic one is *simulation*, allowing the user to examine the behaviour of a CP-net by playing the token game. Simulation of CP-nets has been supported by the Design/CPN tool since 1989, and been successfully applied in a large number of projects. Simulation is indispensable to increase the understanding of a system, to validate, to test ideas, and to find flaws and bugs. However, simulation is inadequate to verify a system.

For verification, formal analysis methods are needed. For CP-nets, there are two main methods: The first is the occurrence graph method which is described in Sections 2.2 and 2.3 below. The second is the *invariant method*, whose prime constituent is *place invariants*. Here, the basic idea resembles the way invariants are used in ordinary verification of computer programs. Equations are established, which hold for all reachable markings of a considered CP-net. These equations provide the basis for conducting mathematical proofs to establish certain properties. Presently, preliminary support for place invariant analysis exists for the Design/CPN tool [84].

In computer science, the approaches to verification of systems are often divided into two distinct categories with very different characteristics. One category is the *state space methods*, to which the occurrence graph method belongs. The other category is often referred to as *theorem-proving*, and contains the invariant method. Both categories are large, active research areas. In this overview paper, the focus is confined to consideration of state space methods; a general discussion of theorem-proving is outside the scope.

2.2 O-Graphs and the Occurrence Graph Method

In its simplest form, an *occurrence graph* for a CP-net is a directed graph with a node for each reachable marking and an arc for each occurring bind-

ing element[1]. The source of an arc is the marking in which the associated binding element occurs, and the destination is the marking resulting from the occurrence. This kind of graph is called a *full occurrence graph* (*O-graph*) [46]. When it is finite, it can be used to derive an abundance of verification results, e.g., reachability, boundedness, liveness, and fairness properties. Within computer science, the idea of occurrence graphs, or in general state spaces for transition systems, is old.

In this overview paper, the term *occurrence graph* is used generically, and includes all variants to be encountered below.

The *occurrence graph method*, also referred to as *occurrence graph analysis*, prescribes the following basic approach to verification of a given CP-net: First, generate the occurrence graph; then, run appropriate algorithms investigating the graph to get answers to the questions of interest. This sounds deceptively simple — and indeed it is. The practical applicability of the occurrence graph method is hampered by a severe drawback, the well-known *state explosion problem*: Even for relatively small CP-nets, the occurrence graphs are often so large that they cannot be constructed given the available computer technology of today. Alleviation of this seemingly inherent complexity problem is a major challenge of research. Several ingenious approaches have been proposed. One of these, OS-graphs, is a main focus of this thesis, and is introduced in Section 2.3 below. A number of other approaches are described and discussed in Section 4.

O-graphs for CP-nets have been supported by the Design/CPN tool since 1992. Some projects, which have applied O-graph analysis, are discussed in Section 5.

2.3 OS-Graphs

An *occurrence graph with permutation symmetries* (*OS-graph*)[2] [46, 47] is a reduced representation of an O-graph. For a given CP-net, the OS-graph, properly inscribed, contains the same information as the O-graph. Moreover,

[1] In CP-nets, there are *variables* for transitions, which are used to determine the occurrence modes. A *binding* assigns values to all variables of a transition. A *binding element* is a pair consisting of a transition and a binding.

[2] In this paper, general OS-graphs, *occurrence graphs with symmetries*, as defined in Chapter 3 of [46] are not considered. Therefore, without confusion, the abbreviation *OS-graphs* is used for the subclass of *occurrence graphs with permutation symmetries*, as defined in Chapter 4 of [46].

a lot of useful information is easily retrievable from the OS-graph. Typically, the OS-graph is orders of magnitude smaller than the O-graph. The original idea of OS-graphs was published in 1986 [40].

OS-graphs are based on the notion of *symmetry*, which appears when a system is composed of similar components, whose identities are immaterial with respect to occurrence graph analysis. As an example, consider the well-known dining philosophers system. A state of this system, in which the eating philosophers are numbers 1 and 3, is symmetric with a state, in which the eating philosophers are numbers 3 and 5. The first state can be mapped to the second by the permutation, which rotates philosopher i into philosopher $i+2$ (modulo the number of philosophers). Symmetry is also present in many real-world systems.

Definition of an OS-graph for a CP-net requires that two equivalence relations are present — one on the set of markings and one on the set of binding elements. The OS-graph has a node for each equivalence class of reachable markings (for two equivalent markings, either both or none of them are reachable). The OS-graph has an arc between two nodes, if and only if there is a marking in the equivalence class of the source node in which a binding element is enabled, and whose occurrence leads to a marking in the equivalence class of the destination node. There is exactly one arc for each equivalence class of binding elements with this property.

The equivalence relations are required to be on a certain form. They must be induced by *symmetry groups*, which are algebraic groups. One symmetry group is associated with each colour set of the considered CP-net. The symmetry group determines how the colours are allowed to be permuted, e.g., by arbitrary permutations, by rotations only (for an finite, ordered colour set), or by no permutations at all.

For the *atomic* colour sets, which are the colour sets defined without reference to other colour sets (e.g., booleans or enumeration types), the symmetry groups are chosen by the user. For the other colour sets, the *structured* colour set (e.g., Cartesian products or lists), the symmetry groups are inherited, i.e., automatically derived, from the user-chosen symmetry groups.

The symmetry groups induce a group of *permutation symmetries*, whose elements are functions that can be applied to markings and binding elements. Two markings are *equivalent*, also called *symmetric*, if and only if there exists a permutation symmetry mapping one of the markings to the other. Similarly for binding elements.

When the symmetry groups are chosen such that inherent symmetries of

the considered CP-net are captured (in a well-defined way), the OS-graph is said to be based on a *consistent permutation symmetry specification*. The consistency ensures that the OS-graph in fact does contain the same information as the corresponding O-graph.

The algorithm to generate OS-graphs is a straightforward modification of the standard algorithm to generate O-graphs [46]. The test of equality before a new marking is inserted, is replaced by a test of equivalence. In addition, an equivalence test before insertion of a new arc is introduced.

For some CP-nets, e.g., those containing only atomic colour sets, the equivalence test can be performed efficiently [2]. In general though, the test is hard. In [26], the equivalence test is proved to be at least as hard as the graph isomorphism problem [55], and it is speculated that the test is even NP-complete. If the group of permutation symmetries is finite, the equivalence test may be performed by a brute force application of all permutation symmetries, one by one, to see if there exists one that maps a newly generated marking to one already represented in the graph. This works satisfactorily for CP-nets with a small number of permutation symmetries. E.g., this is the case if all colour sets are small. However, in general, the computational complexity of the brute force approach is prohibitive. Often the number of permutation symmetries is exponential in the size of the system under consideration. Efficient heuristics to aid the equivalence test are needed.

Tool support for OS-graphs has been lacking for many years. Because of that, until recently, the method had only been used on tiny examples, like the dining philosophers system, for which the calculations could be done by hand. The novel *Design/CPN OS-Graph Tool* (*OS-tool*) [50] now allows the method to be used in practice, and a few interesting experiments have already been carried out.

A generalisation of the concept of OS-graphs called *occurrence graphs with equivalence classes (OE-graphs)* [46] exists. In OE-graphs, it is not required that the equivalence relations are on a certain form, e.g., they do not have to be induced by algebraic groups like in OS-graphs. Any two equivalence relations on markings and binding elements, respectively, satisfying a well-defined consistency requirement may be used. OE-graphs are applicable to systems where there is no symmetry in the sense of OS-graphs, i.e., expressed as permutations of colours, but some other kind of equivalence. The information about a system, preserved by an OE-graph, is dependent on the given equivalence relations. The OS-tool supports OE-graphs as well.

3 The Projects and their Contributions

This section sums up the six papers comprising this thesis. The papers have a common topic, but are written independently. Therefore, there are differences in style, terminology, and typography between some of the papers.

The Design/CPN, OS/Lamport, and Equivalence papers can be read without prior knowledge of CP-nets. A reader, who is unfamiliar with CP-nets, and who wants to read the entire thesis, is recommended to read the OS/Lamport paper first, because it informally explains and formally defines all the necessary concepts and notation. The Design/CPN paper contains a short informal introduction to CP-nets, which gives a sufficient background to read also the DistBETA and B&O papers. A reader, already familiar with CP-nets, may read the papers in any order.

The ordering chosen for this section is the following: First, two papers, whose main focus is development of tools (Design/CPN, OS/Lamport), are summed up; then, two papers describing large, practical experiments (DistBETA, B&O); and finally, two more mathematical papers proposing new application areas and improved algorithms for the considered methods and tools (Equivalence, Backtrack).

In addition to summaries of the papers, this section contains a discussion of their contributions, and identification of relations between the projects.

3.1 Design/CPN

The Design/CPN paper [22] describes the Design/CPN tool [48], which was the first graphical computer tool supporting CP-nets. Design/CPN has been under ongoing development since it first emerged in 1989. The tool was described as it appeared at the time of writing, i.e., in the beginning of 1997.

Design/CPN is developed in close cooperation between the company Meta Software Corporation, Cambridge, Massachusetts, and the CPN group at Computer Science Department, University of Aarhus, Denmark. Approximately 40 man-years have been invested in the development[3]. Design/CPN is being used world-wide by more than 200 companies and research institutions.

Design/CPN uses a language called CPN ML for declarations and net inscriptions in CP-nets. CPN ML is Standard ML [65, 85] extended with

[3]Before the PhD study, this author worked two years full-time on the development, support, and maintenance of Design/CPN.

some syntactical sugar to ease definitions of colour sets, variables, etc. Design/CPN has three basic parts — an editor, a simulator, and an O-graph tool — plus some extensions, e.g., the libraries [50, 84].

The *editor* supports construction and modification of CP-nets. Sophisticated and comprehensive facilities for making nice drawings are provided. Also, the editor enforces a number of built-in syntax restrictions, e.g., it is not possible to draw an arc between two transitions or between two places. However, it is impossible to catch all syntax errors efficiently that way. Hence, there is a *syntax checker*, which can be invoked, when the user wants to ensure that a created model constitutes a legal CP-net.

The *simulator* supports execution of CPN models. Different modes of simulation are provided suitable for different purposes. In *interactive mode*, the user is in full control, sets breakpoints, chooses between enabled transitions, possibly changes markings, and studies the token game in detail. Typically, a few steps per minute are executed. In *automatic mode*, the simulator itself makes random choices between enabled transitions, and the token game is not displayed; feedback has a different form, e.g., graphical animation or write to a file. Many steps are executed in a short time.

The *O-graph tool* supports generation, analysis, and drawing of O-graphs. Generation control is provided, allowing the user to focus on a certain aspect of a model, corresponding to generating only a *partial O-graph*, which is a subgraph of the complete O-graph. Analysis of a generated, partial or complete, O-graph is supported by *query functions*. Some *standard queries* are relevant for many models, e.g., to give generation statistics (number of nodes and arcs), list of dead markings, and information on liveness of transitions. Other queries depend on the model being investigated. Design/CPN provides a general query language implemented in Standard ML for that purpose.

Design/CPN was a cornerstone for this thesis. The tool played a major role in all five other projects, either focussing on further development (OS/Lamport), practical use (DistBETA, B&O), or underlying theory and algorithms (Equivalence, Backtrack).

The Design/CPN paper draws from experiences gained in the B&O project that was conducted as part of the work for this thesis. Therefore, the Design/CPN paper has some overlaps with the B&O paper.

The contribution of the Design/CPN paper was a description of an existing tool and provision of a recent important industrial example of use. In contrast to the other papers comprising this thesis, the Design/CPN paper is survey-like.

3.2 OS/Lamport

The OS/Lamport paper [51] describes a project in which the OS-tool [50] supporting the OS-graph method was developed, and integrated into Design/CPN.

The OS-tool allowed the first real practical experiment with OS-graphs to be conducted. With the OS-tool, correctness of Lamport's Fast Mutual Exclusion Algorithm [56] was established. A significant increase in the number of system states, which can be analysed using OS-graphs instead of O-graphs, was demonstrated. It was argued that the verification was more comprehensive and reliable than those that had been published previously, including Lamport's own.

The considered algorithm is used to ensure mutual exclusion between processes running in an environment with a single shared memory supporting atomic read and write operations, and where a number of CPUs are connected to a common bus.

Lamport's Fast Mutual Exclusion Algorithm, as a piece of pseudo-code, contains a for-loop that treats the processes in increasing order according to an associated integer identifier. This, in fact, does entail an asymmetry between otherwise symmetric processes. Therefore, to be able to apply OS-graphs, a generalised version of the algorithm had to be considered. It was necessary to impose a symmetry, not assumed in the original algorithm, between the processes. The generalised version replaces the original for-loop with a for-loop making a random selection between processes not yet treated.

The model of the generalised algorithm has a larger O-graph than the model of the original algorithm. Thus, even though OS-graphs yield big savings, in some cases the starting point for using them is worse than the starting point for using O-graphs. However, it is still worthwhile to use OS-graphs: For a fixed number of processes, the O-graph for the CPN model of the original algorithm is larger than the OS-graph for the CPN model of the generalised algorithm.

[51] is written to be self-contained: CP-nets and OS-graphs are informally explained and formally defined. To cater for the latter, [51] contains sections that have a high similarity with material from [45, 46].

The main contributions of the OS/Lamport project were to make the OS-tool available, and to provide a more comprehensive and reliable verification of Lamport's Fast Mutual Exclusion Algorithm than previously published. A minor contribution was an efficient algorithm to calculate the size of the

O-graph from the OS-graph [52] plus an exhibition of the usefulness of this algorithm to increase the confidence in the results of a verification using OS-graphs.

3.3 DistBETA

The DistBETA paper [54] describes a practical application of CP-nets and the Design/CPN tool to investigate a communication protocol. A considerable effort was put into modelling and simulation — a CPN model consisting of 12 modules was built. However, with respect to research in Petri Nets, the focus was on applying O-graph analysis and place invariant analysis.

A protocol [8], which supports distributed program execution in the object-oriented language BETA [60], was studied. An object on one computer may invoke a remote object, i.e., an object on another computer. The remote object invocation was modelled on the level of threads (lightweight processes) with emphasis on the competition for access to critical regions and shared resources.

The O-graph analysis established two crucial properties of the protocol — that it can never deadlock, and that each active object always will get a chance to do a remote object invocation. The O-graph analysis process also revealed that the configurations that could be analysed only had a few objects. A configuration with one sender object on one computer and one receiver object on another computer was formally verified, i.e., the basic communication scheme was proved correct. However, analysis of configurations with three or more objects was prohibited by the state explosion problem.

An attempt was made to apply structural net reductions [36] before the O-graph analysis, to increase the number of objects that could be analysed. Significant reductions in the sizes of the O-graphs were demonstrated, but the goal of being able to analyse three threads was not achieved.

The aim of the place invariant analysis was to increase the confidence in the CPN model that was built. Throughout the modelling process, the modellers had in mind a set of important properties that a sensible model should satisfy. Using place invariants, it was possible to prove that the final model actually had these properties. Most of the properties were quite similar. They stated that certain sets of tokens were constant. Another property said that a monitor construction was correct.

The main contributions of the DistBETA project were demonstrations of both the applicability and the limitations of O-graph analysis, structural

net reductions, and place invariant analysis on a real-world model. Minor contributions were discussions of how to create a proper model of a complex system, and how to modify a model to make it tractable for O-graph analysis.

3.4 B&O

The B&O paper [21] describes another practical application of CP-nets and Design/CPN, quite different from the DistBETA project. The latter was carried out by a small group of CPN researchers alone, while the B&O project involved engineers from a large industrial company, the renowned Danish manufacturer of audio/video systems Bang & Olufsen A/S (B&O). B&O wanted to investigate CP-nets as a way to improve their methods for specification, validation, and verification of communication protocols.

In the main experiment, an engineer from B&O, supervised by the author of this thesis, built and investigated a 13-module CPN model of a communication protocol used in the so-called BeoLink® (BeoLink) audio/video system. A BeoLink system connects the audio and video devices of a home in a network. This allows sharing of resources such that, e.g., a person can remotely use a CD player located in another room. The considered protocol is used to grant exclusive access to various services. The purpose is to prevent disorder, e.g., that track 11 is selected on a CD, if two users simultaneously request track 1.

The engineer validated the behaviour of the model using simulations. He relied heavily on the new functionality of Design/CPN to give feedback from simulations in terms of message sequence charts [18, 43], which display the message passing between the different components of a distributed system. Compared to watching the token game, the use of message sequence charts made the work of the engineer both more pleasant and more effective. Moreover, it enabled him to discuss results of simulations with colleagues unfamiliar with CP-nets. The engineer formally verified crucial properties of the model using the O-graph method, e.g., that a certain key used to ensure mutual exclusion is always generated, when a BeoLink system starts from scratch.

CP-nets was also used to examine important aspects of a possible future revision of the BeoLink system, and to check compatibility between the new and the existing version.

Based on this project, B&O concluded that CP-nets can be a useful aid for specification, validation, and verification of communication protocols in

the future.

After the B&O project, the process itself — as opposed to the resulting product — was analysed [23], with participation of this author. The analysis was based on [80], in which design is seen as a tentative interaction with materials (here, CP-nets), instead of an algorithm-like, constantly forward-moving, rational process. The analysis was based on a diary maintained by this author and tape-recordings of conversations during a work session. The perspective offered in [80] provided interesting insights on the work practices of the engineer.

Compared to the DistBETA project, the B&O project made a new contribution with respect to O-graph analysis. In the B&O project, the considered CPN model was *timed* [46], which posed a new and hard challenge with respect to O-graph analysis, because the O-graphs get infinite. Thus, it was necessary to focus on selected aspects, and use partial O-graphs to derive sensible verification results.

The B&O project benefitted from being a successor of the DistBETA project. The experiences gained in the latter, e.g., with respect to how to make a CPN model tractable for O-graph analysis, were highly valuable.

Besides the results with respect to O-graph analysis, another contribution of the B&O project was to demonstrate that it is indeed possible, within a reasonable time frame, to convey the ideas of CP-nets to industrial engineers, enabling them to work independently. Finally, the power of sophisticated graphical feedback from simulations of CP-nets was exhibited.

3.5 Equivalence

The Equivalence paper [53] describes an experiment with OE-graphs[4], the generalisation of OS-graphs presented at the end of Section 2.3.

The basic idea behind OS-graphs is to exploit symmetry. Symmetry is a structural, statical notion, based on permutation of similar components. Use of symmetry, in various disguises, is well-known in the research of verification of parallel and distributed systems, e.g., described in four papers in [28]. On the contrary, not much attention has been payed to OE-graphs in their full generality.

OE-graphs were originally presented in [46] as a theoretical generalisation of OS-graphs. The author of [46] noted that the experiences with practical

[4]In [53], OE-graphs are called state spaces with equivalence classes (SSEs).

use of OE-graphs were rather limited, and the examples given were all equivalences defined using only the structure of the systems under consideration. The usability of OE-graphs to capture equivalences which are, in a well-defined sense, dynamic was recognised for the first time in this project.

The motivation to write [53] came from the work with developing the OS-tool to support OS- and OE-graphs. The generality of OE-graphs allowed experiments with different kinds of equivalence relations. During these experiments, the usefulness of OE-graphs, not based on permutations as in OS-graphs, was realised. OE-graphs allow expression of equivalences that are dynamic, in the sense that they can capture that some information becomes irrelevant as the execution of a system progresses. The paper analysed a small protocol as example. The system under consideration has a sender and a receiver communicating over an unreliable network. The correctness of the protocol was formally established, i.e., it was proved that if the network looses only finitely many packets, the receiver will eventually receive, in the right order, all packets sent by the sender.

Definition of the equivalence relations for the OE-graph was based on the observation that during execution of the system, some packets may become similar: The network may contain copies of packets that have already been properly received. All packets of this kind are treated exactly the same way by the considered protocol, and thus the actual data content of such packets can be ignored.

The system was modelled with the capacity of the network as parameter. For different capacities, both OE- and O-graphs were generated, and statistics were presented to compare sizes and generation times. Use of OE-graphs yielded significant reductions and speed-ups; and enabled analysis of the system for larger values of the system parameter than with O-graphs.

The main drawback of general OE-graphs is that in order to apply them, the user must prove a consistency requirement. As demonstrated in the Equivalence paper, this may amount to manual conduction of a complex mathematical proof.

The contribution of the Equivalence project was to show that OE-graphs are useful to capture a more general notion of equivalence than one based on symmetry as in OS-graphs.

3.6 Backtrack

The Backtrack paper [49] describes a method for more efficient construction of OS-graphs. The method is justified, both by identifying an important general complexity property and by obtaining encouraging experimental performance measures.

A *self-symmetry* for a marking is a permutation symmetry that maps the marking to itself. Calculating the set of self-symmetries for each node inserted in an OS-graph allows for both fewer and faster equivalence tests, and, thus, aids more efficient construction of OS-graphs. The Backtrack paper suggests a new method, the *Backtrack Method*, for calculation of self-symmetries.

The work behind the Backtrack paper began with a literature search within the area of computational group theory. This led to the discovery of the existence of the *Backtrack Algorithm* [11], which is a general-purpose algorithm to search a permutation group for members satisfying a given property. The Backtrack Algorithm is the foundation of the Backtrack Method.

The *Backtrack Method* for calculation of self-symmetries treats the places of the given CP-nets in some order. The marking of each place, which is a multi-set, is split into a number of sets — one set for each positive coefficient appearing, containing all elements with that coefficient. As shown in [2], the set of self-symmetries for the marking is the intersection of the sets of self-symmetries for these sets. For each of these sets, the Backtrack Algorithm is applicable, because calculation of the set of self-symmetries for a set is a special instance of the general problem solved by the algorithm. The Backtrack Method for calculation of self-symmetries amounts to application of the Backtrack Algorithm to each of the sets derived from the marking combined with a computation of intersection.

The applicability of the Backtrack Method was tested using the general mathematics software package GAP [81], which contains an implementation of the Backtrack Algorithm. Encouraging experimental measures were obtained on two examples. A complexity property, referred to as the *fast tester property*, was discovered and proved. The fast tester property says that when a permutation group of size less than $n!$, for some natural number n, is searched for members satisfying a property which is constantly true, then the Backtrack Algorithm tests at most $n - 1$ permutations.

The fast tester property has a great impact on the success of the Backtrack Method. Because of deliberate redundancy in CP-nets, it is often the case

that after having treated only a few places, the set of self-symmetries looked for is actually computed. However, if it is not known that this is the case, computational resources will be wasted in trying to cut down on a search domain that cannot be reduced further. The Backtrack Algorithm is able to detect when no further reductions are possible very quickly, due to the fast tester property.

The Backtrack project illustrates well the value of experiments in computer science research. In the practical experiments with the Backtrack Method, this author made observations that seemed to indicate the fast tester property. However, it is, of course, not possible to conclude a general complexity property from a number of concrete performance measures. It is necessary also to study and understand the theory behind the algorithm under investigation, and subsequently prove that the expected property actually holds. But, without the experiments, an isolated theoretical study most likely would not have led this author to discover the fast tester property.

The contribution of the Backtrack project was the discovery of the existence of the Backtrack Algorithm, and suggestion and justification of the Backtrack Method.

4 State Space Analysis Methods

This section sums up main results of other researchers within the area of state space analysis methods. Sections 4.1 and 4.2 describe methods which are particular for High-level Petri Nets. Sections 4.3 to 4.6 describe methods which are based either in Petri Nets and/or other formalisms. This means that the viewpoint is widened: Instead of looking narrowly at *analysis of CP-nets by the occurrence graph method*, the more general area of *state space analysis of parallel and distributed systems* is considered.

With respect to analysis of CP-nets in particular and High-level Petri Nets in general, this section, together with Sections 2.2 and 2.3, do survey the most prominent state space methods. With respect to state space analysis of models based in other formalisms, this section does not give an overview of the state-of-the-art; it merely presents selected, related work, considered here because of potential, future influence on the analysis methods for CP-nets.

This section concludes (Section 4.7) with a comparison of the OS-graph method[5], one of the main foci of this thesis, with the other methods pre-

[5]With respect to OE-graphs, to the knowledge of this author, no similar idea has been

sented in this section. In particular, possible combinations are discussed. With respect to the O-graph method, all the described methods are, when applicable, improvements. Thus, a comparison would be trivial.

4.1 Parameterised Reachability Graphs

A *parameterised* or *symbolic reachability graph* [58] is a smaller graph containing the same information as the full reachability graph. The idea was originally proposed for Predicate/Transition Nets in [57]. Instead of having a node for each reachable marking of a considered Predicate/Transition Net, the nodes are assigned *parameterised markings*, which are symbolic expressions containing parameters or variables. A symbolic expression may be instantiated by assignment of concrete values to the appearing variables. In this way, each node represents a set of similar markings.

In the procedure to generate a symbolic reachability graph, occurrences of transitions are recorded symbolically, i.e., without assigning concrete values to the variables. The marking resulting from an occurrence is represented by an expression containing the variables of the involved transition.

In [78], the ideas of parameterised reachability graphs were transferred to another class of High-level Petri Nets, the *Algebraic Petri Nets* [75].

To the knowledge of this author, no implementation of the parameterised reachability graph method has been published, and a conclusion of [79] is that the technical details are quite difficult to manage.

4.2 Well-formed Coloured Nets

Well-formed Coloured Nets [15] resemble CP-nets, but have narrow syntactical restrictions on arc expressions, guards, and initial markings. These syntactical restrictions ensure that detection of symmetric markings is fully automatic, i.e., requires no user intervention. Given a Well-formed Coloured Net, a *symbolic reachability graph* can be constructed, in which symmetric markings are automatically lumped into one node.

Well-formed Coloured Nets belong to the class of *Stochastic Petri Nets* [61]. The purpose of creating a model in this class is not only to study logical behaviour, but also to study performance issues. Well-formed Coloured

published in the area of Petri Nets. However, it may well be that OE-graphs and the way they were applied in the Equivalence project, in some disguises, have been investigated within other formalisms.

Nets and their symbolic reachability graph arose from pursuing more efficient methods for Petri Net based performance evaluation: The complexity of the method applied to derive performance results, solution of Markovian chains, is strongly improved by using the symbolic reachability graph instead of the often much larger full state space of the considered model.

Well-formed Coloured Nets are supported by the *GreatGSPN* tool [14] from University of Turin, and have been applied successfully for analysis of real-world systems, see, e.g., [7, 16, 17].

4.3 Stubborn Sets

The *stubborn set method* [87, 88, 89] is developed originally for Petri Nets and later for process algebras[6]. The method effectively reduces the number of different interleavings of independent events that are represented in a state space. The basic version of the stubborn set method allows only detection of deadlocks and the existence of infinite execution sequences. More advanced versions preserving more properties exist, but in general, the stubborn set method throws away information.

The method is described for PT-nets in [87]: Instead of constructing a state space containing all reachable states of a system, the stubborn set method constructs a smaller one. In each state encountered in a state space generation, only occurrences of a subset of the enabled transitions are investigated, i.e., only a subset of the successor states are generated. The subset of transitions selected for occurrence is required to be *stubborn*, meaning, in the basic version, that no transition outside the set can change enabling and disabling of transitions included in the stubborn set. Stubborn sets allow occurrences of some transitions to be postponed until a later stage.

The stubborn set method is developed for CP-nets in [88]. The approach relies on the fact that any CP-net can be unfolded into an equivalent PT-net, as proved in [45]. The stubborn set method for a given CP-net simply amounts to application of the PT-net version of the method on the equivalent PT-net. This approach to analysis of CP-nets is, in general, not desirable. For many systems, unfolding entails a serious, if not devastating, complexity problem.

An implementation of the basic stubborn set method is in the *TORAS*

[6] *Process algebra* is a category of formalisms for modelling and analysis of parallel and distributed systems — see, e.g., [37, 64].

tool [4] from Telecom Australia Research Laboratories. More advanced versions are supported by the Predicate/Transition Net tool *PROD* [74] from Helsinki University of Technology. No tool support of stubborn sets for CP-nets exists. Implementation of such support would require heuristics for reasonably efficient calculation of useful stubborn sets for markings of CP-nets. Useful in the sense that the stubborn sets yield a decent reduction in the size of the generated state space. In general, in a given marking, many stubborn sets exist. E.g., the set of all enabled binding elements is trivially stubborn, but not a good choice, since obviously, it yields no reduction. On the other hand, there is no guarantee that the best choice is to try to calculate a minimal stubborn set. Heuristics for finding useful stubborn sets are required, and [88] contains some preliminary discussions of that subject.

As noted in [89], the stubborn set method has proven to yield dramatic reductions in the sizes of the state spaces for small regular systems, like the dining philosophers system and the alternating bit protocol. However, successful application of the method on real-world systems is still to come.

4.4 Compositionality and Modularity

Many systems are constructed in a modular way from smaller subsystems. It would be immensely attractive, if analysis results could be carried over from the subsystems to the entire system.

One proposal to do so is the *compositional state space analysis method* described in [86]. The basic idea is to define an equivalence relation on systems, and use this equivalence to construct a smaller system. The method has its origin in process algebra, where *labelled transition systems* (see, e.g., [64]) often are used to describe systems. In this case, there is no difference between a system and its state space. Thus, reduction of systems is the same as reduction of state spaces.

The choice of a suitable equivalence is a trade-off between the amount of reduction of the state space that can be obtained, and the information that a user wishes to preserve as discussed in [86]. The two most extreme cases, a system is equivalent with itself only, and all systems are equivalent, are not useful. The former allows no reduction while the latter preserves no information. Fortunately, there are many useful equivalences in between.

Given an equivalence relation and a system constructed from a number of subsystems, the compositional state space analysis method substitutes each subsystem with a smaller equivalent one. This is legal when the equivalence

relation is a congruence with respect to all the operators available to construct systems from subsystems. The method may be applied recursively, yielding smaller and smaller equivalents of the original system.

A problem, often thought to be inherent to state space analysis, is the following: If a system has a system parameter, e.g., the number of some component, then the results gained from a state space analysis depend on that parameter: If the parameter may take infinitely many values, the system cannot be verified fully with state space methods, because it would require generation of infinitely many state spaces. Using the compositional state space analysis method, [91] shows that this is not always true. As one example, the alternating bit protocol is verified for all (infinitely many) values of an integer system parameter, the maximal number of retransmissions of messages.

The compositional state space analysis method is supported by the *ARA* tool [90], developed by the Technical Research Centre of Finland in cooperation with other Finnish research institutions and companies. ARA is not a Petri Net tool. A system given as input to ARA must be described in the textual language Basic LOTOS [6]. ARA can optionally use stubborn sets for reduction, and has been applied successfully to verify small regular systems.

For Petri Nets, related approaches to compositional state space analysis are described in [68] (PT-nets) and [24] (CP-nets). In the *modular state space analysis* of [24], systems are constructed from subsystems by means of sharing of transitions. It is explained and proved how a state space for each of the subsystems, plus a so-called synchronisation graph, contain the same information as the ordinary state space for the entire system. No tool support of modular state space analysis for CP-nets exists.

4.5 Supertrace

The *Supertrace* technique [38] is motivated by the following basic philosophy: An approximation to verification is better than no verification at all — or equivalent, for the aware user, unreliable results are better than no results. The Supertrace technique, in general, does throw away information.

The idea behind the technique is the recognition that the state spaces, which a user wants to generate, usually are too large to fit the computer available. The Supertrace technique prescribes a way to get optimal use of memory: A hash table precisely fitting the available memory is allocated.

When a new state is encountered, a hash value is calculated. Instead of storing the states (and state changes) themselves, a bit is set in the hash table indicating a hit. If the hash function makes a uniform distribution of the states, this approach provides a random traversal of the state space.

If the hash function is perfect, i.e., there are no collisions, the Supertrace technique yields an ordinary exhaustive traversal of all reachable states. If the hash function is not perfect, the Supertrace technique cannot be used for verification in the rigourous sense of the word, but in this case it provides an approximation to verification.

Because nothing but hash values are stored, the traversal must necessarily rely on *on-the-fly* verification: Each time a new state is generated, it is checked if it has a certain property, e.g., if it is a deadlock. This is in contrast to the ordinary approach, in which a full state space is generated before it is used to derive any analysis results.

The Supertrace technique is implemented in the *SPIN* tool [38, 39] from AT&T Bell Labs. SPIN is a general tool for design, validation, and verification of parallel and distributed systems, supporting both simulation, full verification (when possible), and approximative verification exploiting the Supertrace technique. SPIN is not a Petri Net tool. A system to be analysed by SPIN must be given in the C-like textual language PROMELA [38], and properties must be specified in Linear Temporal Logic (LTL) [72]. SPIN is widely used, and [39] includes a long list of references to projects carried out in which SPIN was successfully used, e.g., [5, 83].

4.6 Binary Decision Diagrams

Binary Decision Diagrams (*BDDs*) [1] have proved to be a powerful aid for analysing systems with astronomical numbers of states. A BDD offers a compact representation of boolean functions [9], and the state space of a system can be encoded as a boolean function. Thus, BDDs provide a compact representation of state spaces [10, 63]. A state space represented as a BDD contains the same information as the full state space.

Assume that each state of a considered system can be described by a number, say N, of boolean variables. The boolean function representing the state space has 2N variables. The function yields true on an argument, if and only if the first N variables describe a state which has a state described by the last N variables as immediate successor.

A boolean function is represented as a graph. There is a root, and each

leaf is labelled with either true or false. Each node not being a leaf is labelled with a function variable, and has two outgoing edges — one for each of the two possible values of the variable. The value of the function on a given argument is equal to the label of the leaf, reached by a traversal of the graph, starting from the root and determined by the assignment of boolean values to the variables in the argument.

A boolean function has many different representations as a graph. However, if the ordering of the variables is fixed, a set of reduction rules can be applied to yield a unique, often small, canonical form [9]. The core of the idea is to lump and share identical subgraphs.

A software packet for manipulation of BDDs is provided by Carnegie Mellon University [59]. BDDs are believed to be particular useful for verification of very regular systems. Impressive case studies, most notably in the area of hardware designs, have been published, e.g., [27]. It is yet unknown whether the BDD technique can be effectively generalised to, of particular interest here, CP-nets with their very elaborated notion of state.

A method for using BDDs to represent state spaces for bounded PT-nets supplemented by encouraging experimental figures was published in [69]. The idea was pursued further and sustained by more experiments in [76].

4.7 Comparisons and Combinations with OS-graphs

Below, the state space analysis methods described above are compared with the OS-graph method. In particular, possible combinations are discussed.

A parameterised reachability graph and an OS-graph are essentially the same. Both preserve the full information that is contained in the ordinary full state space. The idea of having one node representing a set of similar or symmetric markings merely has different incarnations. Therefore, it does not make sense to combine the two methods.

Likewise, the ideas behind symbolic reachability graphs for Well-formed Coloured Nets and OS-graphs are also essentially the same. However, the former has an advantage: A meaningful application of the OS-graph method for a CP-net requires that a consistent permutation symmetry specification is given. It is up to the user to ensure the consistency. It can be done by examining all the arc expressions, guards, and initial markings appearing in the considered CP-net. [46] describes ideas for computer support to aid this task, but for now, it must be solved manually. In contrast, detection of symmetries in Well-formed Coloured Nets is fully automated thus effectively

eliminating the need of conducting a consistency proof. The drawback of Well-formed Coloured Nets compared to CP-nets is the narrow syntactical restrictions of the former.

The stubborn set method and the OS-graph method have the same goal: Instead of constructing a state space containing all reachable states of a system, a smaller one is constructed. The criteria for leaving out a reachable state in the two methods are very different. In fact, the criteria are believed [45, 88] to be independent in the sense that a simultaneous use of the two methods is possible and worthwhile in general. This claim is substantiated by two small examples in [45] and [88], respectively. One of the examples demonstrates how a state space of exponential size in some system parameter drops to quadratic size when either OS-graphs or stubborn sets are applied; and drops to linear size when the two methods are combined. Of course, a combination of methods cannot preserve more information than the weakest contributor: If OS-graphs, which preserve full information about a system, are combined with the basic version of stubborn sets, which only preserve deadlocks and the possibility of an infinite execution sequence, at most the latter properties can be preserved. Fortunately, these properties are in fact preserved as proved in [88].

The compositional and modular state space analysis methods seem to combine well with the idea of OS-graphs, with the intuitive argument given in [24], that the approaches rely on independent characteristics of a considered system.

The Supertrace technique may be combined with the idea of symmetry of OS-graphs. If it is undesirable to use memory to record that more than one representative of a set of symmetric states have been encountered, then a hash function must be chosen that maps symmetric states to the same value.

BDDs together with symmetries have been exploited to design a model checking algorithm [26] of formulas in the temporal logic CTL* [25, 29]. With respect to symmetries, the basic ideas of this approach are to a large extent a reinvention of the ideas behind OS-graphs. The model checking algorithm is implemented in the *SMV* tool [63] from Carnegie Mellon University. BDDs have pushed the limits on the number of system states, which can be analysed, several magnitudes. What must be remembered, of course, is that a state is a relative notion. It does matter whether a state can be described with, e.g., a small number of boolean variables, or it takes a huge number of variables with very complex data types. Any number of bits may be required to store one state of a system in the memory of a computer. It

takes a detailed look at a proposed method to judge its quality — impressive experimental figures do not suffice. A recent experiment [19] describing verification of a small communication protocol sustains that claim: It is shown that the OS-tool can analyse a system with 10^{288} states in about twenty minutes. This is certainly not an interesting result in itself. First of all, because a state is a relative notion as noted above. Secondly, because the 10^{288} states correspond to the value 600 of a system parameter, and really, if it is possible to analyse this highly regular system for small values of the system parameter, it is debatable what insight is gained by going far beyond that. In other words, being able to verify, e.g., the dining philosophers system for, say, 1000 instead of 10 philosophers, is not really interesting. What matters is verification of truly complex systems.

5 Occurrence Graph Analysis Projects

This section discusses projects, which attempted to verify real-world systems. In comparison with the widened viewpoint of Section 4, the perspective is again narrowed to *analysis of CP-nets by the occurrence graph method*, where the main projects, which have been documented, are considered. In the more general area of *state space analysis of parallel and distributed systems*, a plethora of projects have been carried out, and it is neither possible nor adequate to try to provide an overview of these here.

All the projects discussed below applied CP-nets, the Design/CPN tool, and O-graph analysis. The OS/Lamport and Equivalence projects, done as part of the work for this thesis, are, to the knowledge of this author, the first projects which applied OS- and OE-graphs.

In Section 5.1, the projects are discussed in chronological order. The first two projects were described in papers published in 1992, the third in 1994, and the last two in 1996.

This section concludes (Section 5.2) with a comparison of the discussed projects with the DistBETA and B&O projects, carried out for this thesis.

5.1 Other Projects

Verification of a hardware design was described in [35]. A CPN model of a so-called arbiter cascade was created and verified. The dependency of the initial marking inherent to occurrence graph analysis mentioned in Section 4.4 was

elegantly solved: The system was characterised with one single integer system parameter d. For $d = 1$, the model was verified directly using an O-graph. Mathematical induction established the proof for all values $d > 1$. Using occurrence graphs in conjunction with induction is very appealing, whenever applicable.

Analysis of an Ada program was described in [62]. A large 30-module CPN model was considered. Due to the state explosion problem, it was not possible to generate a complete O-graph for the entire model. As an alternative, O-graph analysis was used for three purposes. First, to reconfirm correctness of calculations that had been made with paper and pen prior to having tool support. Second, to discover errors in the design, as with simulations; and, third, to investigate much simplified versions of the considered model. According to [62], the versions that could be analysed were not at all proper reflections of the considered system. The project was mainly a demonstration and expected recognition of the power of tool-supported as opposed to paper-and-pen O-graph analysis.

Analysis of control procedures at a chemical plant was described in [32]. The models in this project were created as Predicate/Transition Nets. However, with respect to modelling, Predicate/Transition Nets and CP-nets are similar. Therefore, it was possible to create and analyse the models with the CP-net tool Design/CPN. The main part of the paper dealt with creation and simulation of models; O-graph analysis was briefly mentioned in the conclusion. The generation and analysis of one small O-graph was reported. Also here, O-graphs were primarily used to find errors in the system.

Design of an alarm system by a small Danish company was described in [73]. Again, O-graphs were used primarily for debugging, but also to verify a number of very small configurations of the considered system.

Design of a networks gateway for the Australian Defence Force was described in [30]. O-graphs were used to verify selected important parts; not to verify the entire model.

5.2 Comparisons with Thesis Projects

Below, the projects described above are compared with the DistBETA and B&O projects.

In the hardware design project, induction was applicable because the considered model was quite simple and very regular. It will be difficult in general to apply induction to complex communication protocols, like the

ones that have been considered in this thesis. The B&O paper discusses the complications of a concrete attempt.

Both in the Ada program project, the control procedure project, and the alarm system project, O-graphs were primarily used to debug and to discover problems in the proposed models, i.e., to derive negative analysis results. In contrast, in the networks gateway project, as well as the DistBETA and B&O projects, O-graphs were primarily used for verification, i.e., to derive positive analysis results. Even though the results were obtained for small configurations and/or limited behaviours of the considered models, the scenarios considered had a size and relevance such that the confidence in the protocols as such was greatly increased by the O-graph analysis.

An important difference between all the projects described in Section 5.1 and the B&O project was that in the latter, from some point, the O-graph analysis was mainly carried out by industrial engineers. In the other projects, the O-graph analysis was done by a research group alone. In fact, the B&O project demonstrated that it is possible, within a reasonable time frame, to successfully teach the O-graph method to engineers from the industry.

6 Conclusions

After the description and discussion of related work in the previous sections, the focus again turns to this thesis.

When the work for this thesis started in 1993, the support for occurrence graph analysis in the Design/CPN tool was very new. The only documented projects, which had used it, were the first two described in Section 5.1. In both, quite small CP-nets were considered, only O-graphs were used, and in one of the projects, there was a serious discrepancy between the CP-net modelling the real system, and the CP-net that was analysed.

The status of occurrence graph analysis at the time of writing this overview paper, in the beginning of 1997, is that the support for occurrence graph analysis in Design/CPN is much matured. The O-graph method is better supported. In addition, a number of larger projects using O-graphs have been carried out, which prove that the method, in spite of the state explosion problem, may be really useful, if sensibly applied.

Moreover, the OS-graph method has been implemented, and, thus, made available. It has been recognised, as expected, that OS-graphs allow analysis of systems with orders of magnitude more states than O-graphs. On the

other hand, some inherent obstacles for application of OS-graphs have been identified. The problem with the for-statement in the OS/Lamport project, discussed in Section 3.2, is more general than explicitly stated there. Consider an algorithm, containing a standard iteration control-structure like while or repeat, which manipulates some component in each iteration. A concrete implementation of this algorithm, in a programming language not supporting non-deterministic choice, will have to fix a certain ordering in which to treat the components. This may, in fact, entail an asymmetry between otherwise symmetric components. In this case, OS-graphs are suitable for verification of the abstract algorithm, formulated using non-deterministic choices; not for verification of a concrete implementation of the algorithm.

Finally, the usability of OE-graphs for verification of systems possessing a certain kind of equivalence, which is believed to appear frequently, has been identified: OE-graphs can capture that some components of a system become equivalent dynamically, i.e., as the execution of a system progresses — in contrast to OS-graphs, which can only capture that components are always, or statically, equivalent.

Possible extensions

Of course, much more effort must be put into the research of occurrence graph analysis of CP-nets in the future. A subject of particular interest is to test the applicability of OS- and OE-graphs on models of larger systems. The DistBETA and B&O projects may serve as starting points.

Although significant verification results were obtained, using O-graphs, in both projects, the DistBETA project revealed that the state explosion problem really is a practical hindrance. It would certainly have been valuable to be able to verify larger configurations of the considered model, than was actually possible. Most likely, OS-graphs had been a viable means to do so. So, why were OS-graphs not applied in the DistBETA project? When the project was carried out, the OS-tool did not exist. On the other hand, it would, of course, have been possible to revisit the considered model after the OS-tool had been developed. However, this has been postponed until the next version of the OS-tool is available, for the reason described below.

In the current version of the OS-tool, the user must provide two predicates describing the equivalence relations on markings and binding elements, respectively. Although the OS-tool includes a number of auxiliary data structures and utility functions to help the user with this task, still, many lines of

code must be written manually to implement the predicates. Therefore, it is time-consuming, cumbersome, and error-prone to verify large CPN models.

In the next version of the OS-tool, the two equivalence predicates will be compiled automatically. All the user has to do, is to attach a symmetry group to each declaration of an atomic colour set, e.g., attach a keyword like *allperm* to allow all permutations, or a keyword like *noperm* to allow no permutations. A project [3] to provide such a compiler has started, co-supervised by the author of this thesis.

In the B&O project, the CPN model analysed was timed, and, thus, had an infinite occurrence graph. In spite of that, it was possible to obtain important verification results with partial O-graphs. On the other hand, it would be highly valuable, if there was a way to define equivalence relations on markings and binding elements of a timed CP-net, such that an infinite graph was collapsed into a finite one. With the generality offered by OE-graphs, this may turn out to be possible. However, for now it is only speculation. Experiments are necessary to investigate the possibilities.

Summary of main contributions

A number of developments within occurrence graph analysis of CP-nets within the last few years can be attributed to the work done for this thesis. Steps towards practical application of the occurrence graph method, and, thus, verification of CP-nets, were actually taken.

The contributions of the thesis were already discussed in Section 3. The main ones are repeated below:

- Description of the Design/CPN tool as it appears in the beginning of 1997.
- Development of the OS-tool, advancing the OS-graph method from being theoretically promising to practically applicable, with the limitations regarding the current version discussed above.
- Demonstration of the applicability of the OS-graph method in providing a more comprehensive and reliable verification of Lamport's Fast Mutual Exclusion Algorithm than previously published.
- Creation of a CPN model of the protocol supporting distributed program execution in the BETA language.

- Demonstration of both the applicability and the limitations of the O-graph method and the place invariant method on the real-world CPN model from the DistBETA project.
- Establishment of CP-nets as a means for specification, validation, and verification of communication protocols at Bang & Olufsen A/S.
- Demonstration of the practical applicability of O-graph analysis on the timed real-world CPN model from the B&O project.
- Recognition of OE-graphs as a means to capture that components become equivalent dynamically, i.e., as the execution of a system progresses.
- Suggestion of the new Backtrack Method for calculation of self-symmetries to aid more efficient construction of OS-graphs. Justification of the suggestion. Theoretically by discovering an important complexity property, empirically by conducting performance tests on two examples.

Acknowledgements

The author wishes to thank:

Kurt Jensen and Søren Christensen for indispensable supervision, cooperation, and inspiration throughout this PhD study.

Lars Michael Kristensen and Kjeld Høyer Mortensen for fruitful cooperation, not the least in co-authoring various papers.

Rikke Drewsen Andersen, Vincent Becuwe, Jørgen Brandt, Søren Brandt, Allan Cheng, Afshin Foroughipour, Torben Bisgaard Haagh, Jesper Gulmann Henriksen, Thomas Hildebrandt, Peter Huber, Ludovic Joly, Kristian Lund, Kim Halskov Madsen, René Wenzel Schmidt, Robert Shapiro, Alexandre Valente Sousa, Kim Sunesen, Niels Toft Sørensen, and Jan Toksvig for various efforts — please refer to the individual papers for details.

Antti Valmari for hosting a nice and rewarding visit to Tampere University of Technology during this PhD study.

Preben Holst Mogensen for reading and commenting this overview paper.

Susanne Brøndberg for proof-reading.

The work done for this thesis has been supported by grants from the Danish Research Councils SNF and STVF, from the Faculty of Science at University of Aarhus, and from University of Aarhus Research Foundation.

References

[1] S.B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers, Vol. C-27, 6*, 1978.

[2] R.D. Andersen, J.B. Jørgensen, and M. Pedersen. Occurrence Graphs with Equivalent Markings and Self-symmetries. Master's thesis, Computer Science Department, University of Aarhus, Denmark, 1991. Only available in Danish: Tilstandsgrafer med ækvivalente mærkninger og selvsymmetrier.

[3] V. Becuwe and L. Joly. An Improved Interface for Permutation Symmetry Specifications for Coloured Petri Nets. Technical report, Computer Science Department, University of Aarhus, Denmark, 1996.

[4] J. Billington, A. Valmari, and G. Wheeler. Baby TORAS Eats Philosophers but Thinks about Solitaire. In *Proceedings of the 5th Australian Software Engineering Conference, Sydney, Australia*, 1990.

[5] B. Boigelot and P. Godefroid. Model Checking in Practice: An Analysis of the ACCESS Bus Protocol Using SPIN. In *Proceedings of Formal Methods Europe, Oxford, UK*, volume 1051 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[6] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 1987.

[7] O. Botti, S. Donatelli, and G. Franceschinis. SWN Models of Parallel Architectures in the Field of Plant Automation Systems. Case Studies Within the 16th International Conference on Application and Theory of Petri Nets, Turin, Italy, 1995.

[8] S. Brandt and O.L. Madsen. Object-Oriented Distributed Programming in BETA. In R. Guerraoui, O.M. Nierstrasz, and M. Riveill, editors, *Object-Based Distributed Programming*, volume 791 of *Lecture Notes in Computer Science*, Kaiserslautern, Germany, 1993. Springer-Verlag.

[9] R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers, Vol. C-35, 8*, 1986.

[10] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of the 5th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1990.

[11] G. Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[12] L. Cherkasova, V. Kotov, and T. Rokicki. On Net Modelling of Industrial Size Concurrent Systems. In M. Ajmone Marsan, editor, *Proceedings of the 14th International Conference on Application and Theory of Petri Nets, Chicago, Illinois, USA*, volume 691 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[13] L. Cherkasova, V. Kotov, and T. Rokicki. On Scalable Net Modeling of OLTP. In *Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse, France*. IEEE Computer Society Press, 1993.

[14] G. Chiola. GreatSPN 1.5 Software Architecture. In *Proceedings of the 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Turin, Italy*, 1990.

[15] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On Well-Formed Coloured Nets and Their Symbolic Reachability Graph. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*. Springer Verlag, 1991.

[16] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed Coloured Nets and Multiprocessor Modelling Applications. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*. Springer Verlag, 1991.

[17] G. Chiola, R. Gaeta, and M. Sereno. Modelling and Simulation of a Communication Protocol by SWN. In *Petri Nets Applied to Protocols, Proceedings of a Workshop of the 16th International Conference on Application and Theory of Petri Nets, Turin, Italy*, 1995.

[18] S. Christensen. *Design/CPN Message Sequence Charts Library Manual*. Computer Science Department, University of Aarhus, Denmark. Online: `http://www.daimi.aau.dk/designCPN/`.

[19] S. Christensen. We can Handle State Spaces of Sizes up to 10^{288}. Electronic correspondence on the Design/CPN mailing list, July 1996, `http://www.daimi.aau.dk/designCPN/email/p/post960719a.txt`.

[20] S. Christensen and L.O. Jepsen. Modelling and Simulation of a Network Management System using Hierarchical Coloured Petri Nets. In E. Mosekilde, editor, *Proceedings of the 1991 European Simulation Multiconference*, Copenhagen, Denmark, 1991. Society for Computer Simulation.

[21] S. Christensen and J.B. Jørgensen. Analysis of Bang and Olufsen's BeoLink® Audio/Video System Using Coloured Petri Nets. In P. Azéma and G. Balbo, editors, *Proceedings of the 18th International Conference on Application and Theory of Petri Nets, Tolouse, France*, Lecture Notes in Computer Science. Springer Verlag, 1997. To appear. Also available as DAIMI PB-514, ISSN 0105-8517, February 1997.

[22] S. Christensen, J.B. Jørgensen, and L.M. Kristensen. Design/CPN — A Computer Tool for Coloured Petri Nets. In E. Brinksma, editor, *Proceedings of the Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Twente, The Netherlands*. Springer-Verlag, 1997. To appear. Also available as DAIMI PB-511, ISSN 0105-8517, February 1997.

[23] S. Christensen, J.B. Jørgensen, and K.H. Madsen. Design as Interaction with Computer Based Materials. *Submitted to DIS 97, the 1997 Conference on Designing Interactive Systems, Amsterdam, the Netherlands*, 1997.

[24] S. Christensen and L. Petrucci. Modular State Space Analysis of Coloured Petri Nets. In G. de Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Turin, Italy*, volume 935 of *Lecture Notes in Computer Science*. Springer Verlag, 1995.

[25] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems, Vol. 8, No. 2*, 1986.

[26] E.M. Clarke, T. Filkorn, and S. Jha. Exploiting Symmetries in Temporal Model Logic Model Checking. In C. Courcoubetis, editor, *Proceedings of the 5th International Conference on Computer Aided Verification, Elounda, Greece*, volume 697 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[27] E.M. Clarke, O. Grumberg, H. Hiraishi, D.E. Long, K.L. McMillan, and L.A. Ness. Verification of the Futurebus+ Cache Coherence Protocol. In D. Agnew, L. Claesen, and R. Camposano, editors, *Proceedings of the 11th International Symposium on Computer Hardware Description Languages and their Applications, Ottawa, Ontario, Canada*, volume 32 of *IFIP Transactions A: Computer Science and Technology*. North-Holland, 1993.

[28] E.A. Emerson (editor). Formal Methods in System Design, Volume 9, Numbers 1/2 — Special Issue on Symmetry in Automatic Verification. Kluwer Academic Publishers, 1996.

[29] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic. In *Proceedings of the 10th Annual Symposium on Principles of Programming Languages, Austin, Texas, USA*, 1983.

[30] D.J. Floreani, J. Billington, and A. Dadej. Designing and Verifying a Communications Gateway Using Coloured Petri Nets and Design/CPN. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan*, volume 1091 of *Lecture Notes in Computer Science*. Springer Verlag, 1996.

[31] H.J. Genrich. Predicate/Transition Nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*. Springer Verlag, 1991.

[32] H.J. Genrich, H.-M. Hanisch, and K. Wöllhaf. Verification of Recipe-based Control Procedures by Means of Predicate/Transition Nets. In R. Valette, editor, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain*, volume 815 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

[33] H.J. Genrich and K. Lautenbach. The Analysis of Distributed Systems by Means of Predicate/Transition Nets. In G. Goos and J. Hartmanis, editors, *Semantics of Concurrent Computation*, volume 70 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[34] H.J. Genrich and K. Lautenbach. System Modelling with High-level Petri Nets. In *Theoretical Computer Science 13*. North-Holland, 1981.

[35] H.J. Genrich and R.M. Shapiro. Formal Verification of an Arbiter Cascade. In K. Jensen, editor, *Proceedings of the 13th International Conference on Application and Theory of Petri Nets, Sheffield, UK*, volume 616 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[36] S. Haddad. A Reduction Theory for Coloured Nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets, Theory and Application*. Springer-Verlag, 1991.

[37] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[38] G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall International Editions, 1991.

[39] G.J. Holzmann. An Overview of the SPIN Model Checker. Technical report, Computing Principles Research Department, Bell Laboratories 2C-521, Murray Hill, New Jersey, USA, 1996.

[40] P. Huber, A.M. Jensen, L.O. Jepsen, and K. Jensen. Reachability Trees for High-level Petri Nets. In *Theoretical Computer Science 45*. North-Holland, 1986.

[41] P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[42] P. Huber and V.O. Pinci. A Formal Executable Specification of the ISDN Basic Rate Interface. In *Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Aarhus, Denmark*, 1991.

[43] International Telecommunication Union — Telecommunication Standardization Sector (ITU-T). ITU-T Recommendation Z.120: Message Sequence Charts, Geneva, Switzerland, 1993.

[44] K. Jensen. Coloured Petri Nets and the Invariant Method. In *Theoretical Computer Science 14*. North-Holland, 1981.

[45] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 1992.

[46] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Vol. 2, Analysis Methods*. Monographs in Theoretical Computer Science. Springer-Verlag, 1994.

[47] K. Jensen. Condensed State Spaces for Symmetrical Coloured Petri Nets. *Formal Methods of System Design, Vol. 9, 1/2*, 1996. Special Issue on Symmetry in Automatic Verification. Kluwer Academic Publishers.

[48] K. Jensen, S. Christensen, P. Huber, and M. Holla. *Design/CPN Reference Manual*. Computer Science Department, University of Aarhus, Denmark.
Online: `http://www.daimi.aau.dk/designCPN/`.

[49] J.B. Jørgensen. Construction of Occurrence Graphs with Permutation Symmetries Aided by the Backtrack Method. Technical report, Computer Science Department, University of Aarhus, Denmark, 1997. DAIMI PB-516, ISSN 0105-8517, February 1997.

[50] J.B. Jørgensen and L.M. Kristensen. *Design/CPN OS-Graph Manual*. Computer Science Department, University of Aarhus, Denmark.
Online: `http://www.daimi.aau.dk/designCPN/`.

[51] J.B. Jørgensen and L.M. Kristensen. Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries. *Submitted to IEEE Transactions on Parallel and Distributed Systems*, 1996. Also available as DAIMI PB-512, ISSN 0105-8517, February 1997.

[52] J.B. Jørgensen and L.M. Kristensen. Efficient Calculation of the Size of the O-Graph from the OS-Graph. Technical report, Computer Science

Department, University of Aarhus, Denmark, 1996.
Online: `http://www.daimi.aau.dk/designCPN/`.

[53] J.B. Jørgensen and L.M. Kristensen. Verification by State Spaces with Equivalence Classes. *Submitted to the Ninth Conference on Computer-Aided Verification, Haifa, Israel*, 1997. Also available as DAIMI PB-515, ISSN 0105-8517, February 1997.

[54] J.B. Jørgensen and K.H. Mortensen. Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan*, volume 1091 of *Lecture Notes in Computer Science*. Springer Verlag, 1996. Also available as DAIMI PB-513, ISSN 0105-8517, February 1997.

[55] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem — Its Structural Complexity*. Birkhauser Boston, 1993.

[56] L. Lamport. A Fast Mutual Exclusion Algorithm. In *ACM Transactions on Computer Systems, Vol. 5, No. 1*. Association for Computing Machinery, 1987.

[57] M. Lindqvist. Parameterized Reachability Trees for Predicate/Transition Nets. *Acta Polytechnica Scandinavica, Mathematics and Computer Science Vol. 54, Helsinki, Finland*, 1989.

[58] M. Lindqvist. Parameterized Reachability Trees for Predicate/Transition Nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets, Theory and Application*. Springer-Verlag, 1991.

[59] D. Long. *A Binary Decision Diagram Packet*. School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1993.

[60] O.L. Madsen, B. Møller-Pedersen, and K. Nygaard. *Object-Oriented Programming in the BETA Programming Language*. Addison Wesley, 1993.

[61] M. Ajmone Marsan, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley and Sons, 1995.

[62] W.M. McLendon and R.F. Vidale. Analysis of an Ada System Using Coloured Petri Nets and Occurrence Graphs. In K. Jensen, editor, *Proceedings of the 13th International Conference on Application and Theory of Petri Nets, Sheffield, UK*, volume 616 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[63] K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[64] R. Milner. *Communication and Concurrency.* Prentice-Hall International Series in Computer Science. Prentice-Hall, 1989.

[65] R. Milner, R. Harper, and M. Tofte. *The Definition of Standard ML.* MIT Press, 1990.

[66] K.H. Mortensen and V. Pinci. Modelling the Work Flow of a Nuclear Waste Management Program. In R. Valette, editor, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain*, volume 815 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[67] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proceedings of the IEEE, Vol. 77, No. 4*. IEEE Computer Society, 1989.

[68] T. Murata and M. Notomi. Hierarchical Rechability Graphs of Bounded Petri Nets for Concurrent-Software Analysis. *IEEE Transactions on Software Engineering, Vol. 20, No. 6*, 1994.

[69] E. Pastor, O. Roig, J. Cortadella, and R.M. Badia. Petri Net Analysis Using Boolean Manipulation. In R. Valette, editor, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain*, volume 815 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[70] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für Instrumentelle Mathematik, Bonn, Germany, 1962. English translation: Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, USA, Vol. 1, Suppl. 1, 1966.

[71] V.O. Pinci and R.M. Shapiro. An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 524 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[72] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18nd IEEE Symposium on Foundations in Computer Science, Providence, Rhode Island, USA*, 1977.

[73] J.L. Rasmussen and M. Singh. Designing a Security System by Means of Coloured Petri Nets. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan*, volume 1091 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[74] M. Rauhamaa and K. Varpaaniemi. The Stubborn Set Method in Practice. In K. Jensen, editor, *Proceedings of the 13th International Conference on Application and Theory of Petri Nets, Sheffield, UK*, volume 616 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[75] W. Reisig. Petri Nets and Algebraic Specifications. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets, Theory and Application*. Springer-Verlag, 1991.

[76] O. Roig, J. Cortadella, and E. Pastor. Verification of Asynchronous Circuits by BDD-based Model Checking of Petri Nets. In G. de Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Turin, Italy*, volume 935 of *Lecture Notes in Computer Science*. Springer Verlag, 1995.

[77] G. Scheschonk and M. Timpe. Simulation and Analysis of a Document Storage System. In R. Valette, editor, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain*, volume 815 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[78] K. Schmidt. Parameterized Reachability Trees for Algebraic Petri Nets. In G. de Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Turin,*

Italy, volume 935 of *Lecture Notes in Computer Science*. Springer Verlag, 1995.

[79] K. Schmidt. *Symbolic Analysis Methods for Algebraic Petri Nets*. PhD thesis, Institut für Informatik, Humboldt-Universität zu Berlin, Germany, 1996.

[80] D. Schön. *The Reflective Practitioner*. New York Basic Books, 1983.

[81] M. Schönert. *GAP - Groups, Algorithm and Programming. A Reference Manual for GAP, Version 3.1*. Lehrstuhl für Mathematik, RWTH Aachen, Germany, 1992.

[82] R.M. Shapiro. Validation of a VLSI Chip Using Hierarchical Coloured Petri Nets. *Journal of Microelectronics and Reliability, Special Issue on Petri Nets*, 1991.

[83] S. Shukla, D.J. Rosenkrantz, and S.S. Ravi. Simulation and Validation of Self-stabilizing Protocols. In *Proceedings of the 2nd SPIN Workshop, New Brunswick, New Jersey, USA*, 1996.

[84] J. Toksvig. Design and Implementation of a Place Invariant Tool for Coloured Petri Nets. Master's thesis, Computer Science Department, University of Aarhus, Denmark, 1994.

[85] J.D. Ullman. *Elements of ML Programming*. Prentice Hall, 1993.

[86] A. Valmari. Compositional State Space Generation. In *Proceedings of the 11th International Conference on Application and Theory of Petri Nets, Paris, France*, 1990.

[87] A. Valmari. Stubborn Sets for Reduced State Space Generation. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[88] A. Valmari. Stubborn Sets of Coloured Petri Nets. In *Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Aarhus, Denmark*, 1991.

[89] A. Valmari. State of the Art Report: Stubborns Sets. Notes from Advanced Theory Tutorial on Construction and Analysis of Condensed

State Spaces, a Workshop of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain, 1994.

[90] A. Valmari, J. Kemppainen, M. Clegg, and M. Levanto. Putting Advanced Reachability Analysis Techniques Together: the ARA Tool. In J.C.P. Woodcock and P.G. Larsen, editors, *Proceedings of the First International Symposium of Formal Methods Europe, Odense, Denmark*, volume 670 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.

[91] A. Valmari and I. Kokkarinen. Unbounded Verification Results by Finite-State Compositional Techniques: 10^{any} States and Beyond. In *Proceedings of the ONR Workshop on Automated Formal Methods, Oxford, UK*, 1996.