Coloured Petri Nets — a Pragmatic Formal Method for Designing and Analysing Distributed Systems

Kjeld Høyer Mortensen

University of Aarhus, Computer Science Department
Ny Munkegade, Bldg. 540, DK–8000 Aarhus C, Denmark
khm@daimi.aau.dk

A distributed system is one that stops you from getting any work done when a machine you've never even heard of crashes.

— Leslie Lamport (Attributed)

Beware of bugs in the above code; I have only proved it correct, not tried it.

— Donald Knuth

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.

— C.A.R. Hoare

Abstract

The thesis consists of six *individual* papers, where the present paper contains the mandatory overview, while the remaining five papers are found separately from the overview. The five papers can roughly be divided into three areas of research, namely case studies, education, and extensions to the CPN method.

The primary purpose of the PhD thesis is to study the pragmatics, practical aspects, and intuition of CP-nets viewed as a formal method for describing and reasoning about concurrent systems. The perspective of pragmatics is our leitmotif, but at the same time in the context of CP-nets it is a kind of hypothesis of this thesis. This overview paper summarises the research conducted as an investigation of the hypothesis in the three areas of case studies, education, and extensions.

The provoking claim of pragmatics should not be underestimated. In the present overview of the thesis, the CPN method is compared with a representative selection of formal methods. The graphics and simplicity of semantics, yet generality and expressiveness of the language constructs, essentially makes CP-nets a viable and attractive alternative to other formal methods. Similar graphical formal methods, such as SDL and Statecharts, typically have significantly more complicated semantics, or are domain-specific languages.

The research conducted in this thesis, opens a new complex of problems. Firstly, to get wider acceptance of CP-nets in industry, it is important to identify fruitful areas for the effective introduction of the CPN method. Secondly, it would be useful to identify a few extensions to the CPN method inspired by specific domains for easier adaption in industry. Thirdly, which analysis methods do future systems make use of?

Acknowledgements

First of all I wish to thank my supervisor, Kurt Jensen, who is an unusual responsible and dedicated person in the task of educating people. He also gave me the unique chance to work at Meta Software Corporation in Boston. Thanks also to Kurt for introducing me to the challenging art of orienteering running.

The papers and ideas in the present thesis have been influenced by many different sources. Being part of the CPN-group and DEVISE (Centre for Experimental Systems Development) throughout my PhD, there has always been an apparently infinite stream of resources, "strange" ideas, discussions, arguments, and social activities — in summary an inspiring environment within the Computer Science Department, DAIMI. The CPN catalysts constitute Kurt Jensen, Søren Christensen, Jens Bæk Jørgensen, and Lars Michael Kristensen.

One cannot underestimate inspiring influences from external sources. Sharing office one semester with Charles Lakos from Tasmania, provided new perspectives on Petri Nets. BRICS also provided international breaths of fresh air via schools and visitors. Visiting foreign places induced provoking ideas and provided the essential opportunity for meeting new people via conferences, workshops, and schools. I joined, among other events, the International Petri Nets Conferences in Spain, Italy, and Japan; the Advanced Course on Petri Nets in Dagstuhl, Germany; the Workshop on Discrete Event Systems, Edinburgh, UK; the REX School, Noordwijkerhout, The Netherlands; and also shorter visits to Humboldt-Universität, Berlin and Technische Universiteit, Eindhoven. Everyone conveniently connected in the inevitable and ubiquitous Internet.

Thanks to the following persons who have reviewed earlier versions of this overview paper: Søren Christensen, Henrik Bærbak Christensen, Kurt Jensen, Lars Michael Kristensen, Charles Lakos, Jawahar Malhotra, and Peter Ørbæk.

My patient and understanding dear parents provided important support and faith — even though they, today, still don't know exactly what I was studying.

My creative friend and Meta Software colleague, Jawahar Malhotra, introduced me to restaurant life in Boston, and even Århus while doing his PhD.

Last but not least; I'm grateful for the remarkable patience and support of Lene, who I wish I had met earlier.

The work accomplished in this PhD thesis has been supported by grants from University of Aarhus, and the Danish Research Councils STVF and SNF.

Contents

Overview

In this paper I provide the compulsory overview of the PhD work accomplished throughout the past four years of research, with Kurt Jensen as supervisor. This thesis constitutes six *individual* papers, one of which is the present overview paper. The remaining five papers are divided into the following research areas: Two papers are case studies ([78] and [61]), one is related to educational issues ([21]), and two propose extensions to the CPN method ([20] and [14]). These five papers are summarised in this overview, and they are published separately as individual papers. They are included in the thesis without modifications (with the exception of layout). Detailed publication information can be found in Sect. 4.

1 Introduction

The title of this thesis suggests involvement of a number of different topics. The main topic is, of course, Coloured Petri Nets as the primary research area. (The term Coloured Petri Nets is henceforth abbreviated as CP-nets or CPN.) Our application domain is distributed systems which is partly chosen due to the case studies carried out during the PhD, and partly due to the fact that CPN is a suitable language for describing distributed systems. We explain this claim in Sect. 2 below. Viewed as a method for developing systems, CP-nets support the development process on many levels. In the present thesis we limit ourselves to the claim that CP-nets support the two activities of design and analysis (the latter here meaning reasoning about properties). Although a full development method must support many other activities it is, in our opinion, design and analysis which are the more challenging and important. In our case studies it has been the design and analysis activities which have been prevailing.

The important perspective of this thesis is to regard CP-nets as a formal method with emphasis on pragmatics. We use the term of pragmatics in the sense of something being practicable, with emphasis on usefulness and suitability. In fact, the perspective of pragmatism is central to this thesis — a provocative hypothesis, if one wishes. The title may also be interpreted as suggesting that no other formal method promotes pragmatism, which obviously would be likely to be unfair. However, with the title we wish to indicate that CP-nets are potentially more focused on pragmatism than most other commonly known formal methods. We want to emphasise, throughout this thesis, that CP-nets indeed is a practically applicable visual language, a formal method, suitable for describing and reasoning about concurrent systems.

In the following sections of this introduction we explain the central concepts which constitute the title, and describe the structure of the remaining thesis.

1.1 Coloured Petri Nets

CP-nets was in 1979 formulated by Kurt Jensen [53, 54, 55, 56], and is a graphical oriented general purpose language useful for specifying, designing, and analysing concurrent systems. CP-nets are derived from Genrich and

Lautenbach's Predicate/Transition Nets [35, 34] which are a generalisation of Condition/Event Nets and Place/Transition nets. Condition/Event Nets were founded by Carl Adam Petri with his PhD thesis "Kommunikation mit Automaten" in 1962 [84]. With Petri's thesis it was the first time a general theory for discrete concurrent systems was formulated. The formal model of Petri Nets is a generalisation of automata theory such that the concept of concurrently occurring events can be expressed in a simple but powerful framework. The semantics of Petri Nets is mathematically defined, and Petri Nets have the advantage of being executable. For general access to information on Petri Nets and the community researching in Petri Nets, see the Petri Nets home-page on the World-Wide Web [117] (currently maintained and moderated by the author under the auspices of the Petri Nets steering committee). Additionally, there exists a number of introductory articles and books on Petri Nets [79, 92].

The syntactical elements of CP-nets are essentially places, transitions, arcs, and inscriptions. Only the latter is textual while the rest are graphical elements. See Fig. 1 for an overview of these elements. Places are drawn as ellipses and

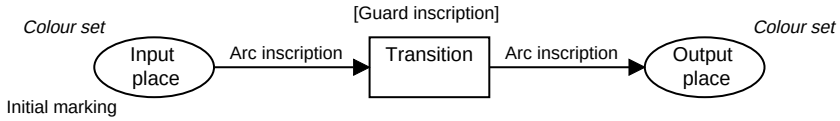

Figure 1: The elements of Coloured Petri Nets.

represent a kind of containers which may contain multi-sets of elements (tokens) of some specified type (colour set). The collection of tokens on the places in a CPN model determine, at any time, the state of the given system. Transitions are drawn as rectangles and represent possible actions. The arcs represent the relation between places and transitions, and determine how a state changes when an action occurs. The arc inscriptions, which are functions, are used to determine the quantity of tokens moved between states. The role of guards is to filter and restrict possible action occurrences. Possible actions not in conflict may occur concurrently, and occurrences happen instantaneously, i.e., as atomic actions. Colour sets are types on places, and initial markings determine the initial configuration/state of a CP-net. Finally, CP-nets include support for large-scale modelling by means of hierarchical decomposition (of transitions). As can be seen, the elements of CP-nets are few and the concepts simple, and it is therefore not surprising that the basics of CP-nets can be taught in a matter of hours. Using and applying CP-nets in projects is a different matter. Experience shows that 1–2 weeks of training is necessary before users can apply CP-nets in non-trivial projects. Because of this, CP-nets have become widely known and have been applied in many domains. Typical application areas of CP-nets include, but are not limited to, protocols, VLSI, embedded, and distributed systems. The CPN research group at University of Aarhus has a World-Wide Web page with information on current activities in the field of CP-nets [114].

The popularity of CP-nets is also due to serious and industrial minded tool support. Design/CPN [57] is a tool which supports CP-nets, and is now maintained and developed by the CPN research group at University of Aarhus. Formerly, Design/CPN was developed chiefly in an industrial setting at Meta

Software Corporation, Massachusetts in close cooperation with researchers at University of Aarhus. A lot of information about Design/CPN exists on the World-Wide Web [115].

1.2 Formal Methods

In essence, *formal methods* denote the application of mathematical approaches to software/hardware development activities such as specification, design, validation, verification, and implementation. Thus formal methods, being characterised by mathematical techniques, provide a rigorous foundation to systems development. Based on mathematics, we can expect that a formal method supports an unambiguous and abstract approach of talking and reasoning about systems. In spite that prejudiced opinions consider formal methods rarely applicable, they are just beginning to become more wide-spread in industry. This change has happened especially during the past few years. Formal methods are typically introduced in the development process where critical conditions arise, such as safety- or life-critical systems, and where systematic testing is difficult. Typical examples of formal methods constitute VDM, the Z notation, RAISE, and LOTOS. There are many general introductions to formal methods some of them being [113, 43, 7], and with an industrial perspective in [8]. On the World-Wide Web there also exists important resources on formal methods [6, 116].

1.3 Design and Analysis

In this work we use a somewhat broad meaning of the concept of design. We consider *design* to cover activities which, by some techniques and methods, have the purpose of making a representation of a system in question, typically by means of a kind of computer based language. The system in question, also called a referent or target system, may be concrete or just an idea. A design captures some aspects of the referent system while abstracting away others, and often has a specific viewpoint in mind, for instance analysis of some in advance known or required properties. A design can be used for several purposes. Some of the more commonly encountered are implementation and analysis. Implementation is a result of using the design as a recipe.

In this work we consider *analysis* to be the effort of obtaining answers to questions about the behaviour of a design or system. We consider *validation* to be the sub-activity of trying to convince others (and yourself) of sound behaviour of the system, and we consider *verification* to be the sub-activity of proving rigorously that a formal design has a formally stated property. Verification has two prevailing approaches, namely model checking and theorem proving. *Model checking* involves exhaustively checking that a property holds in a finite representation of the design, such as a state space. Although a system may have infinitely many states, reduction techniques may be used to create a finite representation. *Theorem proving* involves automatic or semi-automatic proof search. Analysis has the goal of gaining insight and confidence in aspects of behaviour of the design.

1.4 Distributed Systems

The application domain, which is referred to in the title of the thesis, is distributed systems. It turns out that CP-nets are well-suited for describing essential characteristics found in typical distributed systems. What characterises distributed systems is described in the following.

There exist many different instances of defining distributed systems, which depend mostly on the point of view of research. For instance Tanenbaum [101], who works with distributed operating systems, takes the users' point of view:

> "A distributed system is a collection of independent computers that appear to the users of the system as a single computer."

Interestingly in an earlier definition [100] it is additionally required that the computers do not have shared memory.

It is typical that people working with distributed algorithms focus more on correctness. For instance Lynch [68] uses the following definition which is significantly more specific in terms than that of Tanenbaum:

> "Distributed algorithms are algorithms designed to run on hardware consisting of many interconnected processors. Pieces of a distributed algorithm run concurrently and independently, each with only a limited amount of information. The algorithms are supposed to work correctly, even if the individual processors and communication channels operate at different speeds and even if some of the components fail."

There are other books on distributed algorithms such as Tel [102]. More general introductions to distributed systems exist [28, 24, 98]. One of the most general definitions we have seen has been proposed by Garg [33]:

> "Concurrent systems that consist of multiple computers connected by a communication network are called *distributed systems*."

We do not rely on a specific definition. Whether to require shared memory or to demand no single point of failure is not of our concern in this thesis. However, the point is to get a general impression on distributed systems by the above definitions, and it is important to note that the general aspects of distributed systems typically include concurrency and communication between processes (implying protocols).

1.5 Structure of the Overview

The remainder of this paper, the thesis overview, is structured as follows: Below we proceed with topical issues in Sect. 2, where we put our work in a general perspective and discuss current issues related to CP-nets. Related work in general, i.e., in the context of the thesis title, is provided in Sect. 3. Then Sect. 4 summarises the main contributions of the individual papers constituting the thesis, and finally concluding remarks are provided in Sect. 5.

2 Topical Issues

In this section we put our work in a general perspective and discuss the current complex of problems in the context of the main theme of this thesis, i.e., CP-nets viewed as a pragmatic formal method.

CP-nets can be considered as a formal method because CP-nets have a formal definition and therefore have roots in mathematics. The CPN method has adapted a number of techniques for reasoning about systems, such as model checking in the instance of state spaces and theorem proving in the instance of linear invariants. As a description method, CP-nets offer mainly the simple elements and concepts as described in the previous introduction section. For an overview of CP-nets applied as a formal method see, e.g., Jensen's books [54, 55, 56], in particular volume 3.

The success criteria of a formal method in academics differ from those in industry. CP-nets are already widely accepted within the Petri Nets research community, while CP-nets are only beginning to be more seriously recognised in industry. (However, CP-nets are one of the most applied kinds of Petri Nets among practitioners.) In spite of many successful case studies with CP-nets in industrial settings, we are still faced with prejudiced opinions towards formal methods as such [43, 7]. General awareness education on formal methods [26] and guidance for choosing the appropriate formal method are topical issues [8]. Another important factor for more permanent success of a formal method in industry is standardisation. SDL [96], wide-spread in the telecommunications industry, is a recommended telecommunication standard. Exactly with this in mind, some people in the Petri Nets community are currently working on a standard for high-level Petri Nets. For more information see the Petri Nets World-Wide Web [117].

Does the CPN method really keep its promises regarding the claim of pragmatism? Again by referring to Jensen's book [56] there is strong evidence that CP-nets are sufficiently applicable for many different purposes. However, some difficulties have been identified based on the experience gained from case studies. As a result there are a number of suggestions for language extensions such as object-oriented Petri Nets [63, 2, 3] and specific constructions [64, 16]. Two of the papers in this thesis also propose extensions, namely a language extension in [20] and an analysis extension in [14]. The pragmatism of CP-nets is especially interesting to study in the context of teaching. Are students in general able to grasp the basic concepts and apply the CPN method? The paper in [21] discusses educational issues in a specific undergraduate course.

It has been discussed, for some time now, why, and if, graphical languages (also referred to as visual or diagrammatic languages) are useful or better than sentential languages such as purely textual languages. Due to the lack of precise definitions of the visual nature of languages, the research in this area is largely based on empirical studies, unfortunately sometimes disappearing in noise due to poor argumentation. Some of the more serious work on assessing graphical languages indicate that graphics and diagrams may not be as accessible as textual representation as currently considered [83]. Even empirical research on comparing Petri Nets and textual representations has been accomplished putting Petri Nets in a new perspective [76, 39]. The results may be different for the case of CP-nets due to the possibility of making more succinct representations. Usually the general conclusions are, that although the accessibility

of graphical representations are criticised, users still see more possibilities in graphics, typically due to more flexibility with respect to perceptual cues [112]. In this thesis we have a few comments on the usability of CP-nets as a graphical language in relation with teaching in [21].

Are CP-nets, and formal methods in general, applicable everywhere or should one choose projects with more care? The world is not a formal system. Therefore we should not expect CP-nets to be able to describe every aspect of the world. In fact, no formal method claims to be a panacea, but is usually specialised to effectively treat its own niche. For instance, SDL [96] is specialised for the telecommunications industry. However, CP-nets are not as domain specific as SDL. The target systems for CP-nets are usually simply expected to be concurrent discrete event systems. Thus we can expect, e.g., that chaotic, highly dynamic, fuzzy systems, such as human social processes, are inappropriate target domains for CP-nets. Success in industry relies on gradual introduction of formal methods as a supplement to existing practices as a tool to handle critical aspects of systems, while emphasising the pragmatics of formal methods.

The target domain we treat in this thesis is distributed systems. In the introduction (Sect. 1) we identified the characteristics of distributed systems, namely that they typically include concurrency and communication between processes. CP-nets are generally considered to be appropriate for describing concurrency and protocols which are inevitable in computer communication. Reisig and his research group are currently very active in the design and analysis of distributed algorithms by means of (high-level) Petri Nets [111]. This group is mainly focused on theorem proving techniques, such as linear invariants and partial order methods, in the verification of Petri Nets designs.

We have now provided sufficient introduction of concepts and a general overview of topical issues in order to proceed with related work below, and in particular the summary of the five papers constituting the body of this thesis.

3 Related Work in General

In this section we emphasise other research activities which are related to the theme of the present thesis, namely design and analysis of distributed systems with formal methods. For a complete overview of formal methods, see the World-Wide Web page assembled by Bowen [6].

There are many formal methods for various purposes, some even developed and used in industry. The most widely known formal methods are VDM (Vienna Development Method) [59], the Z notation [99], RAISE (Rigorous Approach to Industrial Software Engineering) [80], LOTOS (Language of Temporal Ordering Specifications) [108], and Larch [41]. VDM and Z are both based on sequential languages while LOTOS and RAISE include concurrency, and otherwise combine a number of methods. For instance, RAISE attempt to combine, among other methods, VDM and CSP. There are also temporal logic formal methods such as TL [70] and TLA [66] (thus putting into perspective our temporal logic paper in [14]). None of these are based on graphical languages. We comment on two graphical languages below, namely SDL and Statecharts.

SDL (Specification and Description Language) [96] is a general purpose concurrent language for describing communication systems. However it is most commonly know in relation to the telecommunications industry. The basic be-

havioural model is that of communicating processes each represented by an Extended State Machine. Communication is signal oriented, and is handled by means of message queues. SDL share some concepts with CP-nets such as states and transitions, which are graphically represented, but SDL lacks hierarchy structure. An advantage of SDL is that it has a standardised static and dynamic semantics recommended within the auspices of the International Telecommunication Union (ITU). While CP-nets have rather few language constructs, the language of SDL is very rich and complex. As a result, SDL does not share the popular verification methods of CP-nets such as linear invariants. However, many attempts of translating SDL into different kinds of Petri Nets has been accomplished, as described in Sect. 4.1 (under related work). Other telecommunications formal methods are Estelle and LOTOS which both are international standards within ISO.

Statecharts [44] by Harel is in the family of synchronous graphical concurrent languages. The processes of Statecharts are finite state machines which can broadcast and receive messages to and from the surrounding environment. As CP-nets, Statecharts incorporates hierarchy, but this is on the level of processes. CP-nets are not inherently synchronous and have much simpler semantics than Statecharts. In fact, Harel [44] speaks highly about Petri Nets and recommends that hierarchies should be introduced into Petri Nets. In other work by Harel [45] he opts for visual formalisms. They must be formal because diagrams are manipulated and analysed by computers, and must be visual because diagrams are created by and communicated among humans. Harel indicates that diagrams are a very suitable medium for the visual perception system of humans. There are other synchronous languages very similar to Statecharts. Argos [71] is inspired by Statecharts and they share many features: both are graphical and synchronous languages. Related synchronous languages are Esterel, Lustre, and Signal although not inherently graphical as Statecharts.

There are quite a lot of people doing research on the design and analysis of distributed systems, but only few where Petri Nets are a central ingredient. In Sect. 2 we referred to the work of Reisig et al. [111], where they concentrate on partial order techniques and fairness properties of distributed algorithms. Partial order techniques are used with the purpose of causal reasoning. A different aspect of verification can be found in a recent book by Shatz on distributed software, where Petri Nets are used to reason about Ada tasking programs [98]. Although referring to high-level Petri Nets, he never seriously considers this family of nets.

There are many other domains where Petri Nets have been applied. Some recent books have been published about other specific domains where Petri Nets are the main tool. Silva et al. have written a book on the manufacturing domain [27], and Balbo et al. on performance analysis with stochastic Petri Nets [72]. In general, for a comprehensive overview of the activities in the Petri Nets community there is plenty information on the World-Wide Web [117].

4 Contributions

In this section an overview of the work done and results achieved during the course of my PhD research efforts is provided. The overview is based on the original papers, which are published separately. Apart from the present paper,

five other individual papers constitute the PhD thesis. The five papers are grouped as follows:

Case Studies

- **Title:** "Modelling the Work Flow of a Nuclear Waste Management Program"
  Publication information: Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain, June 1994 [77].
  Thesis part: DAIMI PB – 518 ([78])
- **Title:** "Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets"
  Publication information: Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Osaka, Japan, June 1996 [60].
  Thesis part: DAIMI PB – 513 ([61])

Education

- **Title:** "Teaching Coloured Petri Nets — a Gentle Introduction to Formal Methods in a Distributed Systems Course"
  Publication information: To appear in the conference proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse, France, June 1997 [19].
  Thesis part: DAIMI PB – 520 ([21])

Extensions

- **Title:** "Parametrisation of Coloured Petri Nets"
  Publication information: Technical report, University of Aarhus, 1997 [20].
  Thesis part: DAIMI PB – 521 ([20])
- **Title:** "Model Checking Coloured Petri Nets Exploiting Strongly Connected Components"
  Publication information: International Workshop on Discrete Event Systems, Edinburgh, Scotland, UK, August 1996 [13].
  Thesis part: DAIMI PB – 519 ([14])

The latter two extension papers are respectively a language extension and extension of analysis methods.

4.1 Work-Flow Modelling [77]

This section treats the paper "Modelling the Work Flow of a Nuclear Waste Management Program" which is written in cooperation with V. Pinci, Meta Software Corporation, Mass., USA. As part of this thesis, the full paper can be found in [78].

4.1.1 Summary of Paper

This paper is about a project I was involved in while working at Meta Software Corporation (Meta), Boston, USA in 1991. The project is concerned with the disposal of nuclear waste in USA. Nuclear power plants produce, as a by-product, nuclear waste. This kind of waste is one of the most lethal poisons produced by mankind and needs to be isolated from human beings for 10,000 years. Currently there are only a few permanent storage places for nuclear waste, and waste is therefore kept at the plants themselves. As local storage place is limited there is an increasing interest and urge to find more stable and permanent storage sites for nuclear waste. A problem description similar to this periodically shows up in the news media.

The project at Meta had the purpose of studying and improving a proposed work-flow model of a nuclear waste management system which had the task of permanent disposal of nuclear waste in a geological repository. The system is essentially a large project organisation. The original model was provided to us in the form of a large number of IDEF0 diagrams.[1] Our task was then to translate the IDEF0 model into a CPN model. The motivation for doing the translation was due to the fact that IDEF0 models cannot be executed, while this is the case for CPN models. The IDEF0 model was rather complex — it consisted of 116 diagram pages, which represented the highly distributed nuclear waste management system. Therefore the engineers of the IDEF0 model were interested in investigating the dynamic behaviour of their work-flow model by means of a formal method, in this case CP-nets. This is interesting because IDEF0 and CP-nets have a number of striking similarities, such as the graphic elements and hierarchy structure.

4.1.2 Results Achieved and Assessment of Methods Applied

The IDEF0 standard describes a diagram syntax and informal semantics. However, the semantics is not based on a formal model but rather on the interpretation of diagram symbols. The standard explicitly notes that the diagram itself is not sufficient in all cases; the standard gives examples of commonly occurring diagram constructions which have ambiguous interpretations. Therefore, the standard recommends additional explanation in the form of textual descriptions written in the human language. It is thus clear that there does not exist a general algorithm for executing (simulating) IDEF0 diagrams.

The task of Meta, who had the CPN modelling expertise, was to translate the IDEF0 model of the nuclear waste management system into a CPN model with the goal of simulation. For this purpose we used an *ad hoc* method for translation. The method is semi-automatic where the IDEF0 model first is automatically translated into a CPN model without inscriptions, which then were completed manually with inscriptions by us. The inscriptions would, of course, depend on the elaborate descriptions by the IDEF0 modellers.

How would we present the simulation results to the IDEF0 modellers? Again an *ad hoc* method was developed which basically consisted of time-event dia-

[1] IDEF0 (Integration DEFinition language 0) is a standard in USA (National Institute of Standards and Technology in USA), and is widely spread in governments and companies. IDEF0 is based on SADT (Structured Analysis and Design Technique), which is the more common notion in Europe.

grams; a graphical representation where time is mapped on one axis and IDEF0 function activities on the other.

The idea of translating IDEF0 models into CPN models with the purpose of execution and animation is a classical example of applying a formal method to an informal domain. The IDEF0 modellers wanted to investigate the dynamic behaviour of the work-flow in the nuclear waste management system. Available technology did not allow simulation of IDEF0 models directly, thus they determined to use other means. They discovered that the CPN language in appearance is very similar to IDEF0 and, on top of that, CPN has a formal semantics and tool support for simulation and animation. Therefore IDEF0 and CPN appeared to be a promising combination. Previous case studies had already made a proof of concept for the viability of the method, see [85] and [97]. Indeed the method proved to be useful in this project also.

There are also critics of combining IDEF0 with CPN — in fact with any formal method. Part of the IDEF0 community expresses the opinion that IDEF0 being open for multiple interpretations is exactly its strength. By using formal methods to resolve the ambiguities and thus enforcing a specific interpretation may suppress important aspects of an IDEF0 model. Indeed this is an understandable argument.

During the project we experienced that animation of the nuclear waste management system was important for the communication of behaviour with the IDEF0 engineers. Finding the right way to express the behaviour of the system in a graphical fashion was essential, and once an agreeable graphical notation was established the IDEF0 engineers would be able to react on whether or not the behaviour appeared as expected. Once simulating the work-flow the IDEF0 engineers never needed to look at the CPN model. Thus they did not need to learn using a new modelling language.

A weakness of using IDEF0 and CPN as a method is related to the inflexibility of keeping consistency in relation with the semi-automatic translation which was described above. Naturally the simulation and animation revealed many errors. Some were due to bugs in the CPN model itself, and they were easy to fix, thus removed immediately. Yet others were due to design errors and thus required changes to the IDEF0 model. Unfortunately, if a change was made in the IDEF0 model then we would have to redo the semi-automatic translation step again for those parts changed — a fairly laborious task even for local changes. However, most of the moderate changes needed only attention in the enclosed textual descriptions of the IDEF0 model — only few demanded changes in the graphics which triggered the need for redoing the semi-automatic translation.

In general there is a mismatch problem in case there is semi-automatic translation between two tools that are not integrated. In our case we needed an alternative which could be an enrichment of the IDEF0 tool. One possibility could be to add CPN elements in the IDEF0 tool, thus making all manual work there and then making a fully automatic translation to the CPN tool. On the other hand this would likely require some CPN skills of the IDEF0 modellers, which in this case would be a disadvantage.

Consisting of 116 pages the model of the nuclear waste management system was one of the largest seen in relation with CP-nets (actually any kind of Petri Nets). As a result we had an opportunity for exercising currently available tools and modelling techniques on a large scale case study. Due to the size of the model we were forced to split the model into a number of modules, and before

simulation the code was linked together. This was done manually. Another technique we desired was simple parametrisation of the model with values in connection with configuration and initialisation before simulation. Again this was done manually by building a special purpose CPN sub-model for loading configuration files used for the initialisation. Techniques and tools for module library handling and parametrisation would have made our work more efficient and less error prone.

4.1.3 Related Work

During our work with the manual translation phase to an executable CPN model, i.e., the work on making inscriptions, we learned more about the possibility of making this translation fully automatic for the case of work-flow models. Working with a lot of different cases gave us a better understanding in the direction of automation. This was again due to the size of the model. During the past few years Meta has developed tool support for work-flow analysis where restricted classes of IDEF0 models are translated fully automatically either to CP-nets or other executable representations in order to do further analysis or animation. Work-flow analysis has over the years become increasingly popular in industry and government institutions. Many case studies, such as the one presented here, have helped to build knowledge and support in the area of work-flow analysis.

There are other visual languages than IDEF0 which share similarities with CP-nets. One of them is SDL. The first standard was SDL'88 [95] which then later was extended with object-oriented concepts resulting in OSDL also called SDL'92 [96]. Both the graphical notation and the ideas behind the semantics are, in some respects, similar to CP-nets. However, SDL lacks many of the popular analysis methods of CP-nets such as linear invariants. Thus, there has been some research on translating SDL to variants of Petri Nets [25, 31, 23, 62, 49]. In [31] the authors identify a number of extensions to PT-nets (SDL Time nets) needed in order to make a fully automatic translation. One problem is due to infinite message queues in SDL, another is about representation of time and timers in SDL. (The problems disappear if CP-nets are used instead of Petri Nets.) The authors of [31] have the viewpoint that the user never needs to look at Petri Nets. Errors discovered when analysing Petri Nets are presented as errors in the SDL representation. This is a great advantage as this means that the user can apply the analysis methods of Petri Nets without learning new languages and methods. A similar approach is taken in [49] with the tool EMMA [50], except that they use Predicate/Transition Nets [35, 34] and the analysis tool PROD [110].

4.1.4 Future Work

Current work at Meta, related to work-flow and IDEF0, is directed towards their automatic work-flow analysis tool and interfacing with existing commercial analysis and animation packages for work-flow. In general the area of work-flow analysis and business re-engineering activities in industry are currently popular.

4.2 Protocol Modelling and Analysis [60]

This section treats the paper "Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets" which is written in cooperation with J.B. Jørgensen, University of Aarhus. A.V. Sousa contributed to previous stages of this work. As part of this thesis, the full paper can be found in [61].

4.2.1 Summary of Paper

CP-nets have many case studies within modelling and analysis of communication protocols [56, 5, 4]. It was therefore natural to apply CP-nets on a new protocol developed at our department of computer science. The goal was to model the communication protocol used in an object-oriented framework for distributed program execution which is part of the Mjølner BETA system [69]. The framework consists of an implementation in BETA and a number of papers explaining the general design [9]. Prior to this project, there had not been any attempts to analyse the framework with formal methods. Therefore the goal was to build a CPN model of the framework and then validate and verify a number of properties.

We chose to model the protocol from the viewpoint of threads (light-weight processes). This was due to the fact that threads induce the main flow of control in the protocol and that threads are competing for shared resources. It was thus interesting to investigate properties such as liveness and safety in relation with threads. The model was built based on an existing implementation, i.e., in a reverse engineering fashion. The protocol designer acted as a consultant for conceptual issues. It was interesting to observe how the designer would benefit from the application of a formal method such as CP-nets.

4.2.2 Results Achieved and Assessment of Methods Applied

It was from the beginning clear that the protocol was too complicated for the analysis methods and tools if no restrictions and assumptions were made about the protocol. We were not so worried about simulation, which rarely takes up many computational resources. The concern was related to already known experience with using the verification methods of state spaces and linear place invariants (henceforth *invariants* in short) [55]. Both are known to be infeasible in some cases depending of the complexity and nature of the CPN model in question. Therefore we used a number of techniques from the beginning of the modelling process to reduce the complexity of the CPN model. The standard tricks are to make appropriate assumptions about the world, and to limit the complexity of data-types and inscriptions. During the state space analysis stage we applied net structural reduction techniques [42] which reduced the size of the state spaces by a factor of two. Due to lack of tool support we did not apply reduction techniques based on symmetries [55]. (However, today such tool support exists.) The invariant tool [103] we used was very sensible to the complexity of inscriptions, thus this imposed a limiting factor on the nature of inscriptions feasible for the invariant method.

Using the state space method is one of the more intuitive and generally accessible verification methods available. This method does not demand a lot of knowledge on the formal model behind CP-nets, if any. As a result the state space method begins to gain increasing popularity in industry where Petri Nets

and other formal methods are far from general knowledge. As a research topic the state space method, or more generally model checking, is a very active area. The result is an abundance of papers, tools, and experience.

In the protocol project we used state spaces to verify a number of properties which we claimed and required the CPN model of the protocol to have in advance. These properties were formulated in a functional language and verified automatically in the state space tool [17] integrated with Design/CPN. In some cases it is not even necessary to write down the properties in advance. A number of standard Petri Nets properties may be derived (by the tool) without any need for writing state space queries. For instance, dead markings can efficiently be derived, especially when the SCC-graph (strongly connected component graph) is taken into account. This and many other proof rules of Jensen [55] may be used to derive useful properties for a given CPN model. In [14], part of this thesis, we make the query language for state spaces even more user friendly by introducing a temporal logic specially designed to be interpreted on state spaces generated from CP-nets.

Apart from the state explosion problem another major drawback of the state space method is the dependency on the initial marking. Before any state space generation can be accomplished one must specify the starting configuration of the given model. This means, that the properties we verify for a state space are in general only true with respect to the specific initial marking configuration. In spite of this, the state space method is still very useful — it may often be the case that you are only interested in verifying properties for a number of carefully selected initial system configurations.

The invariant method is significantly more demanding of the user in terms of mathematical skills. This is in general an inherent problem in the area of theorem proving methods. Therefore the demand for tool support is even more relevant. In the protocol project we used a prototype tool supporting the exploration for and verification of linear invariants [55]. For the case of PT-nets, invariants can be determined fully automatically. In fact a basis of all possible invariants can be calculated for a given Petri Net model. However, for CP-nets this is not practical in general. In fact, due to the generality of CPN inscriptions, it is for some models an undecidable problem. Although all invariants can be automatically generated they do not necessarily have an intuitive interpretation in the original CPN model. For this reason the approach taken in the Design/CPN invariant tool is based on a semi-automatic algorithm where the user must supply some ideas on what could be an invariant. For practical reasons a set of heuristics are used in order to develop a full invariant based on partial information given by the user. Once an invariant is proposed by the user, the tool verifies the invariant fully automatically.

Although invariants are more difficult to use than state spaces, the invariant method has the great advantage of being independent of the system configuration parameters (initial marking) — in theory. (The currently applied prototype invariant tool uses a verification algorithm which is dependent on the colour sets used. Thus the determined invariants are only valid within the ranges of the specific colour sets.) Properties determined from invariants are general results without relation with specific start configurations of the given CPN model. In fact, the initial marking expression often is part of an invariant property as a symbol. Another advantage is that the checking of invariants are independent of the state space and thus the state explosion problem. Even though an in-

variant property is an equation which must hold for all reachable markings, the check can be made entirely in the structure of the CPN model in question. On the negative side the general checking problem is undecidable, and for many cases exponential, as it essentially deals with determining equality of lambda expressions. These are derived from the inscriptions on arcs (and guards) in the CP-nets [103].

Traditionally within the area of formal methods, and as used in this work, analysis is considered to be the process of obtaining answers to questions about the behaviour of a system or proving correctness. Unfortunately, analysis is sometimes considered to be the process of looking for errors only, i.e., debugging. Although we acknowledge that analysis is useful for debugging we believe that gaining insight into behaviour and properties is just as important. In any case the general purpose of analysis is to increase the confidence in the system in question. In the protocol project, for instance, we used both the state space and invariant methods to verify the same properties to increase our confidence in the methods, and therefore indirectly in the system. There is, however, a (philosophical) problem with this as it is the model we analyse — but does the model appropriately reflect reality? This was also an issue in the protocol project. Although the framework designer believed that our analysis of the CPN model was correct it did not have significant influence on his confidence in his design exactly because of the gap between model and implementation. The designer did actually simplify his protocol design, but that was due to increased insight into the behaviour than the discovery of errors. In fact, no protocol design errors were identified during the analysis process, which in this case would have been of high value for the designer.

4.2.3 Related Work

There is an abundance of literature on the validation and verification of protocols. Below we focus on a few of the approaches which are either topical or somehow exhibit alternative methods for analysis of CP-nets.

Genrich and Shapiro [36] have conducted a rather advanced approach to verification of an arbiter cascade (hardware). It is an instance of combining theorem proving with model checking. The structure of the arbiter cascade is uniquely determined by its depth which is again determined by an integer. Thus for small integers the cascade is small, i.e., the state space is manageable. For small depths they verify the cascade with the state space method, and they then apply mathematical induction in order to verify the arbiter cascade for arbitrary depths. Unfortunately, the class of such regular systems is rather small, and it is therefore seldom that this specific method can be applied. However, for those cases where the state space method can be combined with induction, we have a powerful technique for verifying properties. The combination of the state space method with induction is less likely to be limited by the state explosion problem. (Standard induction over integers can be generalised to well-founded induction. However, so far, we have not seen any instances of verification of Petri Nets properties with well-founded induction.) Another related technique for verifying parametrised systems is compositional state space analysis. Valmari et al. uses this technique to verify the alternating bit protocol where the number of retransmissions is a parameter [107].

A recent industrial protocol analysis project was one accomplished by Chris-

tensen and Jørgensen [18]. They used an intermediate presentation format in their validation effort, namely message sequence charts [51] — a telecommunication standard for specifying and visualising communication patterns in protocols. Thus the CPN designers could quickly use automatically generated message sequence charts to show and discuss behaviour of different scenarios without the need to expose the CPN representation to the engineers. Although teaching the CP-net notation is relatively easy it is still more attractive to use already well-known notations.

In another project by Rasmussen and Singh [91], a similar idea was used in order to communicate behavioural information of a security system modelled with CP-nets. Although the CPN method and tools have been more integrated in the company in question, the authors used an already practised concept, called mimic-boards, in order to make an interactive user interface and behaviour visualisation interface [90]. The graphical mimic-board could both be used to show the current state of the model of the security system, and be used to control the simulation of the underlying CPN model. As with the protocol above [18], the mimic-board was used as customised feedback to abstract away from the CPN representation.

Our main obstacle in the protocol project, as described in [61], was the state explosion problem. We attempted to alleviate the problem by restricting the behaviour by means of property preserving CPN structural reductions. There exists a number of other state space reduction techniques where interesting properties are preserved. Some of these methods are reduction by equivalences [55] and stubborn sets [106].

4.2.4 Future Work

Over the past few years we have seen a growing interest in combining model checking and theorem proving methods. For CP-nets it could be useful to investigate the combination of, e.g., the state space and invariant methods. In general a combination of the state space method with theorem proving may be helpful to alleviate the state explosion problem.

Tools are essential in the effort of maturing analysis methods. Todays potential target systems being analysed are becoming increasingly complex as the computational power of machines increases. Therefore, most interesting analysis methods require tool development. Likewise do the development of tools push technology as the demand for speed and memory increases. Currently the most limiting factor of the state space method is space complexity. Thus we see a tendency towards more research activity around space efficient algorithms. For instance, currently a typical state space generation implementation can calculate many hundreds of nodes and arcs per second resulting in memory resource problems in only 1–2 hours. The most commonly known state space reduction methods are symmetries by Jensen [55], stubborn sets by Valmari [106], and BDDs (binary decision diagrams [12]) applied by McMillan [74]. Although BDDs seem to be a successful state space representation for temporal logic models of hardware systems, it is yet to be seen whether BDDs are efficient and practicable representations for CP-nets.

Our protocol project [61] was also a case study of applying CP-nets as a method — an investigation of the suitability of CP-nets applied to a protocol in an object-oriented design of a framework for distributed executions. How well

did CP-nets support the modelling process in such a scenario? The most obvious obstacle was the problem of modelling an object-oriented design by means of CP-nets, which is not inherently object-oriented. The result was to interpret the object-oriented design from a different viewpoint, i.e., that of CP-nets, instead of doing language simulation of objects and method calls (communication). We determined that, for the protocol project, tool support for substitution places by Huber et al. [48] and channels by Christensen and Hansen [16] would alleviate the representation of objects and method calls respectively. In general it would be of interest how the CPN method could be adapted and extended in order to cope better with representing object-oriented and distributed systems. Object-oriented Petri Nets have been studied in two recent workshops [2, 3].

Distributed systems and Petri Nets in combination have been studied by many. However, we limit ourselves in the following to briefly consider recent work by Reisig et al. [111] on modelling and verification of distributed algorithms. The research investigates how to extend Petri Nets in order to provide better support for modelling distributed algorithms, and develop analysis techniques which are inspired by partial order methods. One of the extensions they consider is related to the issue of modelling fairness. Petri Nets (and thus CP-nets) do not support fairness primitives in the language itself and it is therefore in some cases impractical to model fairness. For instance, in our protocol project we wanted to ensure fairness among threads competing at the entrance of critical regions. We did not find a natural and straight-forward solution ensuring fairness among threads. In fact, the fairness solutions we considered were far away from reality since they would have involved global constraints, such as queues on every place. The work by Reisig et al. contributes a notion of fair transitions. Transitions cannot be enabled infinitely often without occurring (see, e.g., [93]). Clarke et al. [22] takes a different approach where fairness is enforced on the (execution) paths considered in the state space in the context of model checking temporal properties (expressed in CTL). Jensen [55] uses the strongly connected component graph (SCC-graph) in order to determine fairness properties. Thus, given the body of research efforts on fairness issues elsewhere, we suggest to consider similar issues for CP-nets.

In the protocol project we experienced the gap between two different languages namely that of object-orientation in the design and that of CP-nets. Another gap, which we were not exposed to, is the difficulty of going from the CPN model to implementation. One approach is, of course, to make the implementation by hand. This is an error-prone process because a human interpretation is needed. Another possibility is automatic code generation. This approach provides high reliability and saves programming resources. In fact, as a spinoff from the security system project by Rasmussen and Singh [91], it was determined in a pilot project that automatic code generation from CP-nets was feasible. The code was transferred to a chip and subsequently executed. However, performance was not acceptable, therefore current research is focused on efficiency. In fact, recent improvements exhibit encouraging performance results.

4.3 Teaching with Coloured Petri Nets [19]

This section treats the paper "Teaching Coloured Petri Nets — a Gentle Introduction to Formal Methods in a Distributed Systems Course" which is written

in cooperation with S. Christensen, University of Aarhus. As part of this thesis, the full paper can be found in [21].

4.3.1 Summary of Paper

Teaching computer science topics on an advanced level is an interesting and important challenge. It is interesting because the teaching process gives new insight, and it is important because dissemination of knowledge is fundamental to the academic community.

Distributed systems is a topic that has become increasingly prevalent during the past decade, not only in academic circles but also in industry. Job advertisements require knowledge on aspects of distributed systems. Thus teaching computer science students about distributed systems has become of interest and of necessity more than ever. During the past 10–15 years, research on distributed systems has become a very active field in computer science, and as a result there is an abundance of published literature.

At the Computer Science Department (DAIMI), University of Aarhus, we have provided a one-semester course on distributed systems for the past five years (since 1993). As part of the course the students are required to undertake two comprehensive project assignments. In the first assignment the students design and validate three layers in a communication protocol for a distributed system by means of CP-nets. In the second they implement their design in an object-oriented language. Our roles in the course were, among other things, to be responsible for the planning and execution of the two compulsory project assignments.

The course is partly based on the second half of a textbook on operating systems by Tanenbaum [100]. In that book his definition of a distributed system is as follows:

> "A distributed system is one that runs on a collection of machines that do not have shared memory, yet looks to its users like a single computer."

A collection of machines implies a concurrent system, the lack of shared memory suggests communication, and communication implies protocols. CP-nets is a well-suited technique for designing, validating, and verifying concurrent systems and communication protocols. Therefore it was natural to start offering a computer science course with the combination of distributed systems and CP-nets.

The primary goal of the course is to introduce basic concepts and techniques for distributed systems. The concepts and techniques are illustrated by the distributed systems that the students use every day, such as file systems, printers, and electronic mail — concepts they already know from using the workstations at DAIMI. In the course the students are exposed to CP-nets as a practically applicable formal method for designing and analysing distributed systems. This is a supplement to the students' background in formal methods which is mostly on a theoretical level. CP-nets are used to illustrate and introduce concepts encountered in Tanenbaum's book in a gradual fashion.

4.3.2 Results Achieved and Assessment of Methods Applied

Teaching involves a lot of different activities, the more essential being considerations of which didactical methods that should be applied, i.e., considerations such as how to convey the basic concepts efficiently, how to choose appropriate project assignments, how to ensure training in applying and combining concepts in problem solving, etc. In particular we are concerned with the two compulsory project assignments encountered in the course as these are the most difficult and realistic application of CP-nets, as a formal method applied to a distributed system.

As the compulsory project assignments are an important ingredient for putting the major subjects of the course together, CP-nets and distributed systems, we want to ensure some level of quality of the assignments posed. We used the following criteria:

- The project assignment should exercise the concepts from the course textbooks.
- The project assignment should have a meaningful context.
- The project assignment should not be alienating.
- The project assignment should have industrial characteristics.
- The project assignment should be solvable within the given time frame, but should at the same time be open to further work for the ambitious and curious students.

In addition to this we also wish the project assignments to be reusable in the sense that we can use variants of the assignments throughout several semesters with only minor changes. A more detailed explanation of the criteria above can be found in the paper in [21].

The criteria regarding the demand for industrial characteristics in the project assignments should not be underestimated. A majority of the students at DAIMI get jobs in systems engineering. It is of importance that the students get realistic assignments in order to prepare them for such jobs. The students should learn to make an approximate solution to an unsolvable problem. Some possibilities are to make the formulation of the assignments intentionally incomplete and ambiguous, to allow a wide range of very different solutions to be acceptable, to implant unsolvable problems, and to imitate well-known real-world distributed systems. For instance, the classical impossibility result of consensus in distributed algorithms (see, e.g., Lynch [68]) is camouflaged in one of the project assignments. Once the students realise the problem they must determine an approximation.

Throughout the project assignments we observed that the students, in fact, to a large extend successfully apply CP-nets on a larger non-trivial design problem in the project assignments. Using a graphical language such as CP-nets often proves to be a helpful didactic method for introducing new concepts. Our experience is that it is possible to start with CP-nets and teach the main principles in a matter of a few hours, and CP-nets can thus be used very early in an intuitive fashion to illustrate both static and dynamic aspects of systems.

Teaching with CP-nets yield a number of general concepts which can be used to speak about distributed systems. For instance, CP-nets offer general

concepts for concurrency, conflict, boundedness, liveness, deadlock, home properties, among others. The course provides the students with a general framework for understanding the usefulness of applying formal methods to complex concurrent systems. Although CP-nets could be replaced with other formal methods, only few offer the simplicity and pragmatics of CP-nets.

However, the most serious obstacle is how to efficiently apply CP-nets as a technique for designing and analysing systems. As with other expressive computer languages it takes time to combine and apply the elements of CP-nets to structure complicated systems. It takes several weeks to master the CP-net technique on a level where the students are able to apply CP-nets to problems of the size of the project assignments.

4.3.3 Related Work

The successful diffusion of formal methods among computer scientists and engineers depends, perhaps more than expected, on didactics in the context of courses on computer science topics. It seems to be important that the pragmatics of formal methods are taught early and it seems that one should not be afraid of teaching selected topics from mathematics as a prerequisite and support. The recent book "Teaching and Learning Formal Methods" [26] contains a number of papers illustrating the importance of teaching formal methods, both to academics and industry.

Teaching CP-nets as a formal method to industry is only just beginning to get attention. Two recent projects in industry, one with Dalcotech [91] the other with B&O [18], have provided valuable experience with teaching CP-nets in an industrial setting. For these two projects the participants from industry were exposed to six days of CPN training. After that only regular technical meetings were offered when needed, in order to ensure the continuation of the projects in question. It can be discussed whether or not six days are a lot of training. However, it seemed that it was sufficient to get the participants started for those two projects.

Earlier work on teaching of Petri Nets exists in academics. In fact, we share the opinions of Oberquelle [81] and Jantzen [52] that Place Transition nets (PT-nets) are a useful language for explaining concepts on both the informal and the rigorous level. However, the students are quickly faced with the limitations of PT-nets because, even for toy examples, such as the Dining Philosophers, they lack encoding facilities for data. CP-nets, and high-level nets in general, are more applicable than PT-nets for visualising larger systems due to the abstraction facilities such as hierarchies and data types. Our experience is that it is possible to use CP-nets as the starting point in teaching the elements of nets, thus avoiding simpler and less succinct net-formalisms such as PT-nets.

4.3.4 Future Work

One issue is how much can be put into the syllabus of a course. Some topics, such as simulation of CP-nets, depend crucially on tools and the power of computers. As tools mature, the more advanced topics can and should be considered for inclusion. Also, the faster the computers are and the more memory they have, has a strong influence on what can be considered feasible topics. As an example, during the period we have taught the course, the memory has doubled and the

speed has quadrupled of our lab computers. As a result, the project assignments have grown in size and complexity — without compromising feasibility. For the future we consider including into the syllabus aspects of verification methods such as the state space method — in fact elementary use of state spaces is included in the current semester. The state space method can be used in an easy fashion without requiring the user to conduct hard and error-prone manual proofs, and is therefore a potential candidate for an easily accessible introduction to verification. Other verification techniques with CP-nets are introduced in a subsequent advanced course based on material in [55].

As part of the requirements of the project assignments the students must document their work. We are considering to use a hyper-media tool developed by the hyper-media group at DAIMI [40]. Design/CPN already supports communication with the hyper-media tool which means that it is possible to create a hyper-text where, e.g., there are links between CPN elements (such as places and transitions), documentation (text), pictures, etc. In this way the students can be exposed to state-of-the-art documentation techniques.

In Design/CPN a library supporting message sequence charts has been implemented, which is a standardised notation prevalent in the telecommunications industry [51]. Message sequence charts are graphical diagrams illustrating communication patterns over time. It is our expectations that message sequence charts can easily be used by the students in the project assignments. In this way they also become familiar with another notation used in industry.

The improvement of the integration of the two parts of the course, distributed systems and CP-nets, is an ongoing task. But we have gained useful experience so far.

4.4 Parametrisation

This section treats the paper "Parametrisation of Coloured Petri Nets" which is written in cooperation with S. Christensen, University of Aarhus. As part of this thesis, the full paper can be found in [20]. The paper reflects the state-of-the-art of parametrisation of CP-nets.

4.4.1 Summary of Paper

When we wish to make a computer representation of a large family of objects of interest from the world around us, we can either choose to represent all individual objects or try making more efficient representations. The perspective on a given problem has influence on the kind of efficiency needed. For instance, space efficiency is often a concern. Different approaches exist for making efficient representations — the one concerning us is parametrised representations. The fundamental idea is to represent only a part common for all objects in a family and characteristic holes of interest which later can be filled in. For instance, flexible manufacturing cells in a bottling system could consist of an input buffer, a transportation system, and a machine. We can imagine a parametrised representation of a generic flexible manufacturing cell where, e.g., the machine would be a parameter (the hole). Thus if we wish to have a packaging manufacturing cell we just instantiate the generic manufacturing cell by inserting a packaging machine into the hole.

We propose a conceptual framework for parametrisation of CP-nets. It is a first step towards the formulation and formalisation of *Parametric* CP-nets. We identify and characterise three useful kinds of parametrisation, namely value, type, and net structure parameters. While the two former kinds are simple to design the latter kind is more complex, and in this context we describe how net structure parametrisation naturally induces concepts like modules and scope rules. The framework is applied to a non-trivial example from the domain of flexible manufacturing. The example is in particular useful because it has many similar components (manufacturing cells) which conveniently can illustrate the use of the various kinds of parametrisation.

4.4.2 Results Achieved and Assessment of Methods Applied

As a result of studying a number of examples it appears that we can cover many interesting cases with three different kinds of parametrisation, namely value, type, and net-structure parameters. In addition we found it useful to introduce runtime system parameters, which are parameters supplied by the simulation engine at instantiation time.

The simplest parametrisation mechanism is using values such as integers, strings, and other constants used in inscriptions of CP-nets. As we use Standard ML (SML) [104, 75] as inscription language in the Design/CPN tool, it is possible to consider functions as values too. In SML, functions are treated as first-class values. Examples of value parameters in a manufacturing system could be the number of resources available determined by a parametrised initial marking inscription, or time delay value parameters in arc inscriptions.

Parametrisation with types can, in fact, be made just as simple to handle as value parameters. Just as with value parameters, a type parameter can appear in any inscription where the syntactical category requires a type. One of the interesting features of types is the concept of polymorphism. Thus introducing type parameters implies polymorphic CPN models. Although there exists several sub-classes of polymorphism, we only consider those supported by SML and the given ML-compiler, namely, parametric and ad hoc polymorphism. Hence inclusion polymorphism, as found in object-oriented languages, is not considered in the framework.

Net structure parametrisation is relatively more complicated to handle, partly due to the many possibilities to consider and choices which have to be made. We have already made the choice to do parametrisation on the level of modules as suggested above. The structuring mechanism of hierarchies in CP-nets is accomplished via transitions, such that a (substitution) transition can represent another module. Likewise, we introduce net structure parametrisation via transitions such that a transition can refer to any parametrised module which fits a given signature determined by the parametrisation specification. Hence, net structure parametrisation supplements hierarchies, and is considered as a structuring mechanism on an equal footing.

The three kinds of parameters proposed above are user supplied. In the conceptual framework we also propose parameters supplied by the runtime system which, in this case, is the CPN simulation engine. We call this *runtime system parametrisation*. For instance, the simulation time type is a runtime system parameter. Simulating CPN models with time is in the Design/CPN tool currently supported for integers and reals. Thus the time type parameter may be

used to make a CPN model which can handle both types.

Using external CPN model library modules induces the issue of what we should do in the case of name clashes. Furthermore, in general there is the issue of the visibility of names and where the boundaries of visibility must be drawn. Such issues are handled by *scope rules*. Scope rules are therefore a natural consequence of introducing parametrised modules in CP-nets. CP-nets already contain scope rules, but only for the case of place fusion groups. We introduce a generalised notion of scope rules which are inspired by the hierarchical structure of CP-nets. Scope rules are generalised both for fusion groups and name declarations.

4.4.3 Related Work

Chiola et al. [15] define a formal model for Parametric PT-nets. Their formal model is restricted to parametrisation of initial markings, i.e., the parameters are integers. The purpose of their paper is to compare the modelling power of several variants of PT-nets within the framework of Parametric PT-nets. As they formalise Parametric PT-nets we acknowledge that their work is in some sense more rigorous compared with our framework which is informal. However, we cannot compare the results directly as both the purposes and net kinds are different. Chiola et al. identifies that it is in some cases possible to reason about net properties on the level of Parametric PT-nets, i.e., instead of analysing a single system they analyse a family of systems. The family is determined by the parametrised initial markings. One of the more interesting analysis methods they consider is the invariant method.

Another Petri Nets language which supports parametrised representations is the ExSpect framework [109]. This framework is interesting because it is related with the CPN formalism and the framework supports the same three kinds of parametrisation as in this work — in their terminology; functions, types, and processors/subnets. However, the parametrisation concept is not built into the ExSpect formalism, only in their tool. Like our work with parametrisation of CP-nets, the ExSpect framework needs to formalise parametrisation in order to get an unambiguous semantics. However, they already have the advantage of having implemented parametrisation in their ExSpect tool — which we have not. Additionally they have not made any work on analysing parametrised ExSpect representations.

Some object-oriented languages have parametrisation capabilities. One of these is BETA [69]. This language indirectly support parametrisation with a language construct called virtual classes. It is a very general construct which is also used for expressing other mechanisms than parametrisation. In a different paper [82] an interesting idea of type substitution (a kind of genericity) is introduced in object-oriented languages which then works as parametrisation. A parametrised class can in this case be instantiated without the need to supply parameters. The type names are already parameters and are thus already legal types. We do not use this approach in parametrisation of CP-nets, although it is possible in principle. One reason is that we are from the beginning influenced by the target implementation language SML which does not support type substitution (or object-orientation for that matter).

The original standard, called SDL'88, was extended with object-oriented concepts [77] and later the object-oriented version SDL'92 [96] was proposed,

also called OSDL. We take interest in OSDL because the language supports concepts such as virtuals and parameters. The parameters supported are values, types, and processes, the latter being similar to the net structure parameters here. Parametrised SDL modules cannot be executed without supplying parameters, however SDL modules with virtuals can.

4.4.4 Future Work

The conceptual framework on parametrisation of CP-nets is considered to be a preliminary investigation of concepts needed in order to provide support for parameters in CPN models. Some of the next important steps is to apply the current framework on a much larger example, and to make a formalisation of parametrisation in CP-nets. Formalisation is important because a formal model is a fundamental contribution which can be used as a reference. Such a reference is necessary when ambiguities need to be resolved, and can also be very helpful during implementation of a tool — here the integration into Design/CPN. A formal model is helpful when studying parametrisation of analysis methods. Our hope is that a formal model for *Parametric* CP-nets can unify the three kinds of parametrisation we have studied here: value, type, and net structure parameters. Although the tool user does not need to know of this level, a unification may result in a simpler and more general formal model.

Once we have tool support for parametrised CP-nets it is important to evaluate the new mechanisms with larger case studies. This is the natural consequence of how theory and application have mutual influence, and where tools play the important role.

In the conceptual framework we restricted net structure parameters to be on the level of transitions. Naturally we should also consider the case of letting places be parameters. This is analogous to considering both substitution transitions and substitution places as with some of the first formalisations of CP-nets. We expect that net structure parameters on the level of places is very similar to the case of transitions being parameters, and we do not see any serious problems with having both in the same framework. The two concepts are in essence dual, and they are both useful from a modelling point of view.

The parametrisation work presented here does not treat the issue of object-orientation with Petri Nets. There are many other people working with introducing object-oriented concepts into Petri Nets [2, 3]. None of them, however, consider parametrisation in their own variants of Petri Nets. The research on object-oriented Petri Nets is very active, but no common directions or agreement on object-oriented Petri Nets have been made yet.

4.5 Temporal Logics [13]

This section treats the paper "Model Checking Coloured Petri Nets Exploiting Strongly Connected Components" which is written in cooperation with A. Cheng and S. Christensen, University of Aarhus. As part of this thesis, the full paper can be found in [14].

4.5.1 Summary of Paper

State spaces generated from CP-nets are represented as graphs with state information labelled on nodes and transition (occurrence) information labelled on edges. Query languages are usually developed in order to help the user to express and explore properties in terms of the state space. Temporal logics are in particular useful for this purpose and are generally popular and widely used in the theoretical computer science community. In this work we are interested in a logic which both can be model checked efficiently and which is sufficiently expressive for practical purposes. The logic CTL [22] is a popular logic which also has an efficient model checking algorithm. However the logic is not able to express properties about transitions in the state spaces of CP-nets. Although some logics can express properties about transitions, they are normally limited to refer to only one transition label. (See, however, the logic of occurrences by Galton [32] who consider the transition domain only.)

We have therefore developed a variant of CTL which we call ASK-CTL. The logic ASK-CTL is very similar to CTL: it is a branching time temporal logic [87] with a linear time model checking algorithm. From a practical point of view we have extended CTL with operators such that we can express properties about both states and transitions in a dual fashion. Furthermore, we improve the standard model checking algorithms of CTL [22] such that the strongly connected component graph (SCC-graph) is taken into account. The SCC-graph is derived from the state space in linear time and space, where a node in the SCC-graph represents a set of nodes in the state space such that any two state space nodes are mutually reachable. Each SCC-node is maximal, and the SCC-graph represents a unique partitioning of the state space.

4.5.2 Results Achieved and Assessment of Methods Applied

Our work with temporal logics serves two purposes: Firstly, we wish to propose a suitable and practical logic for state spaces generated from CP-nets. Secondly, we are concerned with the efficiency of model checking with the logic in question. The latter is, of course, relevant in connection with building tools. Currently the Design/CPN tool features state spaces in a component called the OG-tool (Occurrence Graph Tool) [17]. With the OG-tool there exists a query language for expressing properties about state spaces. Although the OG-tool already contains a number of predefined queries, such as liveness and identification of dead markings, the query language requires some amount of knowledge of the functional language SML. Temporal logics are in general considered compact and intuitive for expressing tense related properties, and are thus a potential candidate for an alternative (or supplementary) query language for the OG-tool. The logic we propose, ASK-CTL, is a variant of the temporal logic CTL, and we claim that the introduction of ASK-CTL in the OG-tool is an interesting alternative to the existing SML-based query language — from a practical point of view.

CTL is designed to reason about state space structures which do only have labels on the nodes, i.e., CTL is designed only to reason about state properties. State spaces generated from CP-nets also contain labels on edges with information about transitions between states. We therefore extended CTL such that properties about transitions, such as liveness of transitions, are also express-

ible. The difference between ASK-CTL and CTL is essentially one operator which is a domain toggle operator, the two domains being states and transitions. With CTL one speaks about paths of states, i.e., possible traversals of the state space graph where one records the states visited respecting the direction of the edges. In ASK-CTL it makes sense also to speak about paths of transitions. A sub-formula expresses a property either in the domain of states or transition, and only the domain toggle operator can switch between state and transition formulas. For instance, with ASK-CTL it is possible to express something like *Invariantly(Possible(ToggleDomain(StartEating)))* should we wish to specify that a philosopher, in the classical Dining Philosophers example without deadlock, can always reach the situation of deciding to start eating. (Note, however, that the notation is more compact in the original paper in [14].) ASK-CTL seems largely to suit our purposes, since it can express many interesting standard properties of Petri Nets. In [14] we give a number of examples supporting this.

In general when introducing a new logic one needs to consider a lot of issues such as decidability, complexity, axiomatisation, etc. The process of introducing a new logic from scratch is laborious, and the methods used to treat all issues can be very difficult. The approach we use is to extend an existing and well-known logic.

In our paper [14], we outline that ASK-CTL is in fact just as expressive as CTL. Thus we immediately avoid all issues listed above and can concentrate on other issues. The main issues concerning us are developing a practical logic for state spaces generated from CP-nets, and efficient model checking. Inheriting everything from CTL is both an advantage and disadvantage. It is an advantage because CTL comes with a large body of research and experience, and it is a disadvantage because CTL has some weaknesses such as the inability to express fairness properties [22]. (FCTL [30], fair CTL, has been introduced to remedy this shortcoming.)

The complexity of temporal logic model checking, in the case of CTL, is linear in the size of the state space and the depth for the given formula. Although we cannot improve the general worst case linear time complexity we suggest in our paper [14] how to improve the model checking algorithm for interesting specific cases of formula combinations. We use the SCC-graph for this purpose. As an example let us consider the formula *Invariantly(Possible(InterestingMarkings))* which says that no matter where we go in the state space (invariantly) it is always possible to reach a given set of interesting markings. This belongs to the well-known class of Petri Net properties called home properties — the possibility of always getting back to something good. Jensen [55] (Proposition 1.14) has already given proof rules for home properties which provides a connection with the SCC-graph. In fact, proposition 1.14 states that it is sufficient to look only at terminal components (those without outgoing edges) in the SCC-graph. The terminal components, the "Interesting Markings" property must hold for all states (invariantly) belonging to those components. Thus even in the case where the state space induces an SCC-graph with only one component it suffices to check *Invariantly(InterestingMarkings)* and we avoid checking the *Possible*-property. These improvements hold for a number of formula combinations as mentioned in [14].

4.5.3 Related Work

According to Lamport [65] some of the first people to apply temporal logics to program verification were Pnueli [86] and Burstall [94]. There exists an impressive body of research on logics, even if we restrict ourselves to temporal logics. In the following we limit ourselves only to mention related work which is relevant in the context of our research, which is chiefly concerned with branching semantics. (We have not worked with the other commonly known semantics such as linear, partial order, and interval semantics.)

There are a number of temporal logics used together with high-level Petri Nets. One of the more interesting combinations is found in the PROD tool [110]. PROD is a state space analysis tool based on Predicate/Transition Nets [35, 34]. They use two different kinds of temporal logics as query languages, namely the branching time logic CTL and the linear time logic LTL [86]. Heljanko [46] has worked on improving their model checking algorithms, and as part of discussing related work they consider the usage of the SCC-graph, and notice that their model checking algorithms can be used with our SCC-graph technique.

The PEP system (Programming environment based on Petri Nets) [38] is a larger environment for the development and verification of parallel systems. The environment is too extensive to be explained here, however two of the components respectively support M-Nets, a kind of high-level nets, and model checking based on a branching time temporal logic. Apart from the basic propositional logic, the PEP logic contains only the two temporal operators possibility and invariance. Thus it is not as expressive as CTL. However, compared with Design/CPN, PEP seems to be a more integrated and complete environment seen from the viewpoint of developing concurrent programs. But it is not known how well PEP scales for large programs.

SMV (Symbolic Model Verifier) [73, 74] by Clarke's research group, is a model checker which verifies CTL formulas. Although not based on Petri Nets, SMV should be mentioned as it is an important tool which is able to handle very large state spaces represented by BDDs [74]. Another important model checking tool, not based on Petri Nets either, is the LTL model checker, SPIN [47]. The model checking algorithm uses on-the-fly verification [37] when model checking LTL.

There are many people using SCC-graphs in relation with temporal logics. However, only few consider using SCC-graphs to speed up the model checking algorithm. Other people use SCC-graphs for different purposes, such as fairness considerations as in the work with CTL^F, CTL with fairness, by Clarke et al. [22]. Recent work by Pogrell [88] uses SCC-graphs in model checking of a less expressive logic (S_4) applied in relation with Time Petri Nets. The logic does not contain until-operators as CTL (and ASK-CTL), and they do not use SCC-graphs on the same level as in our work.

4.5.4 Future Work

Above we indicated that CTL, and therefore ASK-CTL, is not able to express fairness properties. In spite of the fact that the CTL model checking algorithm has favourable linear time complexity, we should still consider alternative temporal logics. LTL is a possible alternative as fairness properties can be expressed, but LTL does not replace CTL as the expressiveness of the two logics

is mutually incomparable, even if CTL is extended with fairness [30]. CTL* encompasses both the expressiveness of LTL and CTL [29], however CTL* is impractical from the viewpoint of model checking [30]. Therefore we should consider to include LTL in Design/CPN as an alternative possibility to ASK-CTL, because CTL and LTL together can express even more useful properties. Alternatively, we could follow up on the advice given by Emerson and Halpern [29] that one should find an appropriate subset of CTL* for the given application.

ASK-CTL is only at a prototype stage. CTL has been applied in many case studies, therefore we expect that ASK-CTL is also just as practical. But a larger case study will increase our confidence in our implementation of the improved model checking algorithm. Extended support needs to be developed in order to provide a better environment for ASK-CTL. Used as a debugging tool we need to be able to show counterexamples to the user in case the supplied formula is false. In the context of existential path quantifiers the tool could additionally show a witness path. Counterexamples and witness paths both help the user to get insight into behaviour of the model in question.

Critiques of temporal logics claim that these logics are hard to learn and in general hard to read due to their succinctness and language idiosyncrasy. Formulas with depth more than 4 are impossible to comprehend. It is not surprising that temporal logics may be difficult for some to learn as it is typically the case that the concepts presented are totally new, and radically different from existing practices. On the other hand, the philosophical background behind temporal logics is very old and could be an alternative approach to teaching about temporal logics, thus avoiding mathematical based didactic methods. After all, temporal thinking is already inherent in the intuitive comprehension of everyday activities. Take an arbitrary temporal logic formula. It is not difficult to read it aloud in ordinary plain English. The difficulty arises when thinking more abstractly about events ordered in the space of time.

In the 1950'ies and 60'ies Prior [89] established and formalised a logic of time and tenses, of which temporal logics, among other logics, are based. (Prior was inspired by, among other people, ideas of Peirce, and triggered by a footnote in a book by Findlay.) The philosophy of Prior was to find the roots of tense logics in the real-world and not only in symbols of mathematics. Although today, many temporal logics are very succinct, we should expect, in the light of Prior, that temporal logics are practicable languages in specifying and reasoning about real-world systems.

It is, indeed, in general, hard to read deeply nested formulas. However, most interesting formulas are limited to depths 2–3. Regarding specific difficulties with ASK-CTL, we have experienced that it may be a disadvantage to mix sub-formulas from the two domains of states and transitions because there is currently no difference in syntax. We will consider to make the syntax more explicit in order to distinguish which domain a sub-formula belongs to.

5 Concluding Remarks

The leitmotif of this thesis concerns the pragmatics, practical aspects, and intuition of CP-nets from the perspective of being a formal method. In this thesis the pragmatism has been explored in three different areas. Firstly, the applicability of analysis methods have been illustrated and evaluated by means of the

case studies. Secondly, didactical methods have been developed with experience from the educational study. Thirdly, it has been shown that the CPN method can benefit from extensions by means of a language extension and an extension to analysis methods.

Our general experience with CP-nets as a formal method is that CP-nets are sufficient in the process of design and analysis within a number of domains, in particular distributed systems. The potential as a graphical language should not be underestimated. Although it appears that diagrammatic oriented descriptions, in general, are just as hard to comprehend as sentential descriptions (perhaps more than expected), this kind of languages has potential for hiding much of the mathematical notation. Hiding mathematical notation potentially enables a formal method to reach a wider spectrum of people in industry. Furthermore, CP-nets offer additional possibilities as compared with sentential languages with the use of perceptual cues by means of the graphics. Compared with other graphical oriented languages, such as SDL and Statecharts, CPN is generally a much simpler language with clean semantics and only relatively few concepts to grasp — and it offers true concurrency as an inherent property of the language. Being simple, and yet expressive, CP-nets offer the flexibility of adaption and specialisation to many different domains. In fact, there already exists a variety of Petri Net formalisms with special purposes. Although not as widely spread in industry, the forthcoming international standard for high-level Petri Nets may remedy this.

A number of formal methods are already being used in industry, such as VDM, the Z notation, SDL, among many other methods. The future seems to be promising with respect to increasing popularity of formal methods in industry as more complex concurrent systems are being challenged. General visibility must be pursued for instance by means of case study publications such as [11, 1, 67, 56].

References

[1] J.-R. Abrial, E. Börger, and H. Langmaack, editors. *Formal Methods for Industrial Applications; Specifying and Programming the Steam Boiler Control*, volume 1165 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[2] G. Agha and F. de Cindio, editors. *Workshop on Object-oriented Programming and Models of Concurrency of the 16th International Conference on Application and Theory of Petri Nets*, Turin, Italy, 1995.

[3] G. Agha, F. de Cindio, and A. Yonezawa, editors. *Workshop on Object-oriented Programming and Models of Concurrency of the 17th International Conference on Application and Theory of Petri Nets*, Osaka, Japan, 1996.

[4] J. Billington and M. Diaz, editors. *Workshop on Petri Nets and Protocols of the 16th International Conference on Application and Theory of Petri Nets*, Turin, Italy, 1995.

[5] J. Billington, G.R. Wheeler, and M.C. Wilbur-Ham. PROTEAN - A High-level Petri Net Tool for the Specification and Verification of Com-

munication Protocols. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets, Theory and Application*. Springer-Verlag, 1991.

[6] J. Bowen. The World-Wide Web Virtual Library on Formal Methods. World-Wide Web. <URL: http://www.comlab.ox.ac.uk/archive/formal-methods.html>.

[7] J.P. Bowen and M.G. Hinchey. Seven more myths of formal methods. *IEEE Software*, 12(4):34–41, July 1995.

[8] J.P. Bowen and M.G. Hinchey. Ten commandments of formal methods. *IEEE Computer*, 28(4):56–63, April 1995.

[9] 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*, Lecture Notes in Computer Science, Kaiserslautern, Germany, 1993. Springer-Verlag.

[10] W. Brauer, editor. *Net Theory and Applications: Proceedings of the Advanced Course on General Net Theory of Processes and Systems*, volume 84 of *Lecture Notes in Computer Science*, Hamburg, October 1979. Springer-Verlag.

[11] M. Broy, S. Merz, and K. Spies, editors. *Formal Systems Specification; The RPC-Memory Specification Case Study*, volume 1169 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[12] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computing*, C-35(12):677–691, 1986.

[13] A. Cheng, S. Christensen, and K.H. Mortensen. Model Checking Coloured Petri Nets Exploiting Strongly Connected Components. In M.P. Spathopoulos, R. Smedinga, and P. Kozák, editors, *International Workshop on Discrete Event Systems, WODES96*, pages 169–177, Edinburgh, Scotland, UK, August 1996. Computing and Control Division, Institution of Electrical Engineers.

[14] A. Cheng, S. Christensen, and K.H. Mortensen. Model Checking Coloured Petri Nets Exploiting Strongly Connected Components. Technical Report DAIMI PB – 519, University of Aarhus, Computer Science Department, March 1997. ISSN 0105-8517. Also in [13].

[15] G. Chiola, S. Donatelli, and G. Franceschinis. On Parametric P/T nets and their Modelling Power. In *Application and Theory of Petri Nets, 12th International Conference*, pages 206–227. IBM Deutschland, 1991.

[16] S. Christensen and N.D. Hansen. Coloured Petri Nets Extended with Channels for Synchronous. In Valette [105], pages 159–178. LNCS 815.

[17] S. Christensen, K. Jensen, and L. Kristensen. *The Design/CPN Occurrence Graph Tool. User's manual version 3.0.* Computer Science Department, University of Aarhus, 1996. <URL: http://www.daimi.aau.dk/designCPN/>.

[18] S. Christensen and J.B. Jørgensen. Analysis of Bang & Olufsen's BeoLink Audio/Video System Using Coloured Petri Nets. Technical report, Computer Science Department, University of Aarhus, Denmark, 1996. To appear in the conference proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse, France, June 1997.

[19] S. Christensen and K.H. Mortensen. Teaching coloured petri nets — a gentle introduction to formal methods in a distributed systems course. Technical report, Computer Science Department, University of Aarhus, 1996. To appear in the conference proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse, France, June 1997.

[20] S. Christensen and K.H. Mortensen. Parametrisation of Coloured Petri Nets. Technical Report DAIMI PB – 521, University of Aarhus, Computer Science Department, April 1997. ISSN 0105-8517.

[21] S. Christensen and K.H. Mortensen. Teaching Coloured Petri Nets — a Gentle Introduction to Formal Methods in a Distributed Systems Course. Technical Report DAIMI PB – 520, University of Aarhus, Computer Science Department, March 1997. ISSN 0105-8517. Also in [19].

[22] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite State Concurrent System Using Temporal Logic. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[23] G. Comparin and G.A. Lanzarone et al. Analysis of SDL Specifications Using Petri Nets Techniques. In *Workshop of Second SDL-Users' Forum*, Helsinki, 1985.

[24] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design (second edition).* Addison-Wesley Publishing Company, 1994.

[25] J.P. Courtiat, M. Ayache, and B. Algayres. A Petri Net Model of SDL. In *Proc of Fifth European Workshop on Application and Theory of Petri Nets*, pages 272–289, Aarhus, Denmark, 1984.

[26] C.N. Dean and M.G. Hinchey, editors. *Teaching and Learning Formal Methods.* Series in Formal Methods. Academic Press, 1996.

[27] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman and Hall, 1993.

[28] S. Mullender (edited by). *Distributed Systems (second edition).* ACM Press, 1994.

[29] E.A. Emerson and J.Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *ACM*, 33(1):151–178, January 1996.

[30] E.A. Emerson and C.-L. Lei. Temporal reasoning under generalized fairness constraints. In B. Monien and G. Vidal-Naquet, editors, *3rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 21–36, Orsay, France, January 1986. Springer-Verlag.

[31] J. Fischer and E. Dimitrov. Verification of SDL Protocol Specifications using Extended Petri Nets. In Billington and Diaz [4], pages 1–12.

[32] A. Galton, editor. *Temporal Logics and Their Applications.* Academic Press, 1987.

[33] V.K. Garg. *Principles of Distributed Systems.* Kluwer Academic Publishers, 1996.

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

[35] H.J. Genrich and K. Lautenback. 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.

[36] 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*, volume 616 of *Lecture Notes in Computer Science*, Sheffield, UK, 1992. Springer Verlag.

[37] R. Gerth, D. Peled, M.Y. Vardi, , and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *IFIP/WG6.1 Symp. on Protocol Specification, Testing, and Verification*, Warsaw, Poland, June 1995.

[38] B. Grahlmann, S. Römer, T. Thielke, B. Graves, M. Damm, R. Riemann, L. Jenner, S. Melzer, and A. Gronewold. Pep: Programming environment based on petri nets. In E. Best and H. Fleischhack, editors, *Hildesheimer Informatik-Berichte 14/95.* Universität Hildesheim, May 1995. World-Wide Web: http://www.informatik.uni-hildesheim.de/~pep/HomePage.html.

[39] T.R.G. Green and M. Petre. When visual programs are harder to read than textual programs. In *Proceedings of Sixth European Conference on Cognitive Ergonomics*, pages 167–180, 1992.

[40] K. Grønbæk and R.H. Trigg. Design Issues for a Dexter-based Hypermedia System. *Communications of the ACM, Vol. 37, 2*, 1994.

[41] J.V. Guttag, J.J. Horning, S.J. Garland, K.D. Jones, A. Modet, and J.M. Wing. *Larch: Languages and Tools for Formal Specification.* Texts and Monographs in Computer Science series. Springer-Verlag, 1993.

[42] 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.

[43] J.A. Hall. Seven myths of formal methods. *IEEE Software*, 7(5):11–19, September 1990.

[44] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[45] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, May 1988.

[46] K. Heljanko. Implementing a ctl model checker. In *Proceedings of the Workshop Concurrency, Specification & Programming 1996*, pages 75–84, Berlin, September 1996. Informatik-Bericht Nr. 69, Humboldt-University.

[47] G. Holzmann and D. Peled. The state of spin. In *Computer Aided Verification, CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 385–389, New Brunswick, 1996. Springer-Verlag. World-Wide Web: http://netlib.bell-labs.com/netlib/spin/whatispin.html.

[48] 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*, pages 313–341. Springer-Verlag, 1991.

[49] N. Husberg. SDL Modelling with High Level Petri Nets. In *Workshop on Concurrency, Specification and Programming*, Berlin, September 1996.

[50] N. Husberg, L. Ojala, O. Penttinen, and A. Vainonen. Report of a preliminary investigation: Emma — an extendible multi method analyser. Digital Systems Laboratory, Helsinki University of Technology, Finland, August 1996.

[51] International Telecommunication Union; Telecommunication Standardization Sector (ITU-T). ITU-T Recommendation Z.120: Message Sequence Chart, Geneva, Switzerland, 1993.

[52] M. Jantzen. Structured Representation of Knowledge by Petri Nets as an Aid for Teaching and Research. In Brauer [10], pages 507–517.

[53] K. Jensen. Coloured Petri Nets and the Invariant Method. *Theoretical Computer Science*, 14:317–336, 1981.

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

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

[56] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Volume 3, Practical Use.* Monographs in Theoretical Computer Science. Springer-Verlag, 1997.

[57] K. Jensen, S. Christensen, P. Huber, and M. Holla. *Design/CPN. A reference manual.* Computer Science Department, University of Aarhus, Denmark, 1996. Available from [115].

[58] K. Jensen and G. Rozenberg, editors. *High Level Petri Nets.* Springer-Verlag, 1991.

[59] C.B. Jones. *Systematic Software Development using VDM.* Prentice Hall International, 1990. 2nd edition.

[60] 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, *Application and Theory of Petri Nets 1996, 17th International Conference*, volume 1091 of *Lecture Notes in Computer Science*, pages 249–268, Osaka, Japan, June 1996. Springer-Verlag.

[61] J.B. Jørgensen and K.H. Mortensen. Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets. Technical Report DAIMI PB – 513, University of Aarhus, Computer Science Department, February 1997. ISSN 0105-8517. Also in [60].

[62] E. Kettunen, E. Montonen, and T. Tuuliniemi. An Interactive PrT-Net Tool for Verification of SDL-Specifications. Technical Report 3, Helsinki University of Technology, Digital System Laboratory, Otaniemi, 1988.

[63] C. Lakos. From Coloured Petri Nets to Object Petri Nets. In G. De Michelis and M. Diaz, editors, *Proceedings of the 16th International Conference on Application and Theory of Petri Nets*, pages 278–296, Turin, Italy, June 1995. Springer-Verlag. LNCS 935.

[64] C. Lakos and S. Christensen. A General Systematic Approach to Arc Extensions for Coloured Petri Nets. In Valette [105], pages 338–357. LNCS 815.

[65] L. Lamport. What good is temporal logic? In R.E.A. Mason, editor, *Proceedings of the IFIP Congress on Information Processing 83*, pages 657–668, Amsterdam, 1983. Elsevier Science Publishers, North-Holland.

[66] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[67] C. Lewerentz and T. Lindner, editors. *Formal Development of Reactive Systems; Case Study Production Cell*, volume 891 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[68] N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., 1996.

[69] O. L. Madsen, B. Møller-Pedersen, and K. Nygaard. *Object-Oriented Programming in the BETA Programming Language.* Addison Wesley, 1993. <URL: http://www.mjolner.com/>.

[70] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.

[71] F. Maraninchi. The Argos language: Graphical representation of automata and description of reactive systems. In *IEEE Workshop on Visual Languages*, October 1991.

[72] M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Wiley Series in Parallel Computing. John Wiley and Sons, 1994.

[73] K.L. McMillan. *The SMV System.* Carnegie-Mellon University, February 1992. World-Wide Web: http://www.cs.cmu.edu/~modelcheck/smv.html.

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

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

[76] T.G. Moher, D.C. Mak, B. Blumenthal, and L.M. Leventhal. Comparing the comprehensibility of textual and graphical programs: The case of petri nets. In *Proceedings of Empirical Studies of Programmers: Fifth Workshop*, pages 137–161, 1993.

[77] K.H. Mortensen and V. Pinci. Modelling the Work Flow of a Nuclear Waste Management Program. In Valette [105], pages 376–395. LNCS 815.

[78] K.H. Mortensen and V. Pinci. Modelling the Work Flow of a Nuclear Waste Management Program. Technical Report DAIMI PB – 518, University of Aarhus, Computer Science Department, March 1997. ISSN 0105-8517. Also in [77].

[79] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[80] M. Nielsen, K. Havelund, K.R. Wagner, and C. George. The raise language, method and tools. *Formal Aspects of Computing*, 1:85–114, 1989.

[81] H. Oberquelle. In Teaching and in Terminology Work. In Brauer [10], pages 481–506.

[82] J. Palsberg and M.I. Schwartzbach. *Object-Oriented Type Systems.* John Wiley and Sons, 1994.

[83] M. Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6):33–44, 1995.

[84] C.A. Petri. *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Also in English translation: New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, Suppl. 1, 1966.

[85] V.O. Pinci and R.M. Shapiro. An Integrated Software Development Methodology Based on Hierarchical Coloured Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1991*, volume 524 of *Lecture Notes in Computer Science*, pages 227–252. Springer-Verlag, 1991. Also in [58] pages 649–666.

[86] A. Pnueli. The temporal logic of programs. In *18th IEEE Annual Symposium on the Foundations of Computer Science*, pages 46–57, Providence, R.I., November 1977. ACM.

[87] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends. In G. Goos and J. Hartmanis, editors, *Current Trends in Concurrency*, number 224 in Lecture Notes in Computer Science, pages 510–584. Springer-Verlag, 1996.

[88] L. Pogrell. A new model checking algorithm for time nets. Technical report, Humboldt University Berlin, April 1996.

[89] A.N. Prior. *Time and Modality.* Oxford, 1957.

[90] J.L. Rasmussen and M. Singh. Mimic/CPN, A graphical Animation Utility for Design/CPN (vers. 1.5). Technical report, Computer Science Department, University of Aarhus, 1995.

[91] J.L. Rasmussen and M. Singh. Designing a Security System by means of Coloured Petri Nets. In J. Billington and W. Reisig, editors, *Proc. of the 17th International Conference on Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 400–419, Osaka, Japan, June 1996. Springer-Verlag.

[92] W. Reisig. *Petri Nets, An Introduction.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[93] W. Reisig. Petri net models of distributed algorithms. In J. van Leeuven, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 441–454. Springer-Verlag, 1995.

[94] R.M. Burstall. Program proving considered as hand simulation plus induction. In *Proceedings of IFIP Congress on Information Processing*, pages 308–312. North-Holland Pub. Co., 1974.

[95] CCITT, Specification and Description Language SDL, Recommendation Z100–Z104, ITU, 1988.

[96] CCITT, Specification and Description Language SDL, Recommendation Z100–Z104, ITU, 1992.

[97] R.M. Shapiro, V.O. Pinci, and R. Mameli. Modelling a NORAD Command Post Using SADT and Coloured Petri Nets. In P.E. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning*, volume 693 of *Lecture Notes in Computer Science*, pages 84–107. Springer-Verlag, 1993.

[98] S.M. Shatz. *Development of Distributed Software: Concepts and Tools.* Macmillan Publishing Company, 1993.

[99] J.M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics.* Cambridge University Press, 1988.

[100] A.S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall International, 1992.

[101] A.S. Tanenbaum. *Distributed Operating Systems.* Prentice-Hall International Editions, 1995.

[102] G. Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, 1994.

[103] 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.

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

[105] R. Valette, editor. *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, Zaragoza, Spain, June 1994. Springer-Verlag. LNCS 815.

[106] A. Valmari. Stubborn Sets for Reduced State Space Generation. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, pages 491–515. Springer-Verlag (*LNCS* 483), 1990.

[107] 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.

[108] P.H.J. van Eijk, C.A. Vissers, and M. Diaz, editors. *The formal description technique LOTOS.* Elsevier Science Publishers B.V., 1989.

[109] K.M. van Hee, L.J. Somers, and M. Voorhoeve. Executable Specifications for Distributed Information Systems. In E.D. Falkenberg and P. Lindgreen, editors, *Proceedings of the IFIP TC 8 / WG 8.1 Working Conference on Information System Concepts: An In-depth Analysis*, pages 139–156, Namur, Belgium, 1989. Elsevier Science Publishers, Amsterdam.

[110] K. Varpaanniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo. PROD Reference Manual. Technical report, Digital Systems Laboratory, Helsinki University of Technology, Finland, August 1992. World-Wide Web: http://topos.hut.fi/~petrinet/prod.html.

[111] R. Walter, H. Völzer, T. Vesper, W. Reisig, E. Kindler, J. Freiheit, and J. Desel. Memorandum: Petrinetzmodelle zur Verifikation Verteilter Algorithmen. Technical Report 67, Informatik-Berichte der Humboldt-Universitt zu Berlin, August 1996.

[112] K.N. Whitley. Visual programming languages and the empirical evidence for and against. To appear in Journal of Visual Languages and Computing.

[113] J.M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, September 1990.

[114] Coloured Petri Nets. University of Aarhus, Computer Science Department, World-Wide Web. <URL: http://www.daimi.aau.dk/CPnets/>.

[115] Design/CPN Online. University of Aarhus, Computer Science Department, World-Wide Web. <URL: http://www.daimi.aau.dk/design-CPN/>.

[116] Formal Methods Europe, a European organization supported by the Commission of the European Union. World-Wide Web. <URL: http://www.cs.tcd.ie/FME/>.

[117] The World of Petri Nets. University of Aarhus, Computer Science Department, World-Wide Web. <URL: http://www.daimi.aau.dk/PetriNets/>.